Building Multi-device, Component-based, Thin-client Groupware: Issues and Experiences

John Grundy, Xing Wang and John Hosking
Department of Computer Science, University of Auckland
Private Bag 92019, Auckland, New Zealand
john-g@cs.auckland.ac.nz

Abstract

The use of groupware, or collaborative work-supporting technologies, has become wide-spread, but many existing groupware systems are too difficult to integrate with domain-specific software applications, only work for specific user interface hardware, or provide inappropriate, thick-client architectural solutions. We describe a set of server-side software components we have developed providing a variety of thin-client groupware solutions (chat, email, annotation, to-do lists, notification etc). These components provide HTML and WML-based thin-client user interfaces and can be readily "plugged into" the server-side architectures of domain-specific applications. We focus on the key issues of designing and realising the user interfaces for such groupware solutions and report on our experiences to date.

Keywords: groupware, thin-client user interfaces, mobile user interfaces, software architecture

1 Introduction

People engage in group work in all organisations and during many activities. Group work ranges from face-to-face meetings where participants can interact freely, sometimes with the aid of computer technologies, to distributed location and/or time group work. Many "groupware" systems have been developed to aid in supporting group work. Examples include chat, email, ICQ, video and audio conferencing, collaborative document editors, shared calendars and to-do lists, and annotation and awareness facilities [Drummond et al 2001, Ellis 1998, Greenberg, 1991]. Most groupware to date has been run using dedicated desktop applications [Begole et al 1999, Chong and Sakauchi 2000, Roseman and Greenberg 1996]. This means much groupware has not been easily accessible when workers are travelling or without their normal desktop computer environments. Recently some specialised groupware systems have been developed to support mobile devices like PDAs and WAP phones [Kurashima et al 1999, Han et al 2000]. Much groupware has been "stand alone" or, when integrated with software packages like word processors and software tools, "hard coded" into these. This means it has been very challenging to engineer groupware and adequately integrate it with other applications users require.

Recent trends in organisations and software systems have included the move to web-based information "intra-nets", the growth of mobile computing technologies, and the use of "software components" to encapsulate, reuse and integrate functionality [Rylet 2001, Hartman and Dirksen 2001, Szyperski 1998]. Web-based systems typically utilise thin-client architectures and have key advantages of location and platform independence of the software, along with centralised software maintenance and data management. Mobile systems are deployed on cell phones, wearable computers, PDAs and laptops, allowing mobile work with networked, usually thin-client, interfaces via WAP/WML and similar technologies. Software components extend the object paradigm allowing software to be constructed from reusable, sometimes plug-and-play, parts.

We describe our work developing some proof-of-concept thin-client groupware for HTML (web browser) and WML (mobile) user interfaces. These support multiple input devices, in that users of the groupware can access the same facilities using different platforms and the groupware provides appropriate support for the device capabilities. They utilise a component-based architecture allowing both presentation and business logic server-side components to be integrated and reused, both among the groupware components themselves and (to date, a limited degree) with domain-specific server-side components.

2 Motivation

Figure 1 shows a scenario of several people having to work together to plan a travel itinerary. The work involves developing an agreed travel plan, which requires deciding locations to travel to, dates, times and transport options, and accommodation and activities. Groupware facilities are needed to provide communication, co-ordination and collaboration facilities [Grundy et al 1998]. Examples of communication facilities might include synchronous video/audio; semi-synchronous chat; and asynchronous email/messaging and document annotation. Co-ordination facilities might include notification events and group awareness clues (what other users doing; looking at; where they are etc).

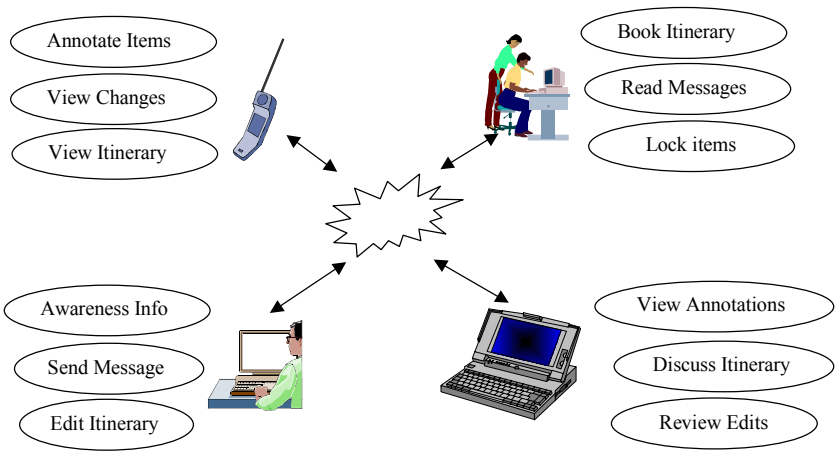

Figure 1. Distributed groupwork with various devices.

