Web Service Architecture for e-Learning

Xiaohong Qiu
EECS Department, Syracuse University
Community Grids Lab, Indiana University
501 Morton N. Street, Suite 224
Bloomington, IN 47404, USA

and

Anumit Jooloor
CS Department, Indiana University
Bloomington, IN 47405, USA

ABSTRACT

Message-based Web Service architecture provides a unified approach to applications and Web Services that incorporates the flexibility of messaging and distributed components. We propose SIMD and MIMD collaboration as the general architecture of collaboration based on a Web service model, which accommodates both instructor-led learning and participatory learning. This approach derives from our message-based Model-View-Controller architecture of Web applications, comprises an event-driven Publish/Subscribe scheme, and provides effective collaboration with high interactivity of rich Web content for diverse clients over heterogeneous network environments.

Keywords: Web Service, MVC, messaging, Publish/Subscribe, SIMD, MIMD, e-education and collaboration

1. INTRODUCTION

The Internet provides a distributed infrastructure for sharing information globally with an estimate that the online population will reach 6,330 million users in 2004 [1]. The very large user market becomes a great motivation for new technologies enabling one to build the next generation of Web based applications. In particular, it is very attractive to develop collaborative applications linking of the growing number of diverse clients with rich media Web content.

This evolution brings fundamental changes to our society in communication and knowledge acquisition pattern ─ anytime, anywhere, people no longer have to meet face to face to communicate while all information is delivered to the client interface online and on demand. The new trend comprises innovative technological features: it offers a platform facilitating ubiquitous access of desktop, PDA and cellular phone (Windows, MacOS, UNIX, Linux, and PalmOS) clients; it supplies an interface with services for easier availability to global resources including data, text, 2D and 3D graphics, video/audio stream, and MP3 music; it promotes an interoperable synchronization mechanism that captures interaction between participants ─ teacher and student, trainer and trainee for real time experience.

Collaboration tools are revolutionizing the training industry. Particularly, Web-based e-Learning solutions are adopting collaboration environments that enhance accessibility to a full range of educational resources supporting rich interaction with participating parties in a synchronous or asynchronous fashion. Distance education has been successful using tools such as Audio/Video conferencing and shared curriculum using either shared display or shared event architectures. Here we explore a richer model with SIMD [2] and MIMD [3] collaboration as the general architecture of collaboration as Web Service [4] model, which can be applied to both instructor-led learning and participatory learning. The premise of this work is building forward-looking architecture of Web applications for scalability, reusability, interoperability, pervasive accessibility, and automatic collaboration.

The rest of the paper is organized as follows: in Section 2 we briefly review technical issues that cover general concept of building message-based Web Service applications, relationship of our message-based Model-View-Controller (MVC) [5] architecture of Web applications and derived SIMD and MIMD collaboration as a Web Service model for e-Learning, and methodology of our prototyping. Section 3 summarizes collaboration framework and presents SIMD and MIMD collaboration as our general architecture of collaboration as Web Service model. We mention our initial white board project exploring the interactive MIMD collaborative architecture. In section 4, we discuss performance issues. Finally, we present our conclusions and propose future work.

2. TECHNICAL ISSUES

2.1 Message-based Web Service model and messaging infrastructure

The history of Internet and Web technology saw the evolution of Web applications with architectures dominated by centralized client-server system with traditional point-to-point (unicast) connection, decentralized self-organizing peer-to-peer (P2P) system that evolved to overlay network with application level multicast mechanism, and RPC-model (e.g. CORBA) derives from method-based system calls for tightly coupled single CPU system (e.g. desktop applications) but with remote procedure calls to support the distributed objects. Client-server and P2P models are suitable for solving problems with features applicable to their patterns but real world problems can be arbitrarily complicated. Examples can be seen in parallel applications with decomposition in high dimensionality. On the other hand, RPC-like model deals well with distributed objects or components for reusability but do not scale well. Message-based Web Service model provides a unified approach that incorporates messaging flexibility with components distribution. It accommodates to the diverse and scaling nature of the Internet and also promotes Web applications development with Web Services for reusability, interoperability, and scalability.

