On the Role of Services in Enterprise Application Integration

Kostas Kontogiannis
Department of Electrical & Computer Engineering
University of Waterloo
Waterloo, Ontario, N2L 3G1, CANADA
+1 519 885 1211 ext. 2840
kostas@swen.uwaterloo.ca

Dennis Smith and Liam O'Brien
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213 USA
{dbs, lob}@sei.cmu.edu

Abstract

Recent advances in Web and middleware technologies offer a promising solution for a number of enterprise integration problems. The convergence of the Internet and distributed-object technologies, which has been referred to as the Internet's third wave, extends this "information-based" Internet to a worldwide "services-based" Web. In this paper we present a model for Enterprise Application Integration (EAI) and we discuss the role of emerging technologies, as they relate to the specification of services, registration of services, data integration, and control integration.

1. Introduction

In recent years the business world is experiencing significant changes due to the growth of new computer technologies. Modern software systems must conform to requirements, such as flexibility, adaptability, time to market, and continuous business process reengineering. Driven by these requirements, the migration of legacy systems towards new network-centric operating platforms and their consequent integration with other back-end applications using Web technologies has become a very effective strategy for many organizations to maintain a competitive edge. This strategy focuses on leveraging existing legacy software assets by minimizing the risks involved in re-implementing large-scale mission-critical legacy applications [7].

Specifically, the Web browser technology, the so-called first-wave of the Internet [8], allowed for the explosive usage of the Internet in terms of shared information resources. The convergence of the Internet and distributed-object technologies extends this "information-based" Internet to a worldwide "services-based" Web [8], where software services and content are distributed openly over the Internet, Intranets, and Extranets. Such distributed services are specified in the form of components in the Web domain and can be dynamically deployed by a Web-server upon the request of arbitrary Web clients. In this context, non-functional requirements such as interoperability, flexibility, customizability, and conformance with continuous business process reengineering activities, are fundamental for the successful deployment and integration of services in a Web-based environment.

As an example, consider a scenario in the new "services-based" Web, whereby a global infrastructure enables software components that have been developed independently, be integrated with each other in order to facilitate complex business tasks. In this way, content (data) and, software components located within an organization, can be combined on an as required basis, forming thus collaborative information systems. However, systems integration requires more than the ability to conduct simple interactions by using standard protocols. The full potential of web services [4] as an integration platform will be achieved only when applications and business processes are able to integrate their complex interactions by using a standard process integration model.

In this paper we present a prototype model for enterprise-wide service integration, and we discuss the issues and emerging technologies that support such a service integration model.

The paper is organized as follows. Section 2 and 3 discusses component-based software development and distributed architectures respectively. Section 4 presents issues in legacy system integration and enabling technologies. Section 5 presents a model for Enterprise Application Integration. In Section 6 the conclusion and further insights on research directions in the field of

Proceedings of the 10th International Workshop on Software Technology and Engineering Practice (STEP'02)
0-7695-1878-8/02 $17.00 © 2002 IEEE

Enterprise Application Integration are discussed. Finally, Section 6 concludes the paper.

2. Component Based Software Engineering

In general, a software component denotes different entities, such as: class libraries, encapsulated software modules, framework environments, CASE models, or simply existing legacy applications [12]. A software component is defined as: "a unit of composition with contractually specified interfaces and explicit context dependencies. A software component can be deployed independently and is subject to composition by third parties" [7], [13]. This definition reflects three characteristic properties of components namely that, a component is a unit of independent deployment; a component is a unit of third-party composition and; a component has no persistent state. These properties have several implications related to the use and integration of components.

For a component to be independently deployable, it must be accessible only through well-defined interfaces. Moreover, a component encapsulates highly cohesive functionality and as an atomic unit of deployment, it can not be deployed partially. In this context, a third party can not access the implementation details of all the components involved.

We can distinguish three major facets for a component [14] namely *packaging*, *functionality*, and *integrity*. The packaging perspective considers a component to be an organizational concept, focusing on the identification of a set of elements that can be reused as a unit. However, the emphasis here is on reuse. The functionality perspective considers a component to be a software entity that offers services (operations or functions) to its consumers. It also emphasizes the notion of a contract between the provider and the consumer of those services. In the major component model infrastructure standards such as DCOM, CORBA, and JavaBeans, services are grouped into coherent, contractual units called interfaces. The emphasis here is on separation of providers and users of services leading to a services-based architecture. Finally, the integrity perspective defines a component as an implementation encapsulation boundary that collectively maintains the integrity of the data it manages, and is therefore independent of this data and of the implementation of other components. This criterion is a necessary condition for components to be readily configurable. The emphasis here is on replacement and transparent upgrades, achieving thus a "plug-and-play" behavior.