Collaboration facilities might include shared editing spaces (e.g. shared "whiteboards" and portals), version control over documents and document/information exchange techniques. Some users may access the system using a desktop-hosted browser; some may use mobile technologies (laptop, PDA, pager or phone).

Key requirements we have identified for groupware deployed in a situation like the above includes:

- *Support for thin-clients and thick-clients*. Thin-client systems (like HTML, WML) have most processing and user interface definition done server-side. This allows these systems to be deployed on any client running a suitable browser, allows developers to upgrade the server-resident systems more easily, and allows new people to join the environment without the need to download/install software. Thin-clients also have light-weight demands on clients, and if carefully designed can have low bandwidth demands.
- *Adaptability*. A wide variety of clients may present, requiring various forms of adaptation by the groupware servers. Different devices require different user interface designs and technologies (e.g. phones are very small; PDAs have limited or no colors). Different users have different capabilities (some can view and change some data; some cannot). Individual users may be doing a task that requires some capabilities and not others. Ideally our groupware should seamlessly adapt to these situations.
- *Compatibility and Consistency*. Groupware running on different client devices for different users should be compatible and consistent where possible. Similar interaction and layout approaches should be used where practicable, functionality should be similar and groupware should be "integrated" e.g. can exchange data/messages between all users whatever their locale/client device etc.
- *Architectures*. Where possible, the same server-side components should be reused for different kinds of groupware to reduce development and maintenance effort.
- *Integration*. Groupware is not typically used in isolation, but with domain-specific applications e.g. the travel planning software outlined above. Where possible, groupware should be accessible in a seamless way from such applications and should exchange events with such applications as appropriate.

Many examples of groupware have been developed. Some key examples include messaging systems (e.g. email, ICQ, IRC) [Drummond et al 2001], collaborative editing tools (e.g. Grove, DUPLEX, CocoDoC) [Ellis et al 1991, Pacull et al 1994, ter Hofte et al 1997], meeting support systems (e.g. MS Netmeeting™, TeamWave) [Roseman and Greenberg, 1996], and workflow and work co-ordination systems [Bandinelli et al 1996]. In general, many groupware systems are desktop applications and often make use of platform-specific hardware and software. These have the advantages of enabling support for high degrees of user interactivity and 3rd party application integration, at the cost of platform dependence and installation and maintenance costs. Much groupware is "custom built" from low-level software libraries and is very difficult to integrate fully with other applications.

Groupware development tools and toolkits enable the construction of new groupware facilities and tools much more easily, as they provide developers with high-level, groupware-oriented abstractions. Many examples exist, including Groupkit, JViews, COAST and Suite [Roseman and Greenberg 1996, Shuckman et al 1996, Dewan and Choudhary 1991]. Many of these systems, such as CocoDoc, TeamWave and COAST, utilise a "software component" model focusing on composing new systems from existing, tailorable parts, rather than complete system development. These component-based solutions potentially allow better reuse of groupware abstractions and integration of groupware facilities with other software applications. Most existing approaches to using software components to build groupware have to date focused on building thick-client, desktop applications or application components.

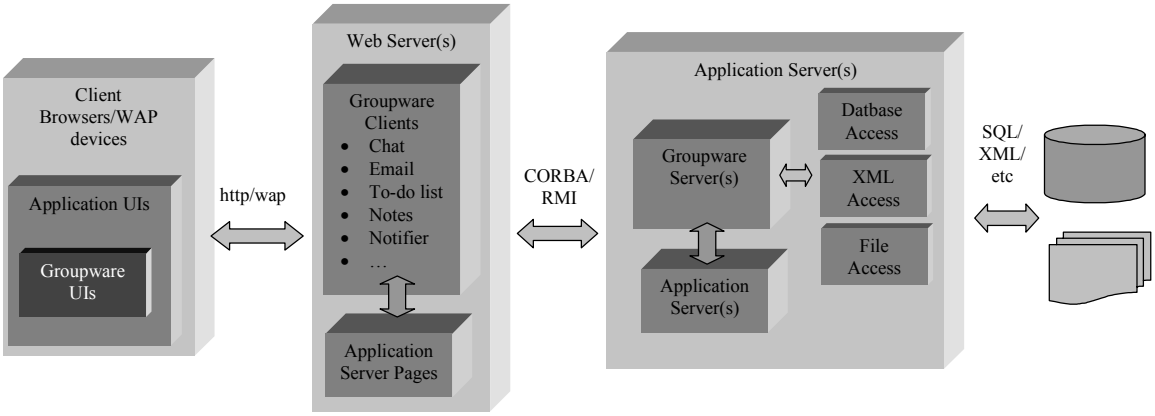

Figure 2. Architecture of our thin-client groupware.

