Making SVG a Web Service in a Message-based MVC Architecture

Xiaohong Qiu, Shrideep Pallickara, and Ahmet Uyar

Keywords
SVG, Web service, message, MVC, and publish/subscribe

Introduction

Web Services oriented architecture with loosely coupled messages is becoming an increasingly important feature in the deployment of Internet applications. The broad applicability of this approach includes enterprise software, e-Learning, e-Science and e-Business. This paper describes in design and implementation how we make SVG a Web Service in a message-based Model-View-Controller (MVC) architecture with publish/subscribe scheme. The work converts Batik SVG browser, a desktop application, into a distributed system. Specifically, it includes decomposition of Batik SVG browser into separate "*View*" and "*Model*" components; modification of its architecture from traditional method-based MVC into message-based MVC. "*View*" including client interface components (Swing GUI and GVT rendering) is dynamically downloaded to client. "*Model*" consisting of DOM and JavaScript modules (see fig. 1) naturally becomes a Web Service running on a Web server. Event-based messages, which communicate through our messaging infrastructure (NaradaBrokering), play the role of "*Controller*". We have started an extensive series of performance measurements to demonstrate the viability of our approach and identify some key issues influencing the performance of message-based Web applications.

Methodology

Web applications centered around messages can achieve important features such as scalability. With Moore's Law's prediction still holds and network bandwidth keep growing, inevitable performance improvements are in favor of making the messaging approach more attractive. The main challenge is to exploit this concept in design and implementation to provide scalable interoperable systems. We investigate a universal modular design with messaging linkage service model that converge desktop applications, Web applications, and Internet collaboration to achieve reusability, scalability, interoperability and pervasive accessibility. We have systematically carried out our work through the following steps:

1) proposing an "explicit message-based MVC" paradigm (MMVC) as the general architecture for Web applications;
2) presenting an approach of building "collaboration as a Web Service" through monolithic SVG experiments;
3) bridging the gap between desktop and Web application by leveraging the existing desktop application with a Web service interface through "MMVC in a publish/subscribe scheme" [1];
4) identifying some key issues that influence message-based Web applications with a detailed analysis of performance in a presentational style application experiment with rich Web content and high client interactivity, which is the focus of this paper;
5) presenting SIMD and MIMD collaboration as the general architecture of "collaboration as a Web service" model [2];

Message-based MVC and SVG

MVC is a frequently used paradigm for almost all modern desktop architecture design including Microsoft Windows operating system. Traditionally, desktop applications employ MVC paradigm in method-based interactions between the components to achieve high performance for interactive applications. Publish/subscribe scheme enables event-based programming to link event source component and event listeners' components asynchronously through callback methods while event messages are hidden at system level. This approach is widely used in object-oriented systems including java AWT, Swing, and

applications built on top of them such as Batik SVG browser. We propose a different approach of "*explicit message-based MVC*" paradigm for applications deployment, which pulls up the hidden messages from system level to application level. By doing so, the tightly coupled connections between different parts of an application are replaced by a loosely coupled messaging linkage service model which has flexibility of scaling.

As our approach is based on investigation of MVC paradigm and messaged-based Web services, which are fundamental design models from desktop to Web applications, deployment of a uniform architecture for desktop and Web applications with automatic collaboration capability has general importance and we have detailed discussions of the design principles in another paper [1]. In this paper, we will provide our solutions to the following questions with focus on SVG implementation:

- Can MVC be implemented in a message-based fashion?
- What is the performance of the message-based MVC and what factors influence it?
- How does it depend on the operating system, the application, machines and network?
- What is the relationship of collaboration and Web services with MVC paradigm?
- How easy is it to covert an existing application to message-based MVC?
- What are the architectural and implementation principles to be used in building applications from scratch in a message-based MVC paradigm?

We have a complete analysis of constituent components and their interactive relationship of Batik SVG browser. The logistic components can be decomposed into a three-stage pipeline, as illustrated in fig. 1. Theoretically, any parts with natural event linkage between client user interface and computation core can produce web services coordinated in a single application. We have substantial experimentations in finding the decomposition point, which is mainly based on considerations to keep system functionalities while avoiding excessive re-construction. We chose to split the SVG browser between the DOM and GVT tree, which allows generalization to other DOM applications.

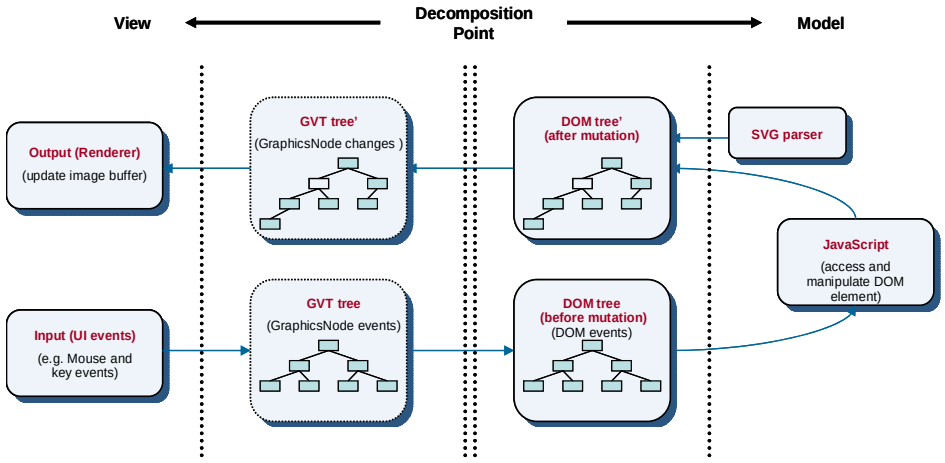

Figure 4 Decomposition of SVG browser into stages of pipeline

Performance

We have designed a series of performance measurements to test the viability of our approach. There are many variables including position of Model, View, and Event Broker (NaradaBrokering) and the choice of type of host computer and network connection. One can also vary the application running in the Model Web service. One can investigate either the single Model and View or the collaborative models. We expect the main impact to be the algorithmic effect of breaking the code into two, the network and broker overhead, and thread scheduling intervenes of operating system between interfaces of SVG application and messaging brokers. Our initial testing results show the client to server and back transit time is only 20% of the total processing time in the local examples.

Conclusions

We've present a uniform architecture with message-based MVC service model which unifies desktop and Web applications. Other research is undergoing in our laboratory in extension of these ideas to more presentation style applications including OpenOffice and PowerPoint using APIs. In our final paper, we will provide detailed performance optimization and analysis to SVG Web service application.

References

1) 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
2) Xiaohong Qiu and Anumit Jooloor *Web Services Architecture for e-Learning* to appear in International Conference on Education and Information Systems: Technologies and Applications, Orlando, Florida, July 2004