In Proceedings of the INTERACT97 Workshop on CSCW in HCI-worldwide, Sydney, Australia, July 15, 1997, pp. 23-28.

Component-based Groupware: Issues and Experiences

John Grundy

Department of Computer Science
University of Waikato
Private Bag 3105, Hamilton
NEW ZEALAND
jgrundy@cs.waikato.ac.nz

ABSTRACT There is a growing trend in both Software Engineering and HCI circles to developing "componentware" systems i.e. systems which are comprised of individual, self-contained systems rather than a single, monolithic application. The advantages of this approach are many, including higher degrees of reuse, more open architectures, end-users being able to choose the "best" components for their needs, and the development of more extensible systems. There may also be disadvantages, such as less than ideal user interface consistency, difficulty in agreeing on integration and inter-operation standards, and lack of high-level, robust componentware architectures. This paper discusses the impact componentware solutions may have on the development of new groupware systems, and gives some examples from the author's recent research.

KEYWORDS CSCW, groupware, componentware, work coordination, distributed systems

1. INTRODUCTION

Many different kinds of groupware systems and tools exist (Ellis et al, 1991). Examples include asynchronous tools, such as email, note annotations (Oinas-Kukkonen, 1996), and version control systems. Various systems provide synchronous collaborative work support, such as IRC (Pioch, 1993), GroupKit (Roseman and Greenberg, 1996a), Rendezvous (Hill et al, 1994), and BSCW (Bently et al, 1995). Many systems, such as Team Rooms (Roseman and Greenberg, 1996b), W4 (Gianoutsos and Grundy, 1996), and Lotus Notes (Lotus, 1993), combine synchronous and asynchronous modes of communication, usually by providing a variety of different groupware tools.

A great range of applications can make use of a variety of groupware tools, such as email, annotations, shared workspace editing, and discussion. Adding such capabilities to each environment that requires them in isolation results in a great deal of redundancy, and often limited or no reuse of tools.

Component-based systems offer a new approach to developing groupware applications, by building small, open and reusable tools which can be plugged together to form an environment. The end-user of an environment may even be able to choose the particular groupware tools they add to their environment, depending on their preferences or the tool capabilities they require. Some aspects of groupware systems are easy to make into components than others, and componentware solutions must be carefully designed to ensure good resulting environments.

2. COMPONENTWARE

Figure 1 illustrates the basic idea of a component-based system. Component-based systems (or "componentware") are built by combining a variety of

small (and sometimes larger) components, which provide a particular kind of functionality.

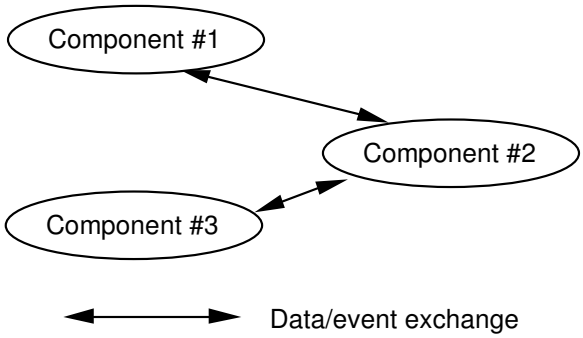

Figure 1. Basic Componentware System Architecture.

For example, for a system requiring multi-user support, components might include an email component, a note annotation component, an IRC-style chat component, a shared drawing editor component, a workflow component, and so on. Rather than continually reinvent the wheel when building systems that require such functionality, we can plug such pre-existing tools into a new system. The great advantage of componentware over e.g. library or framework reuse, is that new tools can be plugged into a componentware system, often while it is running, or old components may be unplugged and replaced.

Components in a componentware system exchange data and control in ways agreed upon by the designers of the system architecture. Components may have their own user interface style, or may agree on a common user interface approach. Some components may even be dedicated to providing an interface, while others deal with data storage/retrieval, inter-component communication, and so on.

Many componentware architectures have been developed in recent times, including OpenDoc (Apple Computer™ Inc, 1996), ActiveX (Microsoft™, 1996) and Java Beans (JavaSoft™, 1996).

