GroupScape: Integrating Synchronous Groupware and the World Wide Web

T.C. Nicholas Graham

Department of Computer Science
York University
4700 Keele St., North York
CANADA
graham@cs.yorku.ca

ABSTRACT Synchronous groupware applications support people collaborating in real time over a distance. The world wide web supports asynchronous collaboration by allowing people to share distributed information repositories. This paper presents a new technique for creating applications that tightly integrate synchronous groupware with the world wide web. The key points of the technique are: two new HTML tags allow synchronous views to be embedded within WWW pages *without programming*; lightweight connection of WWW documents and applications is achieved through the use of *constraints*, and the use of the model-view-controller architecture allows easy integration of applications and WWW pages that were developed separately. This technique has been demonstrated in the context of the new multiuser *GroupScape* HTML browser, developed using the *Clock* groupware development toolkit.

KEYWORDS CSCW, Groupware Development, HTML, WWW

1. INTRODUCTION

Synchronous groupware applications support people collaborating in real time over a distance. Example applications include multiuser text editors, teleconferencing applications and collaborative software engineering tools. The defining property of *synchronous* groupware is that it helps people to work together at the same time, allowing participants to immediately see the effects of other participants' actions. Synchronous groupware is meant to create *group awareness*, allowing people to work with the kind of direct communication they would have if they were all in the same room.

The world wide web (WWW) supports a different kind of collaboration, allowing distributed groups to share information, but not supporting the direct person-to-person communication of synchronous groupware. For example, the WWW has been effectively used to support the collaborative development of documentation in a software reengineering project (Finnigan et al., 1997) and in the collaborative development of an information repository on the Boreal forest (Freemantle et al., 1996). The web therefore supports *asynchronous* group work.

With the growth of the WWW as a dominant technology for information dissemination and collaboration, it is natural to want to integrate synchronous and asynchronous collaboration in a WWW-based tool. Such integration should support people in collaboratively finding and analyzing information on the WWW (Greenberg and Roseman, 1996) and should support the integration of web browsing and applications. Current approaches typically provide restricted integration between web pages and synchronous applications (Rees and Woo, 1994) or require complicated programming in CGI, Java or Olé (Rex, 1996).

This paper presents a new technique for tightly integrating synchronous groupware applications with the

