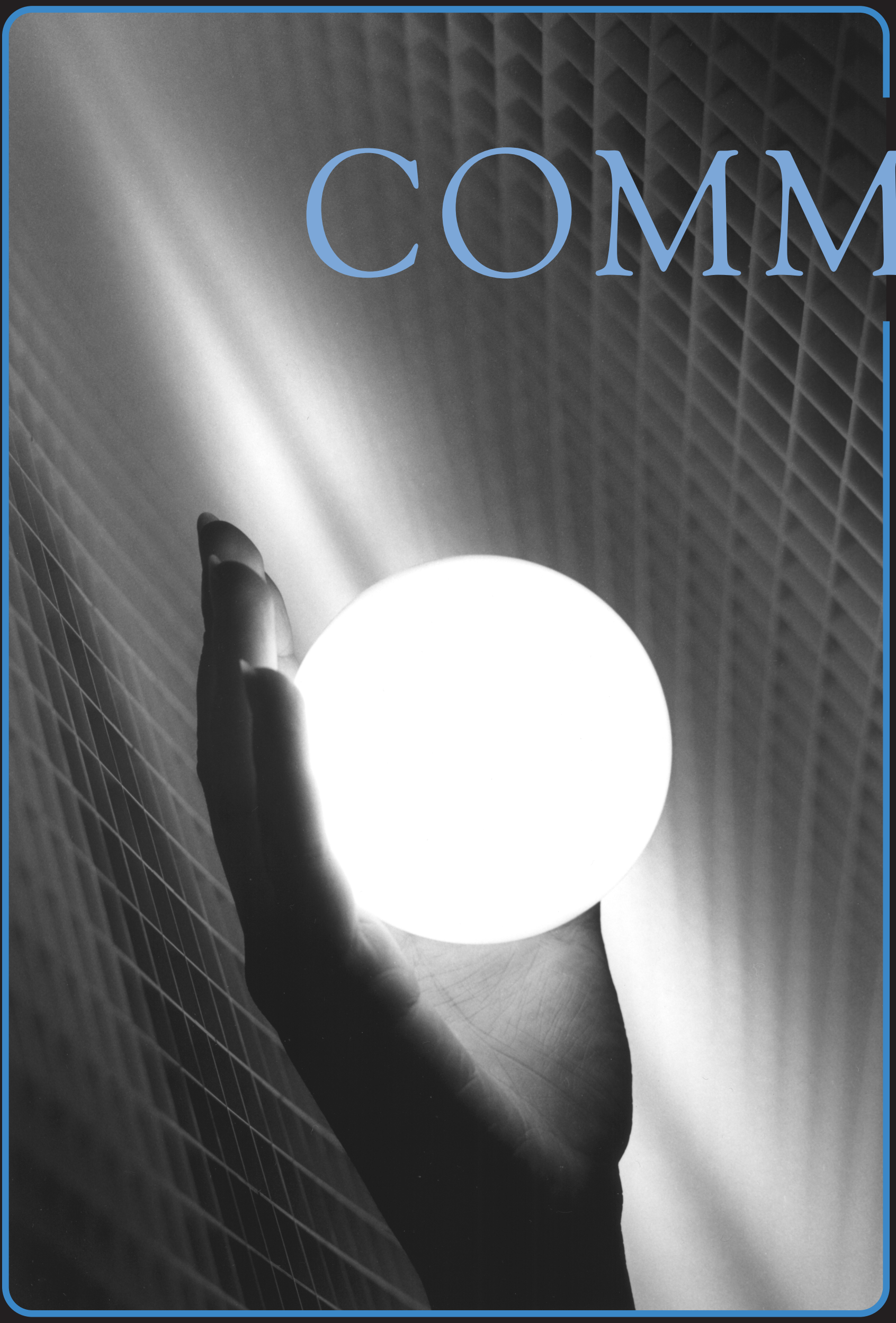

# COMMUNITY GRIDS LAB

www.pervasive.iu.edu
www.svgarena.org

pervasivetechnologylabs
AT INDIANA UNIVERSITY

## Collaborative SVG as a Web Service

We present an approach to make SVG into a Web Service, and to build synchronous collaborative SVG applications as Web services with a message-based event model. By integrating with the messaging environment Narada Brokering and the collaborative session control service XGSP, we show a general way of building W3C DOM based applications as collaborative Web Services. This work has two major goals; firstly it allows us to make a very flexible collaborative SVG browser exploiting our general approach for collaborative Web Services. Secondly it shows how applications and in particular W3C DOM based applications can be built as Web Services in our case utilizing Batik SVG Browser, which has a well written open source code. We have prototyped a shared SVG browser and a collaborative chess game with SVG to demonstrate our design concepts.