The use of "new" interaction devices and technologies has become popular in groupware research and applications. Examples include the use of virtual reality interfaces (e.g. CHIME) [Dossick and Kaiser, 1999], web-based user interfaces (e.g. MILOS, OzWeb and BSCW) [Maurer et al 2000, Appelt, 1999, Kaiser et al 1998], large image displays [Humphereys and Hanrahan, 1999], and mobile devices [Hartman and Dirksen, 2001]. Some approaches, such as WebSplitter, combine multi-display device and groupware support techniques [Han et al 2000].

Many of these systems use custom architectures and implementation approaches, but a few have begun to use basic software component models, such as MILOS [Maurer et al 2000] and WebSplitter [Han et al 2000]. MILOS provides web-based project management software for (currently only) HTML web browsers. MILOS adopts a server-side component model to provide its teamwork support functionality. WebSplitter provides for multi-device and multi-user browsing of documents. It uses XML-encoded screen descriptions for multi-media groupware and uses this to produce mark-up for multiple display devices, including web browsers and PDAs.

Integrating such groupware with desktop applications is possible though limited, but integrating them with other Virtual Environments, web or mobile applications more promising. A number of challenges present, particularly with small-screen mobile device groupware and applications. User interface design for such groupware requires careful attention to detail and use of appropriate device characteristics. However, ideally all thin-client groupware should provide consistent interfaces for similar groupware functionality.

In the following sections we concentrate on describing the motivation for and design of the abstract architecture and user interfaces of our groupware components. We summarise some key aspects of our detailed design and implementation approaches and experiences to date with our thin-client groupware components.

3 Our Approach

We have developed a number of groupware components to provide thin-client groupware facilities for stand-alone use and use in conjunction with other thin-client (or thick-client) applications. These groupware components are summarised in Figure 2. They include:

- Groupware clients. These include user interfaces for text chat, email, note annotations, to-do list and so on. Being thin-client UIs, these are actually server-side presentation management services, accessed by web browsers and WAP browsers and provide HTML and WML marked-up data.
- Groupware servers. These provide centralised groupware messaging and data management facilities. As thin-clients must (typically) access a server, our groupware "clients" access these middle-tier groupware business logic and data management components, rather than doing peer-to-peer style communication (like ICQ, a thick-client application).
- A database access component, storing large numbers of small groupware component data e.g. chat messages, notification events, textual notes, to-do list items and so on.
- An XML access component, providing read/write storage of XML data for coarser-grained but structured groupware data e.g. email, chat and event histories.
- A file access component, providing read/write data management for unstructured groupware data e.g. large email content and attachments.
- Application client Uis, server pagers and application servers. These are the applications we have "plugged" our groupware components into. To date, we have focused on thin-client, HTML or WML applications (i.e. web-based or WAP-based) being augmented by our groupware components.

The client devices (web browser, WAP phone, PDA etc) communicate with the web server components using HTTP or WAP protocols. WAP device requests are directed to WML-supplying web server components and HTTP requests to HTTP-supplying components. The groupware user interfaces may be embedded inside (or linked to from) application interfaces, but can also be provided in their own browser window or WML cards.