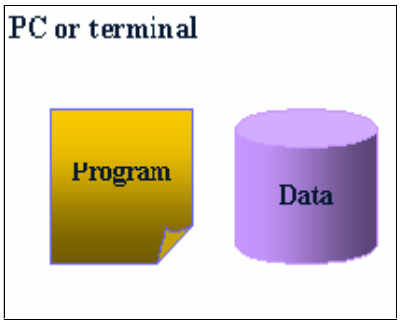

Figure 1. Monolithic Applications

3. Distributed Component-Based Applications

3.1. Enterprise Computing Architecture Evolution

There is a steadily increasing trend in the industry away from the monolithic, single processor, mainframe systems towards distributed, network centric environments. The network centric computing utilizes a LAN or WAN as the communication vehicle between components of application and leverages the computing power of more than one system.

In monolithic one-tier applications such as the one illustrated in Figure 1, all data access, business and presentation logic data are in the same machine namely, the mainframe. Common data can not be easily shared, so multiple copies of data must be consistently replicated and managed in each operating mainframe environment. This results in significant overhead for the consistency maintenance and data synchronization..

An improvement to "one-tier" architecture is the two-tier architecture, illustrated in Figure 2. This architecture is also called client/server architecture and divides a monolithic application in two components enacted as two processes, a client and a server process [18]. The server process provides access to data resources possibly through a relational database management system (RDBMS) and implements some of the business logic. The client process implements the presentation and some business logic which may be shared by the server process. Such client processes are called "fat clients".

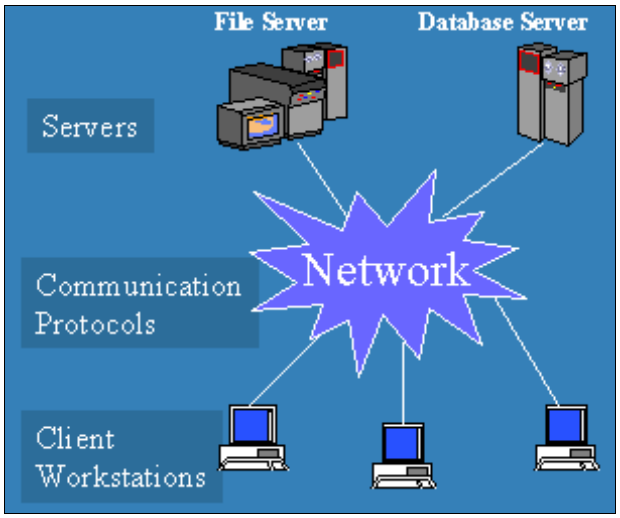

Figure 2. Two-tier Application

Figure 3. Three-tier Architecture

Finally, in a multi-tier architecture such as the one illustrated in Figure 3, the client's tasks are further reduced. Ideally, a client process implements only the user interface, for example, web browser. The business logic and services are realized by the middle-tier [15]. The databases and the repositories of the distributed components reside in the back-end to carry out the requests or data queries originating from the middle-tier. In this way, services can be deployed on-request to the clients, instead of being statically pre-installed at the point of use.

3.2 Distributed Components

With the recent advances in networking, the software components are moving from the desktop paradigm towards enterprise-wide, distributed, network-centric environments. In this respect, distributed component possesses the following characteristics [9], [16]:

- An explicit and well-defined interface defining the services it provides, and the structures of its formal parameters;
- An explicit specification for describing the behavior and usage of the component, the required execution environments, and the location of the component;
- Software independence from its clients by encapsulating the detailed implementation;
- Communication with binders for registering with the clients via the Internet, Intranets or Extranets;
- Means for remote access and remote invocation of services. and finally;
- Provision for run-time dynamic component configuration.

The major middle-ware technologies, including CORBA, RMI, Enterprise JavaBeans, and DCOM, meet some of above requirements and provide the integration environment for the distributed software components.

Overall, two factors are required for software component to inter-operate. First, they must have the common infrastructure to communicate with each other. The network and the major middle-ware components provide this service. Second, they must have access to the interfaces of the available services (methods) in each component. For example, the CORBA Interface Definition Language (IDL), indicates how services can be accessed by clients [19]

The specifications of a component's interface can be further divided into four levels [17]. The first level provides a syntactic contract and is usually specified in IDL. The IDL interface ensures that client and server can communicate across heterogeneous computing environments and operating platforms. The second level provides behavioral contracts, and is expressed in some specification language. In this case, pre- and post-conditions denoted as Boolean assertions can provide some of the operational behavior for each service offered. The third level provides synchronization contracts, and describes the dependencies between services such as, sequencing, parallelism, path expression, and synchronization constraints. Finally, the fourth level provides quality-of-service contracts and quantifies the quality of services.

4. Legacy System Integration

Integration of legacy applications with new applications is a very attractive strategy for several reasons. First, it extends the life cycle of a legacy system by making it available to new applications. Second, it hides the implementation details of the original system from other client applications. Third, it leverages existing legacy software assets by taking advantage of their mission critical functionality. Finally, it minimizes the risk and cost associated with complete re-development or application conversion.