The messaging approach decomposes a Web system into three layers: physical networks or Internet, messaging infrastructure, and Web application. This separation can greatly improve applications' portability by reducing their dependency on underlying connection topologies and platforms. It also reduces deployment overhead of Web applications. On the other hand, it requires a powerful messaging infrastructure on TCP/IP network stack providing a variety of communication services that reconcile the differences between underlying connection topologies and deployment of high-level applications. In our lab, we have developed an open source messaging infrastructure NaradaBrokering [6] that supports a publish/subscribe paradigm. It provides Java Messaging Service (JMS) [7] compliance and JXTA [8] interaction and has been applied to a suite of collaboration tools (Audio/Video conferencing [9], Carousel [10] and Anabas e-Learning platform [11]). NaradaBrokering supports multiple protocols (including TCP/IP, UDP, HTTP, and multicast), firewall tunneling, security, and heterogeneous services (e.g. messaging services and grid services).

2.2 Web application and Internet collaboration

Web application deployment shows diversified directions but have common features ─ namely, user interfaces and services for the sharing of information and resources over Internet infrastructure. The "sharing" can be done asynchronously and synchronously at every possible stage along the deployment pipeline. The objects that need to be synchronized may range from Web contents (e.g. video, audio and raw data streams), user interactions (e.g. editing operations on shared whiteboard document), distributed programs (e.g. distributed large-scale simulation components), to team participants who involve in development or management. The "sharing" can be organized through unicast or multicast style of group communication. Therefore, in the most general sense, collaboration is the core problem and service of Web applications of "sharing" although people usually refer the terminology "collaboration" to real-time synchronous Web applications with compelling time issue or constraints. As key objective of our approach, design and implementation of a uniform architecture for Web applications with automatic collaboration capability has general importance.

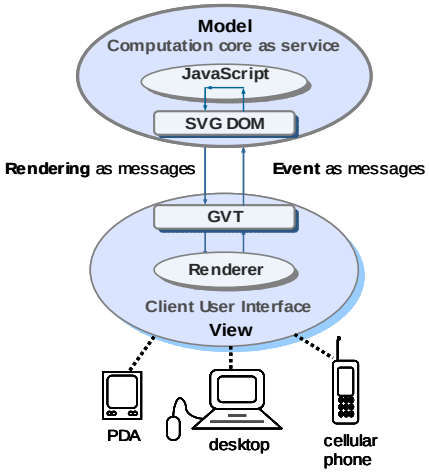

Figure1 SVG browser derived from message-based MVC

We have looked at several examples as part of systematic exploration of our design concepts: We proposed an "*explicit message-based MVC*" paradigm (MMVC) [12] as the general architecture for Web applications. It is built around systematic use of Web services and an event driven message-based separation between *model* and *view* in the MVC pattern. We have carried out initial "*collaboration as a Web Service*" [13] experiments to test viability of our architecture in supporting of interoperable applications with rich graphical contents and tight time constraints through examples of teacher-student scenario and multi-player online game. As an extension to our research scope, we converted desktop application to distributed system at architectural level. This is done through replacing conventional method-based MVC with message-based MVC in Publish/Subscribe scheme [14] for maximum reusability of existing software assets. We have in-depth discussions of performance issues for Web-based applications [15] that help to investigate the process of message-based Web application deployment

and supply feedback for the construction of underlying messaging infrastructure, which is indicative especially when this area is still immature and one expects substantial evolution. Finally, as discussed in this paper, we define collaboration with SIMD and MIMD model derived from our uniform Web Service architecture of MMVC that converge desktop and Web applications.

2.3 SVG and DOM

As key challenge of a new approach with systematic utilizing Web Service for building message-based applications, many subtle factors may not be addressed by general architectural consideration. So, we choose to build a prototype with a forward-looking architecture and conduct systematic experiments to explore and identify general principles and key implementation issues associated with this approach. Multimedia with rich graphics and media stream composition forms an important feature of new general Web applications. We select a presentational style desktop application with two-