Componentware solutions for groupware applications have also been proposed and several have been developed. Examples include TeamWave (Roseman and Greenberg, 1997) and the use of CORBA (ter Hoft et al, 1996).

The following sections outline the author's experiences with component-based groupware solutions, and why I believe they offer the best solution for reusable groupware applications.

3. EXAMPLE #1: SERENDIPITY

The Serendipity environment is a workflow/process modelling system which supports a range of CSCW capabilities (Grundy et al, 1996). These include email-style messaging, version control and configuration management, shareable modification histories, change description annotation, IRC-style chats, high-level group awareness, and synchronous and semi-synchronous editing of diagrams.

Figure 2 shows an example screen dump from Serendipity showing several of these facilities in use on a collaborative software development project. The highlighting of icons in the workflow model (top, right) shows a developer what parts of the process other developers are working on. The developer can add/read note annotations (bottom, left dialogue) and carry out a chat (bottom, right dialogue) with collaborators. The centre, textual view shows change descriptions (changes made) shown in a class header, made to the OOA diagram top, right. These changes have been annotated with information from the workflow model to assist developers in seeing both what changes have been made, but also why they were made.

Serendipity was developed by combining several different tools and environments. These included the a workflow modelling and enactment system, a generic annotation system, a generic text chat system, reuse of collaborative editing and version control abstractions, and integration with tools for performing work (e.g. software development and office automation tools).

A strict componentware solution was not used with Serendipity, although some of the tools used are stand-alone applications, and componentware-style event notification was utilised in many places. We needed to make some modifications to some of the environments to get them to work together, and to ensure consistent mechanisms for user interaction and data persistency.