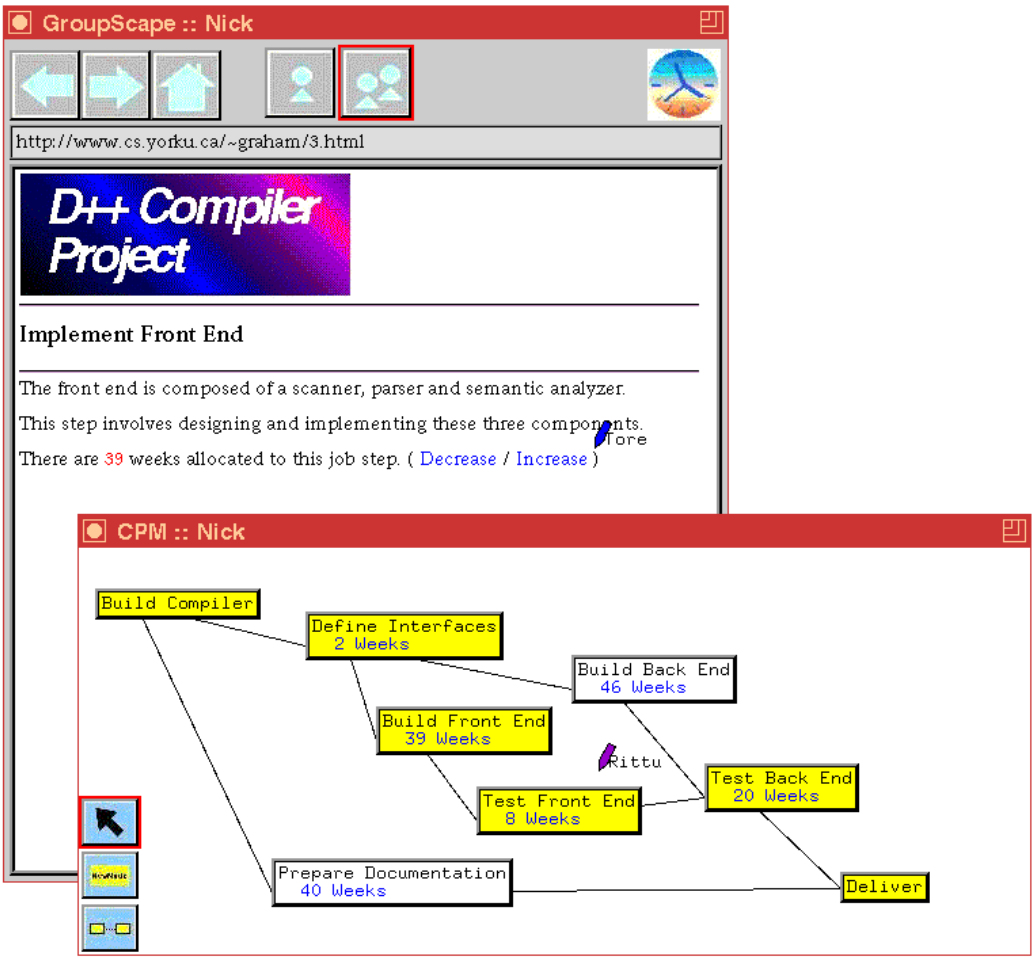

Figure 1: A collaborative project scheduling tool integrating the GroupScape web browser and a synchronous critical path planning application.

world wide web. The technique provides tight coupling of WWW pages and synchronous applications without programming. The technique has three features:

- An object-oriented design pattern, based on the model-view-controller architecture (Krasner and Pope, 1988), is used to permit easy integration of synchronous groupware and a multiuser web browser;
- To bring synchronous functionality into web documents, two new HTML tags are introduced. These tags support synchronous functionality without programming;
- The connection of synchronous applications and web documents is achieved through declarative constraints, allowing a web browser to be connected to synchronous applications without programming.

To demonstrate this technique, we have developed the *GroupScape* browser, a multiuser HTML browser supporting our extended HTML language. GroupScape itself was written using the Clock groupware development tool (Graham et al., 1997).

The paper is structured as follows. The next section introduces an example applicatison to motivate the problems in integrating a synchronous application with the WWW. The following sections introduce the design pattern used to structure applications, the extended HTML language, and the use of constraints to connect the GroupScape browser to standalone applications.

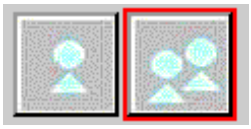

Figure 2: Users can select between private and group browsing modes

2. A GROUPWARE PROJECT SCHEDULER

Figure 1 shows an example application integrating synchronous groupware and the world wide web. The application allows a group of people to schedule resources in a fictional compiler development project. The application window shows the steps involved in carrying out the project and the number of weeks allocated to each step. The critical path through the network is shown in grey. The participants in the planning session can add new nodes to the network, connect nodes and move nodes. This is an example of a synchronous groupware application since all participants see the same network, and changes perfomed by one participant are immediately reflected on the displays of the other participants. When this application is augmented by a voice channel (e.g., using a telephone), managers in the compiler development project can use it to negotiate the allocation of project resources. To help increase group awareness, telepointers are used to show the positions of the other participants' pointers.

The GroupScape browser window is used to view information describing the project phases. By default, all participants see the same page, and all navigation commands apply to the displays of all users. If a participant selects the private viewing mode, however, his/her browser decouples from the shared page, allowing browsing independent of the other participants (figure 2). The telepointers from the application window also apply to the GroupScape window, allowing participants to see who else is viewing the same WWW page as they are.