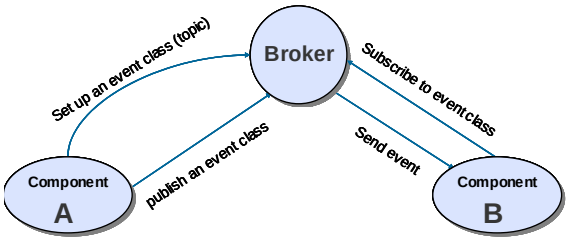

Figure2 Event-driven message-based Publish/Subscribe scheme

dimensional Scalable Vector Graphics (SVG) [16] contents ─ Batik SVG browser [17] from Apache as our testing case for further experiments and evaluations. We have restructured the open source Batik software in the explicit message-based MVC model as illustrated in fig.1. The SVG document is parsed as a Document Object Model (DOM) [18] tree with nodes representing document fragments and graphics objects. Graphical Vector Toolkit (GVT) tree reflects DOM structure and is used for rendering convenience. In this approach, NaradaBrokering supplies all messaging services (interaction between clients, events and rendering within a client) with its Publish/Subscribe architecture.

Building collaborative tools on SVG has some important general features:

a) SVG is an open source standard for 2D vector graphics of World Wide Web Consortium (W3C). It is an important technology for visualization. XML content format and scalable vector graphics feature make it an ideal choice of intermediate transportation and high-resolution rendering. It has been applied to various applications including mapping services in Geological Information System (GIS) and authoring tools (e.g. SVG viewer plug-in from Adobe and Corel). The latter feature is especially important for universal access from small wireless devices.

b) Batik SVG browser has full support of SVG 1.0 specification. The openness of both standard and implementation provides us valuable experience of a complete analysis of system structure and components interaction, which is unavailable from similar commercial tools (such as Microsoft PowerPoint, Macromedia Flash, Adobe Photoshop and Illustrator, and Corel Draw) with proprietary implementation and data format.

c) SVG is built on the W3C DOM [18], which is the natural description of all desktop documents used in Office applications of the type built by Microsoft, Macromedia and OpenOffice. Thus our work can naturally generalize to a new Web service

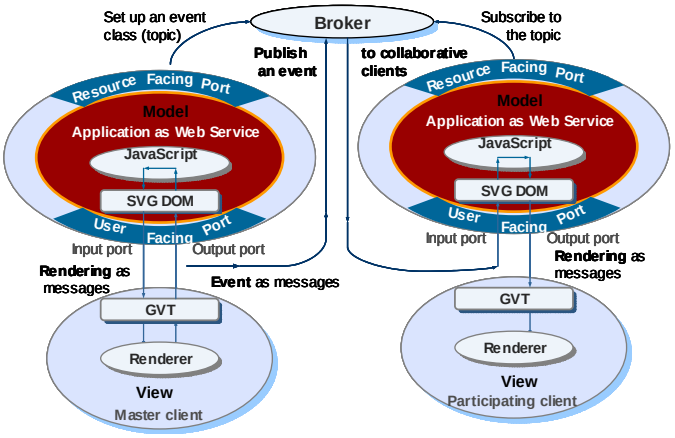

Figure3 Shared Input Port of Collaborative SVG

architecture for a next generation of desktop applications including those used for curriculum authoring. The DOM events specification provides a generic event model that propagates changes in nodes (objects) of the tree structure. This allows this style of application to have a common web service core that drives a variety of client interfaces which can either be standalone or collaborative.

3. COLLABORATION FRAMEWORK

3.1 Event-based collaboration with Publish/Subscribe scheme

Event-based programming has become a widely used programming style that supports interrupt-handling mechanisms for user input-device interactions. Compared with previous interrupt processing procedures, it promotes system modularity and asynchronous response. Most of modern desktop systems, including Microsoft Windows and their applications use MVC paradigm. The

system is decomposed into triads of *Model*, *View*, and *Controller*, which is comparable to computation core (including data structure), visual components, and the communication between those two within a separate class or inherited in either of them. In conventional event-driven method-based MVC, messaging is hidden at system level, whose differences with our method-based MVC approach are discussed in depth in Ref. [14]. Method-based event approach is also extensively used in distributed systems including Java AWT, Swing and their applications. As a common mechanism of event-based programming, event listener components subscribe to event producer component and get notification of event occurrences. In the MVC cases of Swing and Batik SVG browser, visual components (*view*) form the observers of data structure (*model*) with rendering updating corresponding to model changes.