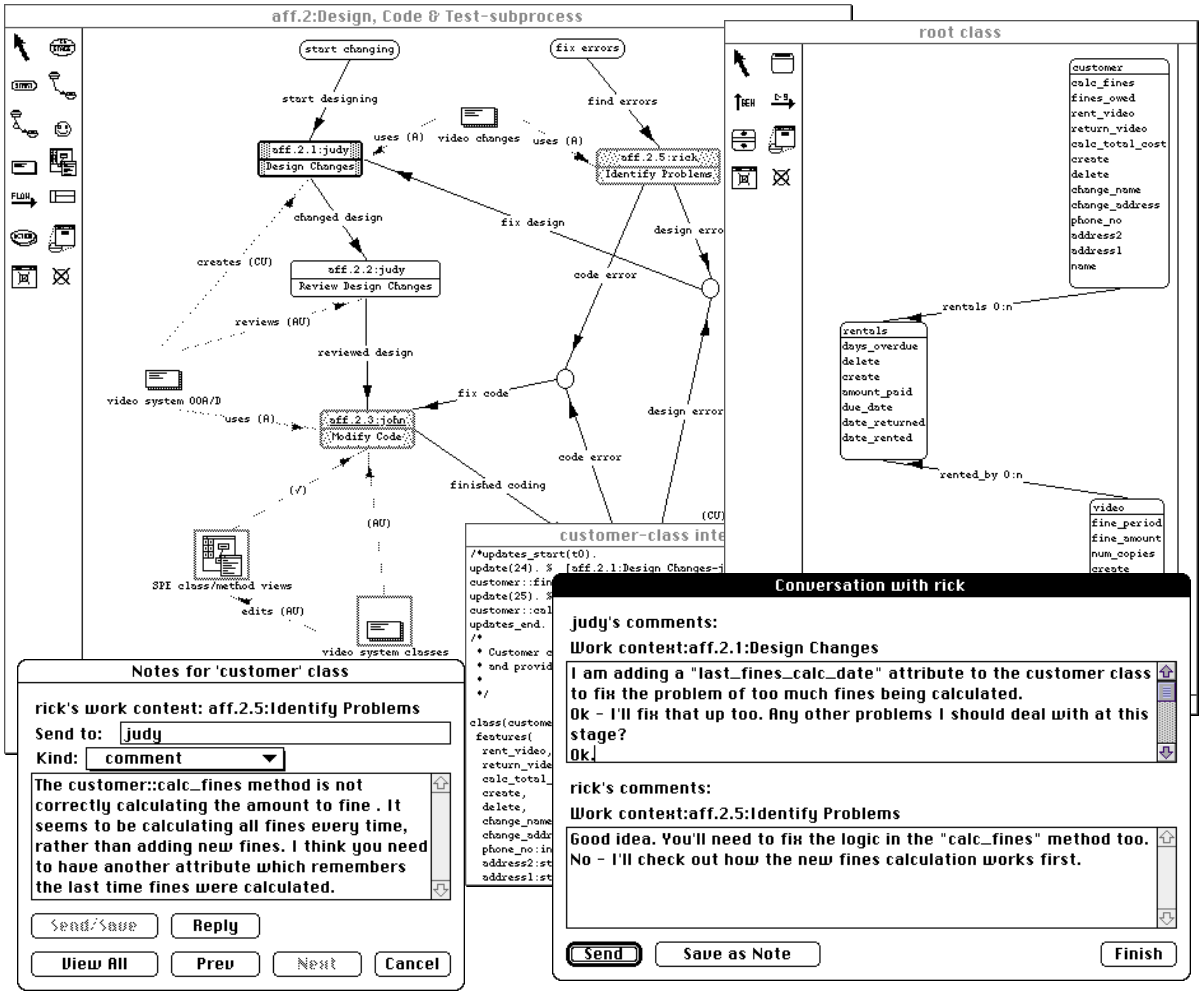

Figure 2. Various CSCW capabilities of the Serendipity process modelling system.

4. EXAMPLE #2: JCOMPOSER

JComposer is a CASE tool for the modelling and generation of environments using a componentware architecture, called Jviews (Grundy et al, 1997a). JViews provides abstractions for building multi-view, multi-user environments using components, and is a successor to MViews (Grundy and Hosking, 1996), the environment used to build Serendipity.

From our experiences with Serendipity and other environments built with MViews, we decided to move our work to Java and make use of the Java Beans componentware API in the construction of new tools and environments. This has several advantages over our previous approach with MViews, which was implemented in an OO Prolog:

- more portable and faster applications
- more open architecture for use of third party tools
- a proper componentware system, with stand-alone and interchangeable components, fostering better reuse of tools and abstractions

- access to better distributed systems capabilities for supporting multi-user applications

Figure 3 shows an example of a running environment (an ER modeller) built using JComposer. The bottom-left view is the users' view of an ER model, with the top-left view a visualisation of the components making up this view. The top-right view is an entity component which has been linked by the user to a filter (rectangle) and then an action (oval). These filter/action components provide reusable components for dynamic event handling. This model specifies that if the entity is renamed, the user should be notified by a message (using an email-like tool). The bottom-right view shows a visual query language we are developing for component structure querying. JComposer provides an environment for specifying the appearance of drawing editor icons and connectors, the structure of repository and view editors, and various reusable and extensible event-handling abstractions.

Third-party Java Beans components can be integrated into the environment and their data and events exchanged with those of JComposer components. A key feature of this work is that both environment developers and end-users can configure the structure of these systems, using the visual notations, providing powerful groupware environment composition capabilities (Grundy et al, 1997b).