The application window and the GroupScape browser are connected in several ways to allow them both to be used in the planning and negotiation process:

- Whenever a participant clicks on a node in the network, the page describing that node is displayed in the browser on all participants' displays;
- In the browser, the text describing a given project step contains the number of weeks allocated to the project. When the allocation is changed, the value displayed in the browser changes in real time;
- In the browser, buttons allow allocations to be increased or decreased. When these buttons are clicked, the allocation on the WWW page is changed, the allocation on the node in the application window is changed, and the critical path is updated as necessary. These effects are carried out in real time on the displays of all participants.

2.1 Requirements of Integration

This application contains a set of features that serve to motivate the ways in which synchronous group interaction may be integrated with web browsing. These forms of integration are:

Group web browsing: Facilities should be available to allow a group to browse the web together. Participants should be able to synchronize the pages they are viewing, to drop out of the group and rejoin it, and to easily see who else is on the same page.

Application based browsing: It should be possible for the groupware application to control which page is being viewed. In our example, clicking on a node in the critical path network directs GroupScape to load that page.

Embedded synchronous views: It should be possible to embed application-dependent views in web pages. These views should be updated in real time, without requiring the page to be reloaded. In our example, the number of weeks allocated to each job step is embedded as a synchronously updated view.

Embedded application events: It should be possible for web pages to control the application. In our example, web pages have embedded hot areas allowing participants to modify the time allocated to job steps.

This paper presents a technique for performing all these styles of integration without programming. Section 3 presents an object-oriented design pattern for integrating the GroupScape browser with synchronous groupware applications. This pattern is expressed in the *Clock* groupware development language (Graham et al., 1997), but should be adaptable to other object-oriented user interface development languages. Section 4 then shows how extending HTML with two new constraint tags makes it easy to introduce application functionality into web pages without programming.

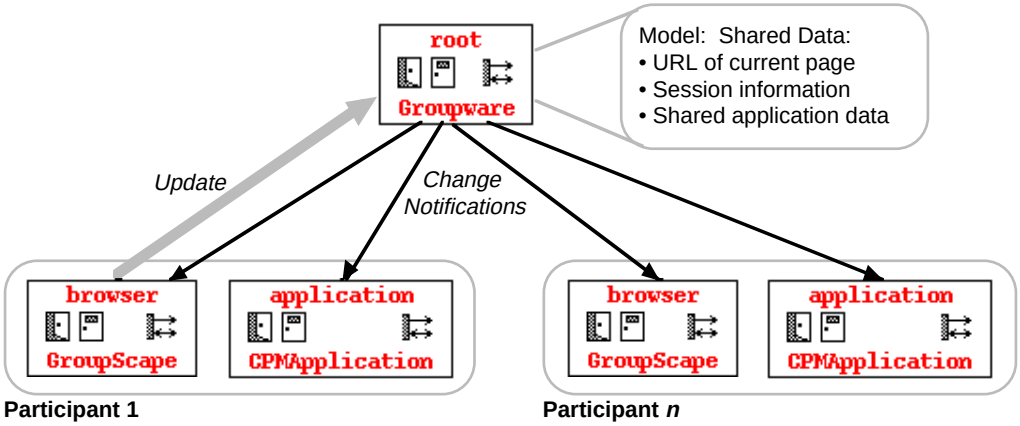

Figure 3: An MVC-based run-time architecture integrating GroupScape and synchronous applications. Each participant node contains view and controller components implementing that participant's user interface.

3. INTEGRATION ARCHITECTURE

In order to integrate web browsing and synchronous applications, we adopt a layered *model-view-controller* (MVC) architecture (Krasner and Pope, 1988) for the run-time organization of the multiuser browsing and application session. This architecture reduces the direct links between components in the session, simplifying the programming of inter-component communication. To allow the use of this model without programming, an object-oriented *design pattern* is provided that captures the details of the MVC communication protocol. To link a synchronous groupware application to the GroupScape browser, the programmer simply instantiates the pattern with the application. As will be seen in section 4, the GroupScape browser recognizes two new *constraint tags* that allow the browser to interact with the application without programming.

This design pattern was developed in the Clock groupware development toolkit, but should be easily adaptable to other object-oriented languages.