Object wrappers play a vital role in integrating legacy applications. These wrappers can integrate the services provided by a single software component, or can be used as an integration gateway that allows for multiple resources from a diverse range of components to inter-operate. Object wrappers may also provide access to high-level system services such as, transaction services, naming services, security services, and persistence services.

In this sense, Distributed Object Computing (DOC) introduces a new paradigm for application development. In DOC, applications and services are viewed as objects with well-defined interfaces. Client applications are restricted to access the server objects only through their public interface. Similarly, DOC allows for flexibility because client applications can be added at any time, as long as they conform to the existing published interfaces.

4.1. Technologies for Web-based Legacy System Integration

The technologies to access and integrate enterprise information with the World Wide Web consist of different types of "Web gateways". Web gateways bridge the gap between Web browsers and the legacy applications and databases. At present, the approaches that are mostly used to develop Web gateways include, Common Gateway Interface (CGI), Server Side Includes (SSI), Gateways as stand-alone servers, Mobile code systems (Java gateways), Service registration languages (UDDI, WSDL), Web Service flow languages (WSFL, WSEL), Web Service invocation protocols (SOAP)

For legacy access, however, CGI and Java-based gateways, and servlets, are the most commonly used technologies [7]. The following section presents an overview of the enabling technologies for service description, registration, localization, composition, and invocation using the Web as the communication and integration medium.

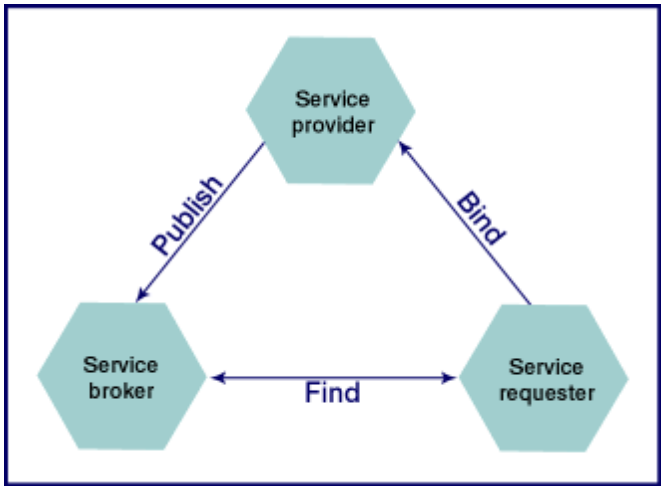

Figure 4. The Services Oriented Architecture Model

4.2 Web Services

Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the World Wide Web. The Web Services architecture describes three roles: service provider, service requester and service broker; and three basic operations: publish, find and bind. A network component can play any or all of these roles [1].

In essence, Web Services are a way of realizing *dynamic e-business*, focusing on the integration and infrastructure complexities of B2B by leveraging the benefits of Internet standards and common infrastructures [2], [3].

The model on which current web-service technologies are built is known as the Service-Oriented Architectures (SOA) (Figure 4). Technologies like UDDI, WSDL, and WSFL are based on the SOA [2].

Each of the three components -- *Service Provide*r, *Service Requester*, and *Service Broker* -- can be conceptualized as nodes on a network.

- A *Service Provider* provides a service interface to a software component that manages a specific set of tasks. A Service Provider can represent the services of a business entity or the service interface for a reusable subsystem.
- A *Service Requester* discovers and invokes software services, either locally or remotely via remote procedure calls (RPCs).
- A *Service Broker* acts as a repository for software interfaces published by service providers.

4.3. Enabling Technologies

4.3.1 XML

The *eXtensible Markup Language* 1.0 standard is a text-based markup language specification from the World Wide Web Consortium (W3C) [5]. Unlike HTML, which uses tags for describing presentation and data, XML is strictly for the definition of portable structured data. It can be used as a language for defining data descriptive languages, such as markup grammars or vocabularies and interchange formats and messaging protocols.

4.3.2 SOAP

Simple Object Access Protocol (also known as *Service-Oriented Architecture Protocol*) is an XML-based lightweight protocol for the exchange of information in a decentralized, distributed environment. SOAP defines a messaging protocol between requestor and provider objects, such that the requesting objects can perform a remote method invocation on the providing objects in an object-oriented programming fashion. SOAP forms the basis for distributed object communication in most vendor implementations of SOA. SOAP is vendor-neutral, and independent of platform, object model, operating system, and programming language [6].

4.3.3 WDSL

As the need grows within the software community to have software distributed over several systems communicate with each other, it becomes important to have a standardized way of doing so. The *Web Services Description Language* (WSDL) does just that. WSDL defines an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages [3].