Our approach of event-driven message-based collaboration with Publish/Subscribe scheme (see fig. 2 and fig. 3) has following implications:

- An "event" defines the incremental change of system state. We have given a complete analysis of events and classify them as UI event, SVG/DOM event, and semantic event categories in our collaboration experiments with SVG [13]. Event-based collaboration system works through timely synchronization with updated event information communicated among participatory parties. Moreover, events can be queued and stored as record for retrieval and replay and we have these services in our messaging infrastructure for supporting system reliability, Quality-of-Service and functionality.

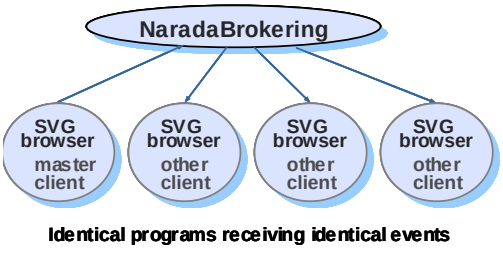

Figure 4 Monolithic collaboration

- The event workflow of a presentation style application can be illustrated by its propagation along a pipeline with stages consisting of objects (constituent system components). As shown in fig. 1, the "U-turn" trip for Batik SVG browser starts from user interaction triggering a mouse event to the completion of update rendering in image buffer. Each stage forms the natural synchronization point for collaboration. In a SVG Web Service model (fig. 3), "Input port" and "Output port" are refer to interfaces between view and user facing port of Web Service in input leg and output/rendering leg of the pipeline.
- Event-based collaboration can be implemented in method-based fashion such as those built on top of RPC-like system (e.g. CORBA). However, we adopt a different approach of event-driven message-based Web Service model with details of underlying platforms hidden in the implementation of the messaging infrastructure level. We have elaborated this in the context of our general approach of Web applications deployment in section 2.1. In our approach, communication among distributed components is conducted indirectly through messaging brokers.
- Publish/Subscribe schemes present the capability of handling complex topologies with multiple topics and multiple clients. Our messaging infrastructure provides topic management service and registration (for Publish/Subscribe) service so that the collaboration system can host virtual collaborative community activities (e.g. shared browsers, multiplayer online game, and share whiteboard) in dynamic and parallel fashion.
- Building on top of the collaboration framework, one can develop SVG applications of instructor-led (SIMD) and participatory (MIMD) programming models with Java and JavaScript. One can expect this approach be applied to other presentation style application and programming languages, and we have in our laboratory other initiatives on OpenOffice and PowerPoint.

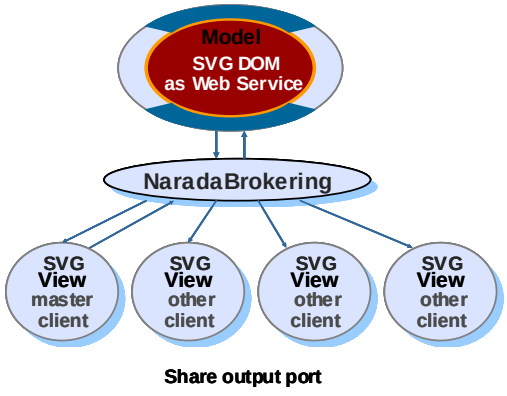

Figure 5 SIMD collaborative Web Service

3.2 Monolithic collaboration and Web Service collaboration

In this paper, we discuss two ways of building an event-based collaboration system: monolithic and Web service.

Monolithic collaboration (see fig. 4), is obtained when all participating components are formed as replications of an existing application without explicit break up into a

separate *model* and *view* component as required by the Web service architecture. This approach works through interception of the events on a master application and allows messaging broker to multicast them to the collaborating clients. It is a common strategy for collaboration systems built on top of vendor's APIs with event exposure with either proprietary or open source implementations. We have demonstrated this in our laboratory with PowerPoint and OpenOffice [20].