3.1. Run-Time Architecture

MVC has proven to be a natural way of organizing groupware applications, and can be implemented efficiently in a distributed context (Graham et al.,1996b, Kindberg et al., 1996). MVC allows the GroupScape browser to be connected to applications without any direct communication between the application and the browser. This means that GroupScape can be integrated with applications without reprogramming either the application or GroupScape itself.

To demonstrate how this architecture works, figure 3 shows the run-time organization of the integrated scheduling application of figure 1. A *model* component implements all data shared by the application and the browsers of the participants in the groupware session. The model contains information about who is in the session, the URL of the page currently being viewed, and shared application information such as the structure of the critical path network and the number of weeks allocated to each job step. For each participant, an instance of the application and of the GroupScape browser is connected to the model.

All communication between components in the run-time architecture is performed through the model. When a participant performs an action, an update is sent to the model to indicate the change that has taken place. The model then notifies the application and browser components of all participants that they must update their displays. For example, if a participant decreases the number of weeks allocated to a job step (by clicking on the "decrease" button in the browser), an update is sent to the model to set a new time for that job step, and all participants are notified of the change.

The advantage of this architecture is that the browser and application do not need to communicate directly. This means that the browser does not need to know what kind of application it has been connected to, or how to communicate with that application. Section 4 shows how this allows us to use constraint tags to embed application knowledge in the browser without programming.

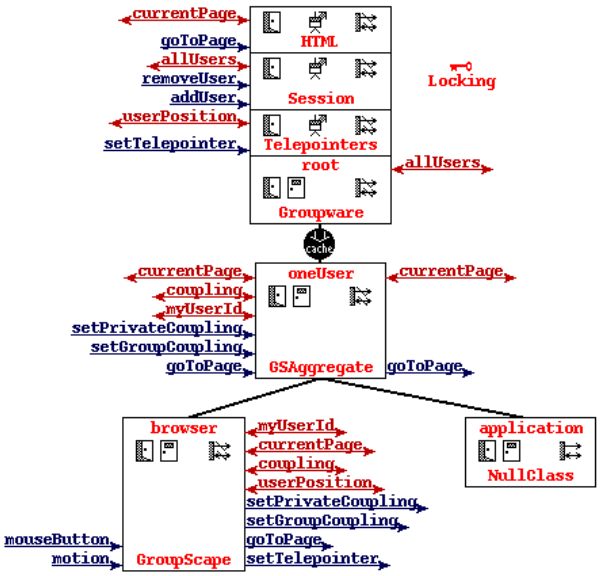

Figure 4: An object-oriented design pattern for coupling GroupScape to synchronous applications.

3.2. A Pattern for Integration

Object-oriented design patterns encode standard ways of organizing programs (Beck and Johnson, 1994). Patterns help programmers apply an existing program organization in new contexts by showing what program components are required, and how they are parameterized to the new context.

Figure 4 shows a pattern for integrating the GroupScape browser with synchronous groupware applications. This pattern was developed using the visual *ClockWorks* programming environment (Graham et al.,1996a), and was used to implement the application of figure 1. In ClockWorks, patterns are available from a library, and can be instantiated and manipulated visually.

In the pattern, the *Groupware* component is used to implement the shared data of the model. The abstract data types implementing this data are attached to the component: the *HTML* ADT represents the URL of the current page, with methods *goToPage* to set the current page and *currentPage* to return the current URL; the *Session* ADT represents information on the current participants in the session, and the *Telepointers* ADT represents the positions of each participant's mouse pointer.

For each participant, an instance of the *GSAggregate* class is created, in turn creating an instance of the browser and the application. *GSAggregate* is responsible for maintaining the coupling status of the browser, supporting group and private navigation.

The *GroupScape* component implements the multiuser browser. The browser always displays the page at URL *currentPage*, so that whenever the current page is modified in the *HTML* ADT, the browser automatically updates the page displayed.

The application itself is given class *NullClass* in the pattern. The application is a *parameter* to the pattern that must be filled in when the pattern is instantiated. To integrate an application with the browser, the programmer instantiates the pattern, fills in the correct application class, and makes any shared application data visible in the model. Normally, this can be done with no programming whatsover, simply by positioning objects in the visual environment.