4.3.4 UDDI

The Universal Description, Discovery and Integration (UDDI) standard is an XML based specification to publish and discover information about remote services. Remote services may include legacy systems wrapped using Distributed Object Technology (i.e. CORBA wrappers) [10], [11], or specific software applications that deliver specific business functionality exposed by an organization through Internet protocols. The core component of the UDDI specification is the UDDI business registration module. The business registration module consists of three major parts namely, “white pages” denoting the address, contact, identifiers for the service offering; “yellow pages” for providing a categorization of the offered service according to standard taxonomies and; “green pages” for denoting technical information about the offered services. The UDDI specification also includes programmatic interfaces so that technical details about the offered services can be linked to WSDL descriptions, and service invocation can be accomplished by utilizing the SOAP protocol. Other UDDI programmatic interfaces allow for the localization, and selection of remote services on behalf of clients given specific search criteria.

The UDDI protocol has already been implemented by various corporations. A UDDI client is available at IBM Alphaworks site [20].

4.3.5 WSFL

Using the aforementioned protocols, services can be discovered and published. What is further needed is a mechanism to describe the composition, interaction, and execution of services in a meaningful and standardized manner. WSFL deals with two types of Web Service compositions: the *Flow Model*, and the *Global Model*. The Flow Model is normally used to describe a *business process*. This is done by specifying the appropriate usage pattern of a collection of Web Services, in such a way that the resulting flow composition describes how to achieve a particular business goal. A Flow Model can be converted into a Web Service by wrapping it within a WSDL document.

The Global Model is normally used to describe overall business *partner interactions*. These interactions are modeled as links between endpoints of the Web Services interfaces. Each link corresponds to the interaction of one Web Service with another Web Service. In fact, a collection of Web Services is in itself another Web Service. Thus, WSFL provides also support for the *recursive composition* of Web Services.

4.3.6 Event Condition Action

Event-Condition-Action (ECA) is a well-investigated paradigm that has been used for modeling reactive systems and especially active databases. ECA workflows are implemented in terms of a scripting language. A script is primarily a collection of ECA rules. In a nutshell, the rule engine reacts to incoming events issued, either by the external environment or by the rule engine itself. These events trigger service requests upon the validation of specific local or global conditions, or constraints. The termination of a service may generate more events for new rules to be considered. In the context of workflow specification, the Event-Condition-Action has solid theoretical support and relates with the foundations of Extended Finite State Machines, Petri-Nets, and Logic specifications [21].

4.4 Enterprise Application Frameworks

Enterprise architectures are high level definitions of the data, applications, and technology needed to support the business. They derive from a framework originally proposed by Zachman [22] that creates a matrix with data, functions and network as the columns and a set of different views (contextual, conceptual, logical, physical and detailed representation) as rows. Planning decisions involve logical clustering of data and business processes into applications. These approaches, which are widely used, enable the important task of developing a high level integration blueprint. They also offer a method for discussing issues at different levels of detail, and for isolating information requirements from their dependencies on existing legacy systems.

These architectures provide a starting point for developing an overall scope for enterprise integration. Their results are an input to software architectures which will then provide the detailed blueprint for application development. Often however, enterprise architectures impinge on the work of software architectures, resulting in redundant and confusing effort. These issues are discussed in more detail in [23].

5. A Model for Enterprise Application Integration

Enterprise integration has the goal of providing timely and accurate exchange of consistent information between business functions to support strategic and tactical business goals in a manner that appears to be seamless. Although there have been some successes, in general there is not a clear roadmap for how to achieve effective integration of information systems. To-date, full-scale enterprise integration efforts tend to focus on inter-departmental system integration, or on Business-to-Business (B2B) applications, between organizations. In this respect, focused and incremental integration models that are applied either within specific related clusters of applications, or between sets of related clusters for resource planning (logistics, production, distribution), may be able to provide a higher return on investment and control in a better way the risks associated with large scale the reengineering, porting, and system evolution related activities.

Based on what we've learned from other models we propose a model for Enterprise Application Integration that aims to focus on the following aspects. First, we consider Requirements and Principles that refer to strategic functional and non-functional requirements that need to be met and drive the whole integration activity. Second, Business Integration that refers to the seamless collaboration between business processes to achieve specific quality and performance objectives. Third, Data Integration that refers to seamless exchange of information between applications. Fourth, Control Integration that refers to publish/subscribe protocols as well as workflow and transaction management. Fifth, Connectivity that relates to protocols for messaging and networking. Sixth, Quality Attributes that refer to user oriented product characteristics: quality, cost, and schedule. Finally, Application Integration that refers to remote invocation protocols, and middleware components.

The layout of the proposed Enterprise Application Integration model is depicted in Figure 5 while its components are discussed in detail in the following sections. Moreover, we aim for a model that can be implemented and deployed by open Web-based protocols.