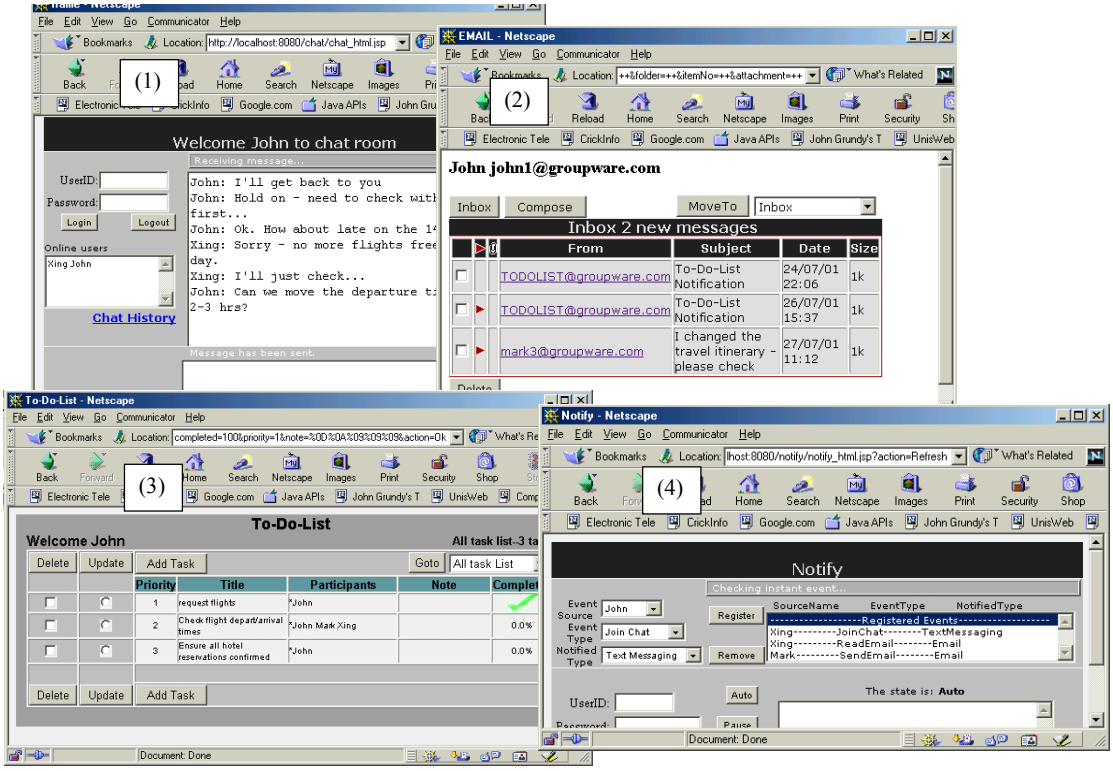

Figure 3. Examples of HTML thin-client groupware user interfaces.

The groupware web server components interact with the application's web server components by being included in application pages, linked to by application pages, and groupware interfaces providing links to application pages.

The web server components communicate with their groupware application server components via distributed object protocols: for example CORBA (if application server objects are CORBA objects) or RMI (if the application server objects are Enterprise JavaBeans components). The groupware servers interact with each other e.g. sending an email notifies the notification server; adding a note annotation to a chat history item notifies the chat server etc. An integrated application's application server components interact with groupware application server components by sending them selected notification messages (e.g. customer placed order; travel agent changed travel itinerary, etc). These typically provide generic subscribe-notify functionality our groupware server components can make use of to be sent application events. Our groupware servers can invoke (via additional plug-in components) specified application server functions. Data is stored/retrieved using a variety of mechanisms (SQL for databases, XML/XQL for XML-encoded data, binary for e.g. Word and Excel documents, GIFs, JPEGs etc).

4 Groupware User Interfaces

In this section we illustrate some of our groupware user interfaces and describe the key design rationale we used when developing these interfaces. Note that we currently implement the web server components using Java Server Pages and Java Servlets, and some of the user interfaces illustrated below have been developed to make use of things easy (or relatively easy) to do using these technologies, and could possibly be further improved with different web technologies or effort.

Consider the scenario of Xing, John and Mark planning a trip. Xing is the travel agent, Mark and John the customers. A travel planning application provides itinerary management, flight schedules and booking, accommodation details and reservation, etc. facilities. Our groupware components are used to provide various collaborative work support for the participants.

4.1. Web-based Groupware Interfaces for Desktop Computers

In Figure 3, John is using a desktop PC running a web browser to access the groupware functionality. He uses a text chat (1) component to discuss, semi-synchronously, with Xing and Mark his overall needs for a forthcoming trip, including a rough outline of locations, dates etc. Subsequently John may message Xing and/or Mark, or receive messages from them asynchronously (when the recipient may be off-line) using an email component (2). In this example, the group discuss revisions to the itinerary. Documents such as costing e.g. an Excel spreadsheet and detailed itinerary e.g. a Word document can also be attached to messages. The group can plan and review co-operative activities, along with being kept aware of others' work, using a to-do list facility (3). Here, John has sketched out tasks he and others perform. To-do list item histories convey historical progress on tasks allowing others to review work progress on shared tasks. Notification agents can be set up to watch for various events (two users begin a chat; a new email arrives; a note annotation is changed, or an application event occurs).