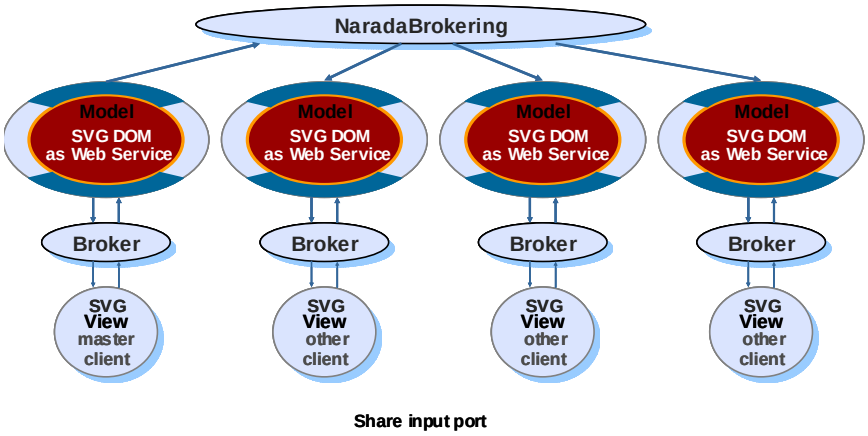

Figure 6 MIMD collaborative Web Service

We have already described the idea of "*Collaboration as a Web Service*" although at that time we demonstrated collaboration features through monolithic SVG experiments [13]. We presented already preliminary result on a collaborative SVG browser and JavaScript multiplayer chess game as a prototype to explore a general approach of collaborative Web Services. We then separated SVG into *model* and *view* components [14] and convert desktop SVG application into a distributed system. In this paper, we are rebuilding the collaboration environment with explicit Web Service models and demonstrate both SIMD and MIMD models.

3.3 SIMD collaborative Web Service

We have explained how one can make message-based network applications collaborative in two modes – *shared input port and shared output port* [19]. In each case one multicasts the messages – either those arriving at a shared input port or those produced by shared output port. Note that in each case a client assigned with "master" token has "master role". Requests for switching between different roles (e.g. "master" vs. "nonmaster" and player vs. observer) can be done dynamically. Fig. 3 illustrates the shared input port model in our architecture.

3.4 MIMD collaborative Web Service

The utilization of messaging services provided by NaradaBrokering fin our collaboration system comprises two cases: one is registration for Publish/Subscribe service in the monolithic, SIMD and MIMD (interfaces among SVG DOM Web Services); the other is the communication between separated *view* and *model* within each application component. In the latter case, the "Brokers" shown in fig. 6 run in NaradaBrokering's point to point mode supplying transport services such as security, firewall and NAT traversal, protocol choices and compression.

3.5 Example of Whiteboard

We have presented a unified collaborative architecture, which supports the above variety of SVG applications. We are applying it to an interactive whiteboard that supports SVG and Java applet interoperability with a common web service architecture. It is integrated with our existing NaradaBrokering enabled tools (shared display and audio/video conferencing) to provide interactive education system. This whiteboard illustrates the ability to tailor applications in the NaradaBrokering enabled architecture. The two views (Java and JavaScript driven SVG) with a common backend Web service illustrate the support of universal clients.

4. PERFORMANCE MEASUREMENTS

The advantages of this modular design imply performance overheads coming from the replacement of method calls by explicit messages. We have started extensive performance tests [15] with encouraging results but we still need further optimization to minimize thread scheduling and transport overheads. If the Web Service *model* and *view* are placed on nearby machines with the message broker on one of these computers, we get an overhead of about 10 milliseconds in the transport from *view* (user input) to broker to *model* and back again for rendering at the *view*. This overhead is the same for both the collaborative and standalone cases.

5. CONCLUSIONS

The SIMD collaboration model can be used for lecturing in distance education; the MIMD collaboration model would support participatory learning. Education requires different mode of interaction ranging from rather passive fashion lecturing to highly interactive and collaborative in participatory learning such as joint projects. A whiteboard represent a good example for interactive project-based learning. It allows multiple people to participate interactively together. Joint modeling projects have the same structure as the whiteboard although using different detailed tools.