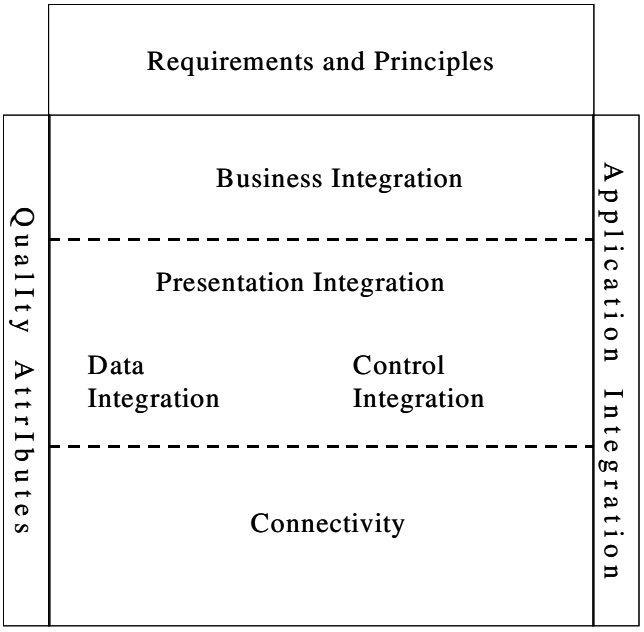

Figure 5. A Model for Enterprise Application Integration

5.1 Requirements and Principles

The state of practice in requirements engineering and management is such that both are subject to interpretation and the maturity level of the organization and the software development processes used.

To make the situation more complex, IEEE and ISO standards are rarely followed during the requirements phase of the project. In this respect, business drivers and the organization's vision, mission and goals are the main input for developing a business architecture. These inputs are obtained from strategic assessments that provide

linkage between the business drivers and down-stream design decisions. Therefore, it is important that every technological, management or strategic decision be traceable to a business driver. Moreover, every functional and non-functional requirement should also be traceable to one or more business drivers. In many cases after requirements have been generated they are often ill-represented, violated, or altered.

Organizations and development teams are progressively beginning to become aware of the need for precise requirements. Developing specific scenarios is one of the techniques for capturing and managing requirements [24]. For the specific model we present in this paper, we argue that requirements for enterprise integration should address the areas of: autonomy; security, performance, customizability, heterogeneity; transaction management, information hiding; distribution hiding and; location transparency.

Requirements engineering for Enterprise Application Integration provides the formalization of strategic, technical, and management drivers that allow for the realization of specific Business Integration model as discussed below

5.2 Business Integration

Business integration involves business process development, business process design and modeling, real-time decision support, and state management. Business processes are abstractions of user perceived models of operations, whereas business workflows become abstractions of computer-enacted models. In this respect, processes relate to the perception of the delivered function while workflows relate to the specific algorithmic sequence of steps required to perform a specific task.

Business processes are evaluated and assessed from the perspective of the business user while specialized protocols automate parts of a workflow and have a direct impact on it, by adding or eliminating some steps.

Business processes are often highlighted when companies merge, or collaborate to achieve a strategic or business objective. The merged organization usually has to alter existing processes and assist in making business process reengineering decisions in compliance with enterprise integration requirements, and objectives.

In this context, there are many questions that need to be addressed with regard to business integration. Some of these related to:

- The organization's capabilities to undertake this process. Independent assessment of capabilities might be more feasible than internal assessment.
- The primary goal of the integration. Understanding the goal can help to define what way to go.
- The boundaries and scope of the integration effort, as well as its possible side effects.

Although technology can often improve business processes and workflows sometimes it doesn't. Development teams and users alike get entangled to processes embedded in the existing systems. In many cases the documentation and the meta-data of the old and the new processes are not detailed or adequately understood. As a remedy to this problem some well-defined methodologies for modeling and documenting business process could be applied. These include, Integrated Definition (IDEF), Structured Analysis and Design, Rational Unified Process/Unified Modeling Language (RUP/UML). Moreover, the Capability Maturity Model (CMM)® has raised the bar concerning the focus on process and its importance.

Therefore, for modeling, evolving, and implementing complex business processes, organizations have to decide if it is worth changing business processes to match a particular integration solution, such as, an ERP or change the solution to match the business or a mix of both. Up to now there are no objective tools or methods an organization can use to support such a decision making process. In the proposed model, the user at the presentation level observes the effects of Business Integration. Presentation integration techniques and criteria that will be discussed below, aim to provide seamless access of contextual information and the actual realization of the business integration.

5.3 Presentation Integration

Presentation integration covers several aspects in the proposed model.