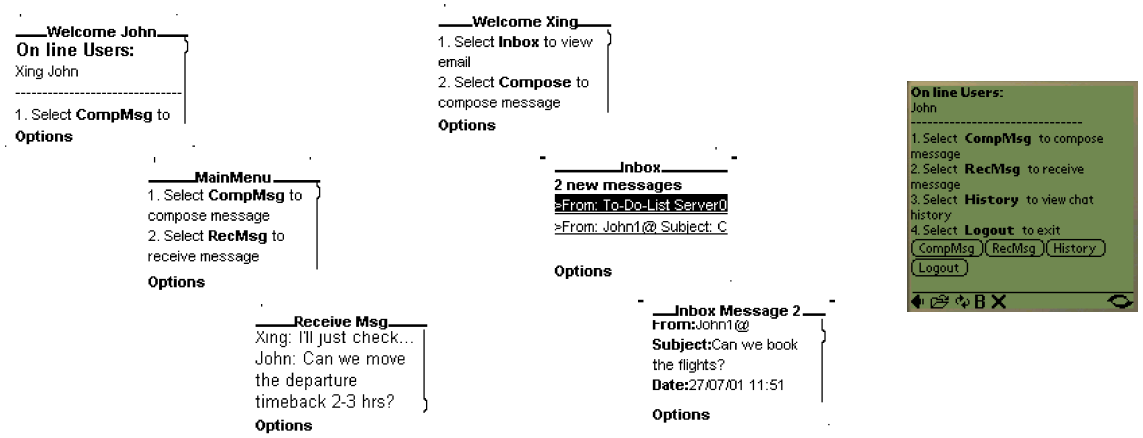

Figure 4. Examples of mobile phone and PDA groupware.

Here, John specifies he is interested in being informed of group discussions by Xing, so he can join in if able, and when Mark has read email from John, so John knows Mark is aware of changes to their shared travel plan (4).

When designing the user interfaces of such thin-client, HTML-based groupware, we focused on a common look-and-feel for most groupware orien\ted around message, to-do, note annotation and event history lists. A common, tailorable list formatting component is used to provide such lists. All of our web-based groupware is "pull-oriented" i.e. uses standard HTTP GET and POST commands to communicate with the server components. Chat, note and email clients periodically poll the server for new or updated information (or explicitly request updates on user-direction) in contrast to many thick-client groupware applications where the server pushes changes to clients. To obtain appropriate user interaction characteristics with the chat in particular, we separated the chat history from the chat message entry via HTML frames with the history frame periodically auto-refreshed (in addition to after the user posts a chat message).

The notify controller provides users with all possible events detectable for a specified component (groupware or third-party). For example, chat started/, joined; email sent, received, read. To-do list item added, updated. Note added, viewed, updated. For each event, the user indicates a notification action. For example, when Xing reads his email, John is told of this by a message. When Mark joins a chat, Xing is informed by a text message, etc. This allows co-operating people to be kept informed of basic groupware events others are performing. Our simple rule guard→action metaphor works well with simple notification tasks, but would need extension to handle more complex multi-event scenarios. In addition, we are extending our support for detecting application events e.g. Xing adds travel itinerary item, John changes hotel reservation, etc. and using our groupware to keep co-operating works informed of these events.

4.2. Thin-client Groupware Interfaces for Mobile Devices

In Figure 4 Xing is using a variety of thin-client, mobile phone-hosted groupware similar to that used by John through a conventional browser above. In the first three screens, Xing receives a chat message from John about a travel itinerary item change and replies. In the second three screens, Xing is reading an email from John telling him about further changes he has made. Users can also review to-do lists, add simple note annotations and set notification parameters. We attempted to maintain an identical set of functions for all WAP-based groupware as provided by the web-based groupware above, though accessed and presented in a way tailored to small mobile devices. While WAP devices can make use of a standard Push Application Protocol, allowing servers to push data to the WAP device, we decided to use auto- and manual-refresh for our groupware (chat, email, notes etc) as with the Web-based groupware. This was so we could both reuse the same server-side groupware components without needing to depend on client device characteristics, but also to maintain the same basic look-and-feel of user interaction across devices. In practice, the use of pull-only WAP interfaces has worked well, both from user interaction and groupware infrastructure perspectives.

A major challenge with small mobile devices is managing wide or deep interfaces. In web-based interfaces, the email, to-do list etc lists, item details and interactors can all often be displayed together. This isn't possible with most mobile device user interfaces. These either require an interface to be divided into a "stack of cards" (separate, inter-linked screens) or require right/down scrolling over a "canvas". We chose to provide a number of cards to divide up lists, details and operation selections. This requires more option selection by users with to-and-fro linking, but we discovered users found this far easier and more intuitive than tiresome tracking across a large area using the small window of the device display.