This pattern encapsulates the difficult programming problems of the run-time architecture of figure 3, so that programmers do not have to solve them every time the browser is integrated with a groupware application. In order to instantiate the pattern, the programmer must fill in the class of the *application* component, and add any new shared ADT's to the model. In ClockWorks, this can be done purely visually, with no programming whatsoever.

4. CONSTRAINT TAGS

In the application of figure 1, the GroupScape browser is used to display pages that are integrated with the project planning application. These pages contained views that needed to be updated in real time (the time allocated to each project) and hot areas that invoked functionality in the application (the *increase* and *decrease* buttons.) Using the GroupScape pattern, this linkage is achieved by embedding special constraint tags in the HTML document being viewed, requiring no modification of the browser code itself. The constraint tags are: the CLOCKVIEW tag which allows a synchronous view to be embedded in an HTML document, and the CLOCKUPDT tag, which allows hot areas to be defined that will cause updates to the application state.

Figure 6 shows the complete HTML code for the document displayed in figure 1. This document contains a description for step "3" in the project, the step of implementing the front-end of the compiler. Other than the new CLOCKVIEW and CLOCKUPDT tags, this document is composed of standard HTML, and will therefore display correctly in any browser (but without the groupware functionality.)

```
<HTML>
<HEAD>
<TITLE>Build Front End</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<img src="d++.gif">
<HR>
<H1>Implement Front End</H1>
<P>
<HR>
The front end is composed of a scanner,
parser and semantic analyzer.
<P>
This step involves designing and
implementing these three components.
<P>
There are
<CLOCKVIEW VIEW=(
  TextColour red (NumText (jobTime "3")))
>???</CLOCKVIEW>
weeks allocated to this job step. (
  <CLOCKUPDT UPDT=(
    setJobTime "3" (jobTime "3"-1))
      >Decrease</CLOCKUPDT>/
  <CLOCKUPDT UPDT=(
    setJobTime "3" (jobTime "3"+1))
      >Increase</CLOCKUPDT>)
</BODY>
</HTML>
```

Figure 6: The HTML code displayed in figure 1.

The CLOCKVIEW tag is used in this document to embed a synchronously updated view in the HTML page. The general form of the tag is:

```
<CLOCKVIEW VIEW=view>alt</CLOCKVIEW>
```

which states that the given *view* is to be embedded in the document at this point. The *alt* text provides an alternate view (expressed in HTML) that will be displayed by browsers that are not aware of GroupScape tags. In the example of figure 6, the document contains the tag:

```
<CLOCKVIEW
  VIEW=(
    TextColour red (
      NumText (jobTime "3")))
>???</CLOCKVIEW>
```

which states that the current job time of step "3" is to be displayed as red numeric text. If the browser does not support GroupScape tags, the text "???" will be displayed.

The view embedded in the tag is a *constraint* that is implicitly attached to the model. The request 'jobTime "3"' is used to obtain the time currently allocated to job step "3". As was seen in figure 4, this request is implemented in the *JobTimes* ADT in the model. Therefore, whenever the time allocated to this step is modified by any participant, the MVC functionality of the GroupScape pattern will ensure that the embedded view is automatically updated. The CLOCKVIEW tag therefore allows synchronously updated views to be embedded in HTML documents, and ensures that they will be connected to the shared application data in the model, and automatically updated in response to any changes. It is important to note that these changes do not require the page to be reloaded, only the view to be updated locally by GroupScape.

The view constraints themselves are expressed in Clock's view language, allowing arbitrarily complex views involving text, graphics, sound and video.