- Search and retrieval of text and graphics: To be effective, results of searches must be ranked by how well they match the search criteria e.g., *exact*, *close* or *distant* match, as well as precision and recall metrics. Technologies to assist in this area include Boolean search, Bayesian inference, fuzzy logic, genetic algorithms and natural language processing. Other techniques such as user-defined search *agents* for common or frequent searches exist and can be extended for searching distributed and heterogeneous non-text data sources, such as audio or video files.

- Knowledge mapping through taxonomies or categorization
  This complements full-text searching by allowing users to navigate information repositories. Taxonomies are being created using approaches that can be manual, automated or, a combination thereof. Moreover, industry-specific templates and wizards are also available and are classified either *a priori* (pre-defined) or *free*. Both use natural language processing, statistical algorithms and clustering techniques such as Bayesian inference, word co-occurrence, and noun-phase extraction. Metadata can also be used to establish contextual spaces that limit the computational effort for searching and assessing content. Visualization techniques are especially useful for maintainers to create visual maps of interrelated concepts. Examples of such techniques include Semio's SemioMap, Inxight's Star Tree, and ClearForest. Most such taxonomies are hierarchical however non-hierarchical ones are emerging through products like Inxight's Hyperbolic Tree which produces map-like and 3-D representations.

- Document and workflow management
  In this category, several issues are involved including physical location of information and versioning of the information. On the one hand, it is important to consider techniques for organizing and maintaining the integrity of unstructured data in documents. On the other hand is important to customize and enact in an efficient way complex workflows that automate specific business process models.

- Personalization and targeted filtering
  This area involves techniques for allowing users to specify the *types* and *amount* of content relevant to them, and its *presentation.* It requires user profiling that is, collecting information about a user's identity, interests, expertise, roles, responsibilities, and access control privileges. This is often done through software agents (called *spiders*). Collaborative filtering uses correlation matrices of users' actions and preferences to filter non-relevant information. Examples include Amazon.com and NetPerceptions for Knowledge Management.

- Collaboration
  Enables users to share data and meta-data synchronously or asynchronously. Technologies that assist in this area are tools for collaborative document editing, document change alerts, group discussions, and publishing tools to searchable repositories.

- Other system level issues
  There are many other issues involved in the presentation integration engine such as its ease of use, intuitiveness, and convenience. In this context, several criteria should be considered when assessing a presentation integration solution. Examples of such criteria deal with whether the presentation fully integrated and functional, whether it is customizable by the enterprise and personalizable by the user, whether it is cost-effective (e.g., thin client access) whether it is scalable and whether it is reliable, maintainable and secure. To address these issues several vendors are offering products and services for presentation integration including Viador, Plumtree, Epicentric, and Top Tier.

Finally, the choice of the presentation engine depends on who are the main users of the portal – customers, employees, partners, suppliers, what are the major applications that will be used by the users, what is the current state of the portal, if any – (i.e., Intranet, Extranet), the protocols employed for the users to gain access to the content sources, the use of models to catalogue and classify the offered content, the techniques used for eliminating content of redundancies, irrelevant or obsolete information and finally, on techniques used for delivering content to the end-user in a personalized, reliable and secure manner.

Related technology is available that can go a long way to achieving presentation integration. However there does not seem to be much awareness of that technology within the community and one of the main reasons for this is that the field is very new. Current implementation of presentation integration is appearing on Intranet architectures and with a few organizations developing portals in the Internet domain such as Yahoo, Excite, and iWon. Presentation Integration is based on underlying technologies that achieve data and control integration as discussed below.

5.4 Data Integration

Data integration relates to seamless exchange and interpretation of data and meta-data between different services and users that possibly operate on different contexts. Issues to be addressed in this area include the following.

- *Data syntax* - deals with the format of the data and whether or not it needs to be translated when is accessed by various services and users into possibly different contexts.
- *Data semantics* – deals with the meaning and the denotation of the data.

- *Data normalization* – deals with defining data structures and architectures that are most flexible and adaptable for the specific organizational applications.
- *Data integrity and validation* – deal with the quality, consistency and, integrity of the shared and exchanged data.
- *Data models* – deal with the development of conceptual maps facilitating data related discussions between technical and organizational users.
- *Data reengineering* – deals with transformation of data structures from an existing format to new formats

Tools and protocols for data integration are already available and well defined. In particular, Schema Transformation technologies for federated databases and XML (eXtended Markup Language) offer promising leads. However, more work is needed in the area of metrics for data integration and there is a need for better understanding of how data representation affects enterprise integration failure as it relates to data. For example there is a misconception that integration between packages and legacy data is easy. As part of integration work, data needs to be reengineered and kept current while reengineering of code, data and business processes must occur in synchronized yet incremental manner.

5.5 Control Integration

Control Integration deals with the specification, and the seamless localization, selection, messaging, and invocation of distributed and heterogeneous services. Issues to be addressed in this area include the following.