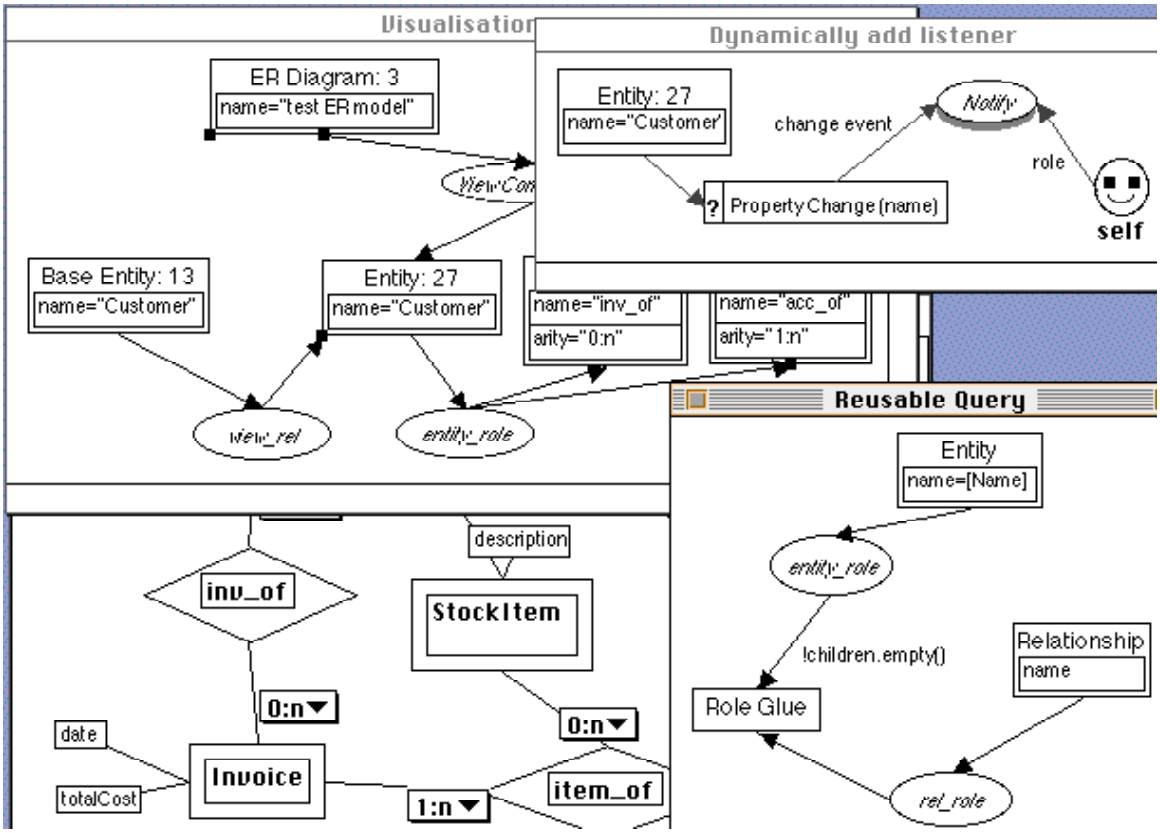

Figure 3. An example Jcomposer environment showing component composition and visualisation.

5. COLLABORATIVE INFORMATION

We are currently designing a new component-based groupware system for heterogeneous, collaborative information visualisation and wok coordination. The components in this architecture are user interface tools for the specification, visualisation and navigation of complex information spaces. Additional components allow the system to interact with WWW, Intranet and Corporate Database information sources. These tools interact with a standard Web browser and various desk top applications, such as word processors, database applications, email and chat systems, and so on.

This system will allow a wide variety of complex information sources to be collaboratively visualised and navigated in novel ways, and allow links between information items from different sources to be deduced or explicitly specified. A component-based architecture allows a variety of new and existing third-party tools to readily utilised, as well as our own tools. Groupware aspects of this system will include JComposer-style work coordination support, messaging and note annotation, collaborative browsing, and various group awareness facilities.

6. FUTURE TRENDS

Our experiences with component-based groupware development has indicated that many aspects of groupware systems can be effectively split into reusable, interoperable components. This leads to groupware systems which are much easier to build than by reusing frameworks or libraries providing low-level groupware capabilities. With the continued development of componentware solutions, including both internet and intranet-based component systems, the development of component-based groupware seems likely to increase.

We have found a major problem with component-based groupware can be in the computer human interface. If great care is not taken to ensure that components have a common look and feel, and common design style, component-based environments can become a mis-mash of poorly integrated tools. Care must also be taken to design component interoperation architectures so sufficient flexibility is provided to integrate new components into a system. We have found that end-users enjoy being able to compose their own component-based systems, but require tools which do not involve complex programming. Our visual languages are an attempt to provide more suitable human interfaces for composing complex component-based groupware.