The CLOCKUPDT tag allows updates to the model to be embedded within HTML documents. The general form of this tag is:

```
<CLOCKUPDT UPDT=updt>text</CLOCKUPDT>
```

The effect of the tag is to display the text *text*; whenever a participant clicks on the text, the message *updt* is sent to the model. In browsers that do not handle GroupScape tags, the text is displayed but is not sensitive to input. For example, the tag:

```
<CLOCKUPDT
  UPDT=(
    setJobTime "3" (jobTime "3"-1))
>Decrease</CLOCKUPDT>
```

implements the *Decrease* button of figure 1. Whenever a participant clicks on *Decrease*, the time allocated to job step "3" is decreased by 1 week: the *setJobTime* method is invoked to set the time of step "3" to 1 week less than the current time allocated to the step (as obtained through the method invocation *jobTime "3"*). The MVC functionality of the GroupScape pattern allows simple modifications in the model to implicitly update the application state, so the application windows and browsers of each participant are automatically informed of the change in the time for the job step, and can update their views accordingly.

Embedding constraint tags into HTML documents provides a flexible means of integrating the GroupScape browser with synchronous groupware applications. The CLOCKVIEW tag allows documents to contain synchronous views that are updated in real time without reloading the page; the CLOCKUPDT tag allows documents to contain hot areas that cause real time modification to applications. Since these tags embed communication with the application, no reprogramming of the browser itself is necessary.

5. ANALYSIS

The technique described in this paper simplifies the process of integrating group web browsing with synchronous groupware applications. By making it easier to connect a groupware web browser with applications, we hope it will be easier to experiment with different styles of collaboration based both on the information repositories of the web and on the immediate communication of synchronous groupware applications.

Through the example of figure 1, we have shown that there are four important ways that synchronous groupware should be integrated with the world-wide-web: through multiuser web browsing, through embedding synchronous views in web documents, through navigation controls in the application, and through application controls in the browser.

The techniques described in this paper support all four of these integration forms: the GroupScape browser is itself a synchronous groupware application, supporting browsing both privately and in a group. Through two new constraint tags (CLOCKVIEW and CLOCKUPDT), GroupScape allows synchronous views and application events to be embedded with HTML documents. Through an object-oriented design pattern, programmers can easily couple synchronous applications to GroupScape without programming.

The main advantage of this approach is the ease with which integrated WWW/groupware applications can be created. Once the design pattern has been developed once and placed in a library, programmers can rapidly develop new integrated applications. Once the application itself has been created, programmers can customize the integrated behaviour by experimenting with the definitions of the tags, without having to modify the application itself. The tags are designed so that people using a standard browser will still see a reasonable result, but will not receive the synchronous functionality. This allows pages designed for GroupScape to be used for standard browsing as well.

The primary disadvantage of our approach is that the design pattern and the new constraint tags require the custom-built GroupScape browser. To be successful in practice, an integration technique should be based on the features of the standard web browsers that are installed on millions of machines. We view our work as providing direction for how standard browsers could be extended to better support synchronous interaction.

5.1 Related Research

There has been substantial earlier research on the integration of synchronous groupware with the WWW. Many of these systems have been a strong influence on our own work.

Systems integrating synchronous groupware and the WWW can be roughly divided into two categories: those that provide multiuser web browsing and those that allow on-the-side synchronous groupware applications to be invoked through web pages. A major goal of the GroupScape project is to combine these two styles, facilitating tight integration of group web browsing and on-the-side applications.

5.1.1 Group Web Browsing

A number of tools aim to allow groups to browse the web together. The *GroupWeb* browser (Greenberg and Roseman, 1996) allows multiple participants to view the same web page simultaneously, slaving views so that as one participant changes pages, the views of the other participants are also updated in real time. To promote group awareness, telepointers, a multiuser scroll bar and a group annotation facility are provided.

The *Sociable Web* browser (Donath and Robertson, 1994) aims to make web browsing a more communal activity by providing dynamic information about who else is viewing the same page, and by providing chat communication tools that can be invoked directly from the current page. People browsing the web can therefore not only find out information about a given topic, but can also find out who else is looking for similar information at the same time.

Group web browsers provide excellent facilities for collaborating on finding and analyzing information. Group browsers (including GroupScape) have the disadvantage, however, of requiring a custom-built browser that supports its synchronous use.

5.1.2 Applications on the Side

Another approach to integrating synchronous groupware and the WWW is to use a standard browser, but to permit web pages to invoke synchronous applications that run on the side. Typically these approaches require participants to run a separate client application for the synchronous application, and provide only limited interaction between web browsing and the synchronous application.