We have proposed a universal modular design with messaging linkage service model that unifies support of desktop applications, Web applications, and Internet collaboration. This approach allows maximum reusability of existing components; use of a flexible messaging scheme with high scalability; automatic and effective collaboration with interactivity of rich Web content for diverse clients over heterogeneous network environments; finally it suggests a uniform interface for the next generation Web client with ubiquitous accessibility. Applied to education, our architecture enables new participatory education tools and a richer distance education environment.

The architecture presented here is being used in our laboratory in several related projects looking at collaborative desktop and visualization tools [20] and together these could enhance the e-education environment.

6. REFERENCES

[1] Global Internet Statistic http://www.glreach.com/globstats/index.php3

[2] Single Instruction Multiple Data (SIMD) at http://en.wikipedia.org/wiki/SIMD

[3] Multiple Instruction Multiple Data (MIMD) at http://en.wikipedia.org/wiki/MIMD

[4] W3C Web Service Description Language at http://www.w3.org/TR/wsdl

[5] G. Lee, **Object oriented GUI application development,** Prentice Hall, 1994. ISBN: 0-13-363086-2.

[6] Community Grids Lab NaradaBrokering system at http://www.naradabrokering.org

[7] Sun Microsystems Java Message Service at http://www.hostj2ee.com/specs/jms1_0_2-spec.pdf

[8] Sun Microsystems JXTA at http://www.jxta.org/

[9] Geoffrey Fox, Wenjun Wu, Ahmet Uyar, Hasan Bulut, Shrideep Pallickara, "*A Web Services Framework for Collaboration and Videoconferencing"*, WACE 2003 Workshop on Advanced Collaborative Environments Seattle June 22 2003 http://grids.ucs.indiana.edu/ptliupages/publications/wace-submissionjune-03.pdf

[10] Community Grids Lab Carousel project at http://grids.ucs.indiana.edu/ptliupages/projects/carousel/

[11] Anabas Conferencing system http://www.anabas.com

[12] Xiaohong Qiu, Bryan Carpenter and Geoffrey C. Fox, "*Internet Collaboration using the W3C Document Object Model"*, Proceedings of the 2003 International Conference on Internet Computing, Las Vegas June 2003 http://grids.ucs.indiana.edu/ptliupages/publications/collaborative_dom_conference_2003_Int_IC_font10_without_title_page.pdf

[13] Xiaohong Qiu, Bryan Carpenter and Geoffrey C. Fox, "*Collaborative SVG as A Web Service"*, Proceedings of SVG Open, Vancouver, Canada, July 2003 http://www.svgopen.org/2003/papers/CollaborativeSVGasAWebService/#S.Bibliography (requires SVG viewer plug-in http://www.adobe.com/svg/viewer/install/main.html for displaying figures)

[14] Xiaohong Qiu, "*Building Desktop Applications with Web Service in a Message-based MVC Paradigm"*, to appear in IEEE International Conference on Web Services, San Diego, California, July 2004

[15] Xiaohong Qiu, Shrideep Pallickara, and Ahmet Uyar, "*Making SVG a Web Service in a Message-based MVC Architecture",* submitted for publication

[16] W3C Scalable Vector Graphics (SVG) version 1.0 Specification http://www.w3.org/TR/SVG/.

[17] Apache Batik SVG Toolkit http://xml.apache.org/batik/

[18] W3C Document Object Model (DOM) level 1 specification http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

[19] Geoffrey Fox, Dennis Gannon, Sung-Hoon Ko, Sangmi Lee, Shrideep Pallickara, Marlon Pierce, Xiaohong Qiu, Xi Rao, Ahmet Uyar, Minjun Wang, Wenjun Wu, "*Peer-to-Peer Grids"*, Chapter 18 of **Grid Computing: Making the Global Infrastructure a Reality** edited by Fran Berman, Geoffrey Fox and Tony Hey, John Wiley & Sons, Chichester, England, ISBN 0-470-85319-0, March 2003. http://www.grid2002.org]

[20] Minjun Wang, Geoffrey Fox and Shrideep Pallickara, "*A Demonstration of Collaborative Web Services and Peer-to-Peer Grids"* to appear in proceedings of IEEE ITCC2004 International, Las Vegas April 5-7 2004. http://grids.ucs.indiana.edu/ptliupages/publications/wangm_collaborative.pdf