(To view full paper, see: *www.svgopen.org/2003/papers/CollaborativeSVGasAWebService*)

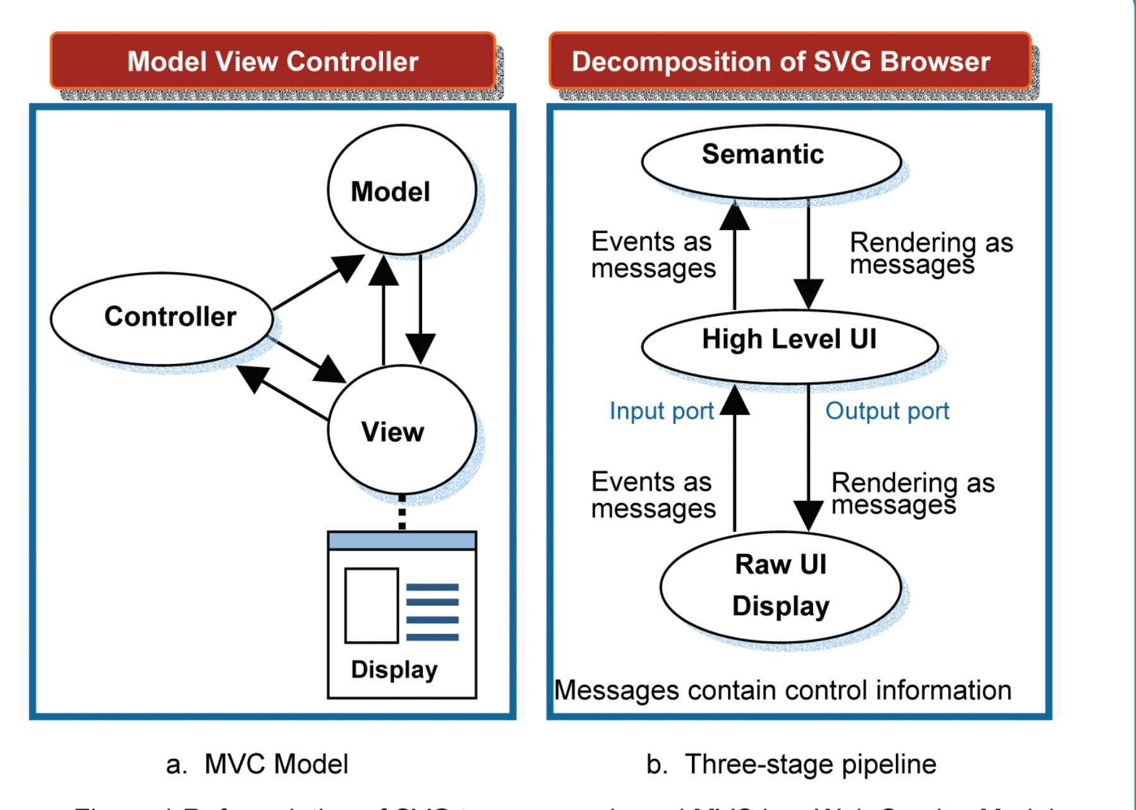

Figure 1 Reformulation of SVG to message based MVC in a Web Service Model

## Decomposition of SVG Browser

Key features of the architecture of SVG and related applications can be derived from the Model-View-Controller picture. The Model component essentially becomes the Web Service while the View becomes the user interface. They are linked by the NaradaBrokering publish/subscribe messaging system; the combination of this with the preparation and interpretation of messages corresponds to the Controller MVC component. We analyze all possible events of the SVG browser and divide them into three types corresponding to the three stages of the pipeline in b). The event types are Raw Events (low level events including mouse and keyboard events), High Level UI Events (DOM/SVG events) and Semantic Events (application events such as shared SVG browser "Open file" events). Raw events are generated in the View and are converted into messages for the Model. One can design different View modules (with trade-offs in complexity and performance) through choice of which High Level UI events and semantic events to process in the Model and which in the View component.