- *Events* – deal with control messages that are autonomous and asynchronous between components that are registered to receive them
- *Queuing* – deals with processing of messages that are sent to various services.
- *Transactions* – deal with an identified set of operations, which, when executed alone, constitute an indivisible operation that transforms from one consistent state into a new consistent state. Either the whole set is applied (commit) or none of it (abort)
- *Interrupts* – deal with the asynchronous transfer of control to some pre-arranged location in response to some event
- *Invocation sequencing* – deals with the sequencing of interactions between components and the enactment of workflows
- *Messaging* – deals with the invocation and data exchange protocols for the information that is passed between the various components
- *Communication modes* – deal with the type of communication between components which can be synchronous (send and wait) or asynchronous (send and no wait)
- *Service integration protocols* – deal with specification, registration, localization, selection, and invocation of distributed services from various possibly thin clients.

Certain control integration techniques such as RPC (Remote Procedure Call), point-to-point, publish/subscribe and CORBA (Common Object Request Broker Architecture), aim to address these issues. However some are based on proprietary protocols, other are heavy weight and other are too brittle to be suitable for large scale deployment, while other may not be generally applicable for heterogeneous environments. In this respect there is a need for open readily available protocols, and Web Services seem to offer some promising solution in this area. Such Web-based emerging protocols have been discussed earlier in this paper. Finally, organizations, such as Australia's CSIRO/CMIS [25], are carrying out independent evaluations of technologies to support control integration such as the IBM MQSeries, the Microsoft Message Queue, or the TIBCO Rendezvous.

Other issues that the community is aiming to address in this area are how to assess the general limitations of control integration solutions or their impact on the overall performance of the system, and how to address the steep learning curves for using and understanding the control integration technology. Data and Control Integration techniques depend on the underlying connectivity and network related protocols in the proposed model. Connectivity issues are discussed below.

5.6 Connectivity

Connectivity deals with protocols and techniques to address both process connectivity and data exchange between applications and computing environments.

Process connectivity deals with the following methods of message passing protocols such as send/receive, RPC (Remote Procedure Call), sockets, object messaging (CORBA, etc). It also deals with message handling services such as queuing, message management) and the semantics of the various protocols.

Data exchange deals with methods of transferring data between systems such as file transfer. It also deals with data replication services and data integrity.

Connectivity between systems could be handled by:

- Application Bridges and Gateways – UDDI (Universal Description, Directory Integration) and Registry. These provide services for describing, publishing and locating services.
- Application Interaction Styles – publish/subscribe, file transfer, request reply.
- Message Handling Services – queuing, security, message management, and administration.
- Basic Communications – communication protocols, point-to-point, broadcast, IP multicast IIOP/ORB, database, and the Web.

5.7 Quality Attributes

In the proposed model, quality attributes pertain to user-oriented product characteristics: quality, security, cost, and schedule. These touch upon most of the elements of the model namely, the Business Integration, Presentation Integration, Data Integration, Control Integration, and Connectivity. These quality attributes are assessed as a realization of the Requirements and Principles by which the Enterprise Integration was designed for. While cost and schedule can be predicted and controlled by mature organizational processes, process maturity does not translate automatically into product quality. If the technology is lacking, even a mature organization will have difficulty producing products with predictable performance, dependability, or other attributes. Eventually poor quality affects cost and schedule because software requires tuning, recoding, or even redesign to meet original requirements. Quality requires mature technology to predict and control attributes.

When the software architecture is specified, designers need to determine the extent to which features of the software architecture influence quality attributes and whether techniques used for one attribute may support or conflict with those of another attribute. Finally, the extent to which multiple quality attribute requirements can be satisfied simultaneously may be considered as a factor for selecting a specific implementation technique.

In this respect, there has been a significant amount of work over the years that lead to techniques for addressing selected attributes (performance and dependability), and to evaluation techniques (Markov models for analyzing performance, TMR for analyzing dependability). However, research in certain attributes, such as modifiability, is still not that mature. Moreover, some evaluation methods don't have quantifiable mathematical techniques and rely on informal techniques such as survey of users. Finally, there is a need to for better definition of quality attributes such as security, performance, modifiability, dependability, usability, and more research on how to translate business requirements and drivers into quality attribute requirements.

5.8 Application Integration

Application integration involves the interoperability of possibly diverse and heterogeneous software applications. Issues related to application integration span across several layers of the integration model as shown in Figure 5. Specifically, application integration involves business event processing issues such as automatic event notification, flow control, content routing, as well as run time application and presentation issues such as transactional integrity, context management, and transformations across presentation platforms.

In this respect, there are several research questions that need to be addressed with regard to application integration. Some of these relate to the ordering of the requirements in terms of their importance, the deployment platform, the user base, the criticality of the application, the usage profile, and whether the application is custom-made or, acquired as off the shelf components.