One of the earliest applications in this style was *Yarn Web* (Rees and Woo, 1994), an electronic meeting system. Yarn Web combines a chat program for synchronous communication with web pages for session management and activity logging. Despite being based on a standard browser, Yarn Web allows a simple form of view slaving by having Yarn clients send page change notifications to the browser.

The *Como* system (Rex, 1996) allows synchronous groupware applications to be launched *within* web pages. Como applications are written as Java applets, and include chat, voting and shared drawing programs.

The *Mushroom* system (Kindberg et al.,1996) also takes this approach, providing the abstraction of the *MROOM*, a virtual room in which shared work can be carried out by one or more participants. MROOMS can be accessed by clicking on links in standard HTML documents,but are themselves synchronous applications. MROOMS are in effect places for synchronous interaction that can be found by traversing the web.

6. CONCLUSIONS

This paper has demonstrated a technique for integrating synchronous groupware applications with the WWW. The technique allows applications to be coupled with the new multiuser GroupScape browser. The connection is performed using a design pattern based on the model-view-controller architecture. To provide a wide range of integration styles, the GroupScape browser supports two new constraint tags, permitting application views to be embedded in HTML pages, and permitting pages to contain application controls.

7. ACKNOWLEDGEMENTS

GroupScape was developed by the author using the Clock groupware development toolkit. Clock and ClockWorks were developed by the author, Tore Urnes, Catherine Morton, Roy Nejabi and Gekun Song. This work was partially supported by NSERC and the ITRC.

8. REFERENCES

Beck, K. and Johnson, R. (1994) Patterns Generate Architectures, in *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'94)*, 139-149.

Donath, J.S. and Robertson, N. (1994) The Sociable Web, in *Proceedings of the Second International WWW Conference.*

Freemantle, J.R., Shepherd, P.R., Dunlop, J.D., Gray, L. and Miller, J.R. (1996) An Integrated Approach to Collection and Dissemination of Airborne Remote Sensing Imagery, in *Proceedings of the Second Annual Airborne Remote Sensing Conference and Exhibition*.

Finnigan, P.J., Holt, R., Kalas, I., Kerr, S., Kontogiannis, K., McDaniel, J., Müller, H.A., Mylopoulos, J., Perelgut, S., Stanley, M., Tourlakis, I., Tzerpos, V., Uhl, J., Wong, K. (1997) The Software Bookshelf. *IBM System Journal* (to appear).

Graham, T.C.N., Morton, C.A., Urnes, T. (1996a) ClockWorks: Visual Programming of Component-Based Software Architectures. *Journal of Visual Languages and Computing* 7, 175-196.

Graham, T.C.N., Urnes, T. and Nejabi, R. (1996b) Efficient Distributed Implementation of Semi-Replicated Synchronous Groupware, in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'96)*, ACM Press, 1-10.

Graham, T.C.N., Rasouli, R. and Urnes, T. (1997) *The Clock Language Reference.* http://www.cs.yorku.ca/~graham/reference.html.

Greenberg, S. and Roseman, M. (1996) GroupWeb: A WWW Browser as Real Time Groupware. In *Human Factors in Computing Systems, CHI Companion Proceedings*, ACM Press, 271-272.

Kindberg, T., Coulouris, G., Dollimore, J. and Heikkinen, J. (1996) Sharing objects over the Internet: the Mushroom approach. In *Proceedings of Global Internet '96*, IEEE Press, 67-71.

Krasner, G.E. and Pope, S.T. (1988) A Cookbook for Using the Model-View-Controller Interface Paradigm. *Journal of Object-Oriented Programming.*(1):3,26-49.

Rees, M.J. and Woo, T.K. (1994) A World-Wide-Web User Interface for an Electronic Meeting Tool. In *Proceedings of OZCHI*, 187-192.

Rex, M. (1996) Welcome to COMO Professional 1.0. http://www4.informatik.uni-erlangen.de/Projects/como/www/products/comopro1.0/www/doc/index.html