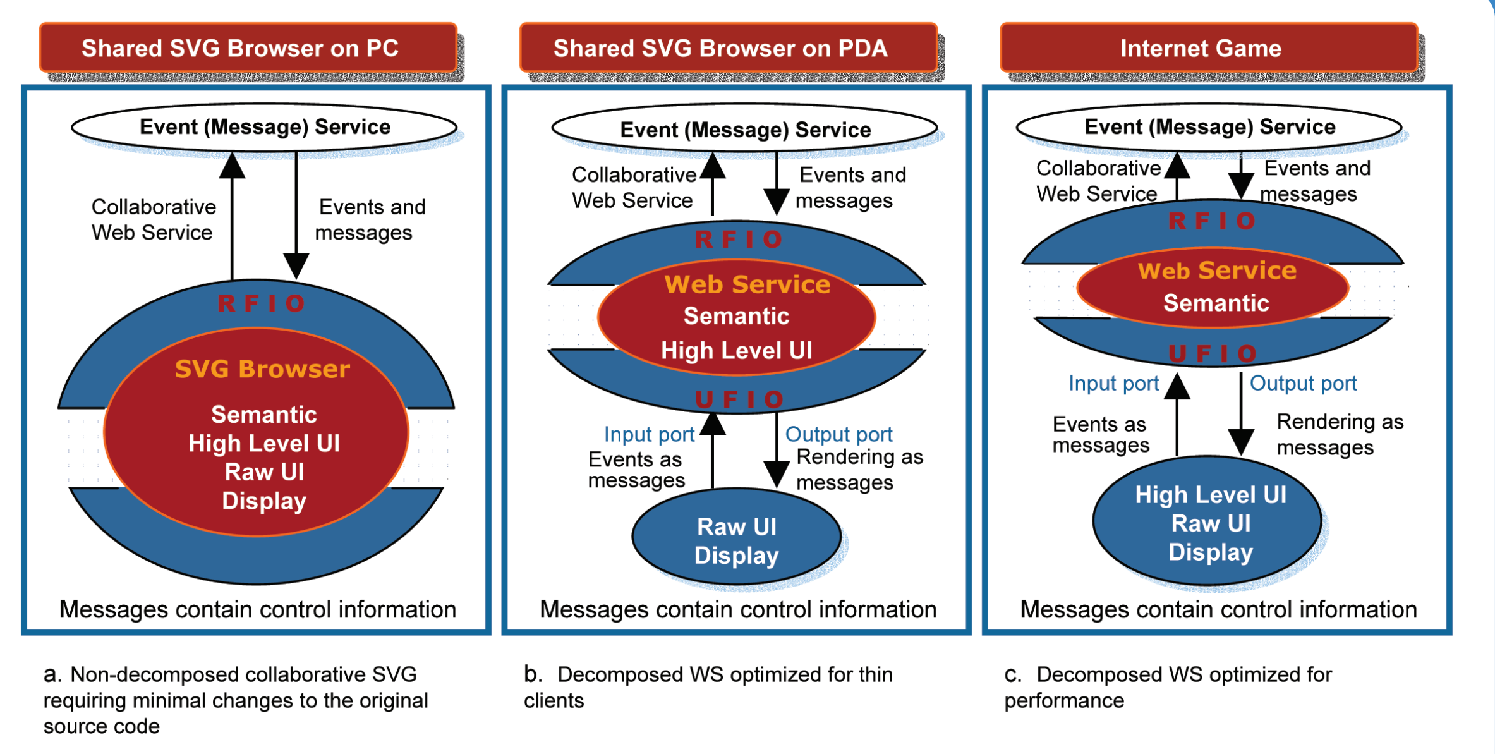

Figure 2 Three among the different ways of decomposing SVG between client and Web Service component

## Collaborative SVG as a Web Service Model

We take the approach that every resource (a data file, a video/audio stream, or even a piece of software like a SVG browser) is an object. A Web Service defines a message based interface for sharing of those distributed objects. In our case here, we share SVG synchronously and make it a Web Service interacting with clients and other applications or Web Services. We define RFIO (Facing Input and Output ) ports and UFIO (User Facing Input and Output ) ports to distinguish distributed and local service interfaces that surround the Web Service.