Clear techniques for making decisions about which application to integrate and what the nature of the problem are also not generally available. However, some standards for integration do exist such as LISI (Levels of Information System Integration), which has a model for levels of integration maturity. This could be used as a way of classifying the level of integration maturity within an organization. Other related application integration support frameworks are the C4ISR, the Architecture Tradeoff Analysis Method (ATAM) [26] for assessing architectural design decisions, and various reference distributed architectures for B2B. However the current state of the practice is ad hoc and each organization sets its own rules for defining integration. In this respect more research is needed for quantifying the impact design decisions have on an organization, and for identifying meta models for Enterprise Application Integration that link various protocols and standards. The following section presents possible research directions in this area.

6. Conclusion

The issue of enterprise application integration is a critical one for modern organizations to address. The problem is multi-faceted and will rely on marshalling an array of technology methods and tools, mature processes and organizational expertise.

This paper has focused on technology solutions, paying specific attention to the emerging role of Web services

and middleware technologies. Open, standards-based, and vendor-neutral platforms with which one can perform system integration across the enterprise provide an important prerequisite for future success.

The challenges in adopting a services based approach are also large. Web services technology is still maturing, and the underlying technologies—XML, SOAP, WSDL, and UDDI— are relatively new. From the developer point of view, there are a large number of new technologies, and engineering paradigms encompassing such disparate areas as distributed components, databases, and networks. From the customer or service-provider point of view, there are infrastructure issues that must be addressed before Web services become as ubiquitous as other operating systems services. However, more development and research is needed before service-based approaches become routine practice.

Emerging web services need to provide a technical advantage over previous solutions and provide a business advantage for the company over its competitors. Web services need to integrate well with technologies already used in existing systems—systems that are mission-critical business assets. Web services should be simple to develop and deploy, with benefits accruing even in smaller scale trials, and with results that are clear and definable. Without a critical mass of other Web service providers and requestors, exposing one's system via a Web service interface would lead to little business benefit.

References

1. http://www-106.ibm.com/developerworks/webservices/library/w-ovr/
2. http://www-106.ibm.com/developerworks/webservices/library/ws-arc1/?dwzone=webservices
3. http://www.w3.org/TR/wsdl
4. http://www-4.ibm.com/software/solutions/webservices/pdf/WPS.pdf
5. http://www.xml.org
6. http://www.w3.org/2000/xp/Group/
7. Umar, Amjad, "Application (Re)Engineering: Building Web-Based Applications and Dealing with Legacies", Prentice Hall PTR, 1997.
8. Paul Dreyfus, "The Second Wave: Netscape on Usability in the Services-Based Internet", IEEE Internet Computing, March/April 1998.
9. Cynthia Della Torre Cicalease, Shmuel Rotenstreich, "Behavioral Specification of Distributed Software Component Interfaces", Computer, July 1999 IEEE.
10. Ram Prabhu, Robert Abarbanel, "Enterprise Computing: The Java Factor", Computer, P115, June 1997 IEEE.
11. David Curtis, "Java, RMI and CORBA", http://www.omg.org, 1997.
12. "New Age Tools for Network Computing: the Future of Application Development Today", IBM White Paper, 1997.
13. Clemens Szyperski, "Component Software: Beyond Object-Oriented Programming", Addison-Wesley, 1998.
14. A. Brown, B. Barn, "Enterprise-Scale CBD: Building Complex Computer Systems From Components", In Proceedings of STEP'99, Pittsburgh, PA, September, 1999.
15. David Krieger, and Richard M. Adler, "The Emergence of Distributed Component Platforms", Computer, IEEE, March 1998.
16. Barry Keepence, Campbell McCausland, Mike Mannion, "A New Method for Identification of Reusable Software Components", 1996, IEEE.
17. Antoine Beugnard, Jean-Marc Jezequel, Noewl Plouzeau, and Damien Watkins, "Making Components Contract Aware", Computer, IEEE, July 1999.
18. Harry M. Sneed, "Encapsulating Legacy Software for Use in Client/Server Systems", Proceedings of WCRE'96, 1996, IEEE.
19. Mowbray, Thomas J., Zahavi, Ron, "The Essential CORBA: Systems Integration Using Distributed Objects", John Wiley & Sons, Inc, 1995.
20. http://www.alphaworks.ibm.com
21. N. W. Paton, Active Rules in Database Systems. Springer, New York, 1998.
22. Zachman, J. "A Framework for Information Systems Architecture," *IBM Systems Journal*, Vol 26, No. 3, 1987.
23. Smith, D., O'Brien, L. Barbacci, M, and Coallier, F. A Roadmap for Enterprise Integration. *Proceedings of STEP 2002*. IEEE Press, 2003.
24. Clements, P, Kazman, R., and Klein, M. *Evaluating Software Architectures*. Addison-Wesley, 2002.
25. http://www.cmis.csiro.au/adsat/
26. http://www.sei.cmu.edu/ata/ata_method.html