Some groupware aspects are more amenable to being made into components than others. The ability to perform shared workspace editing needs to be carefully built into a system, as does provision for the possibility of various group awareness capabilities. We have found our event-based JViews architecture allows component-based awareness and synchronous editing to be supported by a component-based approach. Many existing systems, however, do not provide suitable capabilities to add these features onto an environment.

7. SUMMARY

Component-based groupware systems offer the possibilities of more open, extensible, reusable and, ultimately, more powerful systems than current technologies. Careful consideration must be given to designing the human interface and software architecture aspects of such systems, however, in order to make them feasible. We believe much scope exists for HCI research into these areas, and also into the large problem of end-user configuration of component-based software in general.

8. REFERENCES

Apple Computer Inc (1996) *OpenDoc Programmer's Guid".*

Bentley, R., Horstmann, T., Sikkel, K., and Trevor, J. (1995) Supporting collaborative information sharing with the World-Wide Web: The BSCW Shared Workspace system. In *Proceedings of the 4th International WWW Conference,* Boston, MA, December.

Ellis, C.A. and Gibbs, S.J. and Rein, G.L., Groupware: Some Issues and Experiences, *Communications of the ACM*, **34** (1), p. 38-58.

Gianoutsos, S. and Grundy, J. (1996) Collaborative work with the World Wide Web: Adding CSCW

support to a Web Browser, *Proceedings of Oz-CSCW96*, Brisbane, Australia, 30 August.

Grundy, J.C., Hosking, J.G., and Mugridge, W.B. (1996) Low-level and high-level CSCW in the Serendipity process modelling environment, in *Proceedings of OZCHI'96,* IEEE CS Press, Hamilton, New Zealand, 24-27 November.

Grundy, J.C. and Hosking, J.G. Constructing Integrated Software Development Environments with Mviews, *International Journal of Applied Software Technology*, **2** (3-4).

Grundy, J.C., Mugridge, W.B., Hosking, J.G. A Visual, Java-based Componentware Environment for Building Multi-view Editing Systems, *Proceedings of 2nd Component Users Conference*, Munich, July 14-17.

Grundy, J.C., Mugridge, W.B., Hosking, J.G. (1997) Support for End-User Specification of Work Coordination in Workflow Systems, Proceedings of the 2nd International Workshop on End User Development, Barcelona, Spain, 16-17 June.

.Hill, R. D. and Brinck, T. and Rohall, S. L. and Patterson, J. F. and Wilner, W. (1994) The Rendezvous Architecture and Language for Constructing Multi-User Applications*, ACM Transactions on Computer Human Interaction*, **1** (2), p. 81-125.

JavaSoft (1996) *JAVABEANS™ API*

Lotus Corporation (1993) *System Administration Manual*, Lotus Notes release 3

Microsoft (1996) *ActiveX*

Oinas-Kukkonen, H. (1996) Debate Browser – An Argumentation Tool for Meta Edit+ Environment, *Proceedings of the 7th Workshop on the Next Generation of CASE Tools*, Crete, June 20-21, 1996.

Pioch, N. (1993) *A short primer on IRC*, Ecole Polythechnique, Paris, Edition 1.1b, February.

Roseman, M. and Greenberg, S. (1996) Building Real Time Groupware with GroupKit, A Groupware Toolkit, *ACM Transactions on Computer Human Interaction*, **3** (1), p. 1-37.

Roseman, M. and Greenberg, S. (1996). TeamRooms: Network Places for Collaboration. *Proceedings of ACM CSCW'96 Conference on Computer Supported Cooperative Work*.

Roseman, M. and Greenberg, S. (1997) *Simplifying Component Development in an Integrated Groupware Environment*, Research Report 97-600-02, Department of Computer Science, University of Calgary.

ter Hofte, H., van der Lugt, H., Bakker, H., A CORBA Platform for Component Groupware, *Proceedings of the OZCHI96 Workshop on the Next Generation of CSCW Systems*, Hamilton, New Zealand, 25 November.