The right-hand side screen in Figure 4 shows Mark interacting with John and Xing via a mobile Palm PDA-hosted interfaces. Some of the Palm interfaces we developed use the same WML-based groupware components as the mobile phone interfaces illustrated above. Others are tailored to utilise the increased size available on such mobile PDA devices. In addition, some PDAs provide (limited) use of colour, which can be used to distinguish status of messages and notes, different users, authorship, and so on.

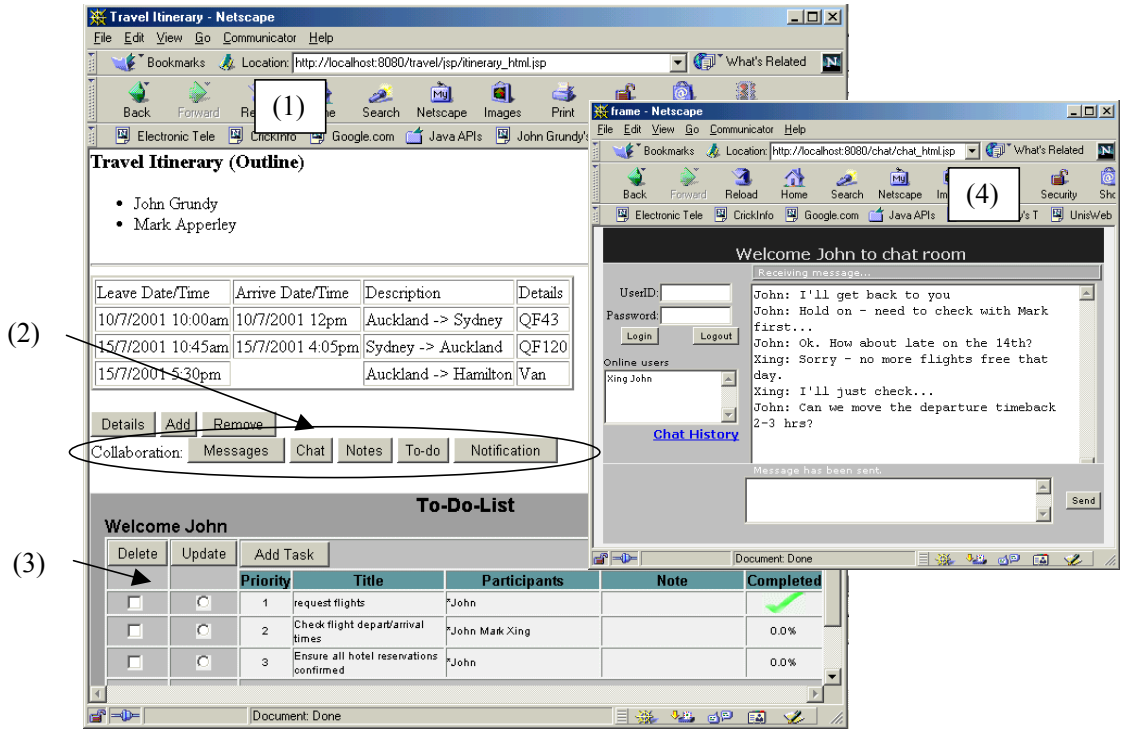

Figure 5. Examples of groupware component and application interface integration.

4.3. Integration with Other Thin-client Applications

Groupware is not typically used in isolation, but in conjunction with other application software e.g,. John, Xing and Mark in the example above want to collaborate to plan a travel itinerary using a thin-client travel planning application augmented with our groupware support. There are three main ways we have identified groupware components can be incorporated with other thin-client application user interfaces: via hypertext links; use of multiple frames or cards; and inclusion of groupware soucre into related interface pages.

- *Hypertext links, buttons or menus*. These provide an entry point to accessing groupware facilities. The groupware interfaces are displayed in another frame, window or in-place in the application user interface after selecting the affordance.
- *Multiple frames or cards.* These place the groupware user interfaces in a different place within the same thin client display. Frames are effective for desktop-hosted browsers; cards for small-screen mobile devices. They have the advantage over display in another window of allowing the user to see the groupware (frames) or skip to it easily (cards).
- *Including groupware pages*. This technique places the groupware thin-client source (HTML or WML) within part of the application user interface. The advantage is of more seamless integration than frames or cards, but can adversely affect the application interface layout and perception by the user (especially for small-screen devices).