We use three collaborative SVG applications to illustrate how we build systems based on different ways of decomposing SVG in the model These are not the only possibilities and in particular one could package SVG as three separate modules as in the previous picture; we only discuss cases here with one or two independent modules. For a shared SVG browser application, we assume a master and participating client scenario. At any time in a collaborative session, there's only one client assigned the "master role" (such as a teacher or player to move in a game of chess) and the others are participating clients (such as students or observers and the player who has just moved in chess).

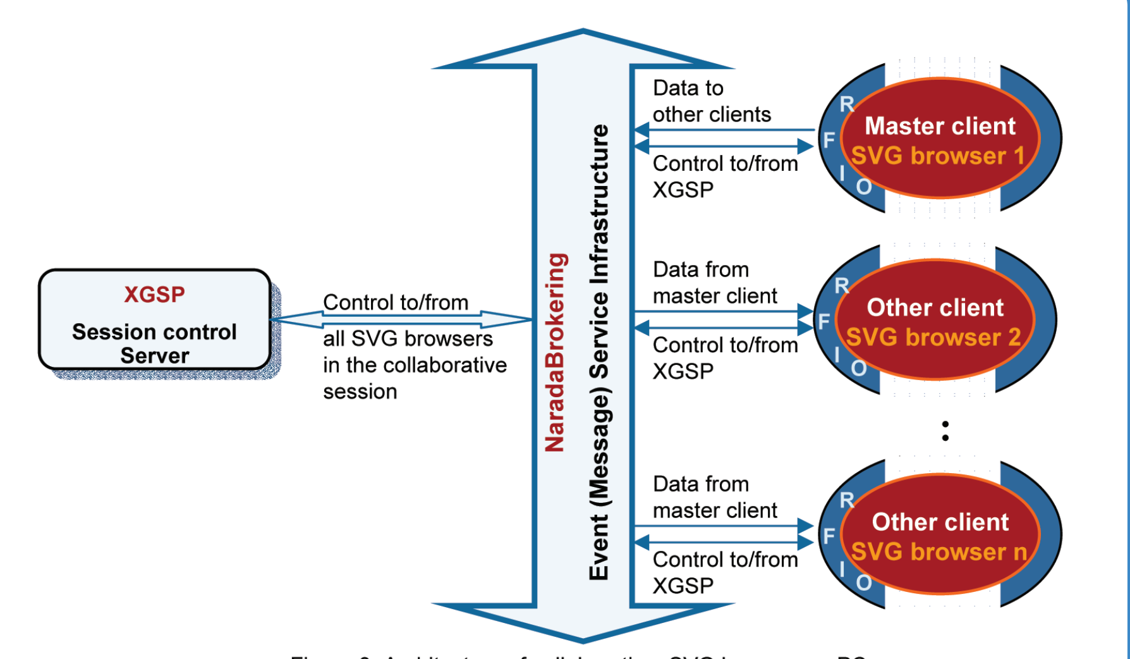
XGSP
Session control Server
Control to/from all SVG browsers in the collaborative session
NaradaBrokering
Event (Message) Service Infrastructure
Data to other clients
Control to/from XGSP
Master client SVG browser 1
Data from master client
Control to/from XGSP
Other client SVG browser 2
Data from master client
Control to/from XGSP
Other client SVG browser n

Figure 3 Architecture of collaborative SVG browser on PC

## Architecture of shared SVG browser

This example shows the way to make a client side SVG browser collaborative in a synchronous fashion and how to integrate it into the collaborative environment. We assume that all clients have the same version of the SVG application so the interpretation and process of semantic events generate consistent results. We can use the messaging framework to either check this for running clients or instantiate the correct version from an SVG "factory" service as for example defined in the OGSI (Open Grid Service Infrastructure) standard. However we did not implement this yet. At any time, only one client is assigned a master role. The XGSP session control server establishes a collaborative session allowing clients to join and responds to requests (such as changing role from participating client to master or vice versa) from the clients. The NaradaBrokering system provides the publish/subscribe mechanism for the clients to be integrated together under the same topic session. It provides the infrastructure for all messages – master events are captured and sent to NaradaBrokering and multicast to other participating clients; control messages (request and response) are delivered between clients and XGSP. In our architecture all messages (low bandwidth control, high bandwidth TCP/IP data exchange and high bandwidth UDP audio-video conferencing) are handled by NaradaBrokering. This allows a uniform approach to transport optimization (such as firewalls), session management (through topics) and fault-tolerance (through agents monitoring publish-subscribe session).