We have focused on the first two approaches and begun investigating the third. Figure 5 shows some examples. In the application's travel itinerary outline screen (1), a set of groupware buttons have been included into the source JSP for this page (2). The actual list of buttons displayed can change if a user enables/disables particular groupware facilities (e.g. the Messages button disappears if the user specifies in our groupware preferences they don't want to use our email facility). Multiple frames have been used to include both the application-specific travel itinerary screen and the reusable groupware to-do list viewer (3). Clicking on the Chat button will open another window for the chat viewer (4).

4.4. Groupware Interface Design Comparison

Key design criteria for our groupware has been to 1) provide the user with an effective, efficient user interface to interact with; 2) realise the same groupware functionality and data across different devices; 3) where possible, preserve similar interaction and information display approaches across different groupware and across different display devices; and 4) preserving groupware and application interaction and display characteristics when composing groupware and application user interfaces.

Figure 6 (a) and (b) illustrate the dividing of the Email user interfaces for HTML and WML devices. In the HTML version intended for desktop PC browsers, the user logs in, selects an email message from a list (which may span multiple pages) and reads email details or composes an email via another page. In the WML version, we have preserved the same logical interface divisions but added additional cards and pages to enable users to be "led through" instructions, email headers and email text.

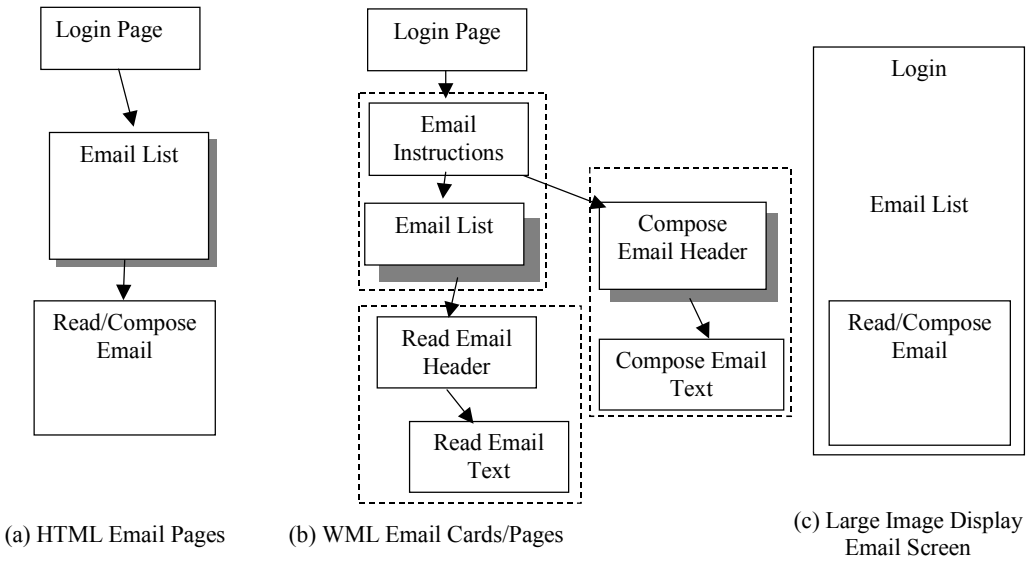

Figure 6. Splitting user interfaces for different device characteristics.

For example, a list of options is presented in the HTML version as buttons on the email list page(s), but on a separate card linked to other cards in the WML version. A set of WML cards enables the user to specify To: and Cc: recipients, message subject and importance and message body, whereas a single page is used in the HTML version. A Large Image Display approach might use a single page, with user log-in at the top, a scrollable list of messages, and reading/composition area at the bottom of the screen.

A problem with the approach we used to realise these different groupware interfaces is that multiple implementations must be provided for different display devices (e.g. Large Image device vs desktop browser vs PDA browser vs mobile phone vs pager).

To some extent the division of user interfaces, choice of interaction options/buttons/menus etc, and the basic data display approaches can be automatically selected for a user's device. We are developing an adaptation mechanism where interface designers specify logical page labels, edit fields, image options, option selection affordances (buttons, menus), page element display priorities, and basic logical groupings of page elements (lists and tables). A single logical interface specification is provided and at run-time an HTML or WML version is created by a web server plug-in component using device characteristics (width, height, colour support, default font rendering width/height, user preferences etc). Parts of the interface that extend beyond specified width/height area for the device are folded into additional cards/pages which are linked from the main page for the interface given to the client device. In addition, user and task adaptation is provided by turning interface components on/off, or hiding them altogether, if inappropriate for the user or the particular user task being performed. We are currently reimplementing our groupware and several thin-client applications using this approach to provide adaptable thin-client interfaces.

5 Groupware Components

We briefly discuss the design of our groupware components in this section. Figure 7 illustrates some of the components that comprise the chat facility. The dashed boxes indicate different hosts these are normally run on. Each of our groupware components comprises one (or more) server-side dynamic web services (we use Java Server Pages). Some of these web services provide clients with HTML, some WML and some WML for different devices e.g. PDA vs mobile phone vs pager. The web services each have a number of associated components (we use JavaBeans) that encapsulate data representation, management and processing. Many of these can be reused by different web services e.g. user and session management, event lists, and message and attachment management. Web services run on the same host as the web server that the client browsers connect to (wireless one via a wireless gateway). The chat service and application interface e.g. travel itinerary viewer, can interact by hypertext links, multiple HTML frames enclosing the two interfaces, or one interface including the other.

The presentation-tier components connect to a set of remote objects that provide application server-tier functionality. Many of these are reusable e.g. user authentication and look-up, event history management, event notification management and so on. All groupware server objects provide subscribe-notify support so the notification facility can subscribe to various events and action these. A number of components provide data mangement: database access, XML file manipulation and binary file manipulation. The application servers managing the remote objects may be deployed on different hosts for increased performance and reliability support. These servers could be implemented in different languages, depending on the implementation technology used (e.g. DCOM and CORBA would support this).

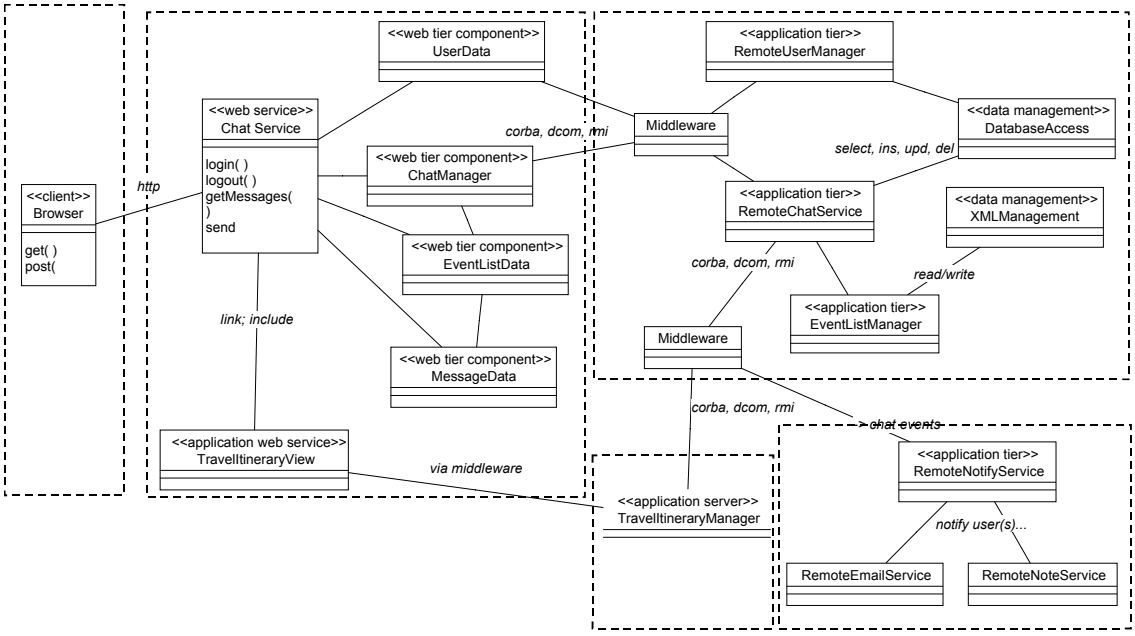

Figure 7. Example of component design for our thin-client groupware.

6 Implementation and Experience

To date we have used Java Server Pages (JSPs) to implement our web services, JavaBeans to implement the web service components, CORBA to implement our remote application server objects, and JDBC and XML to implement data management. JSPs are used to provide server-side presentation logic, receiving user requests and interacting with JavaBeans to fulfil them, or formatting JavaBean-held data for output to users. Some JSPs provide HTML-based input/output while others provide WML (we are working on integrating these into single, adaptive components). JavaBeans encapsulate data and interaction with remote CORBA objects, which provide centralised groupware server functionality. We used CORBA to provide basic remote object functionality for simplicity and to allow implementation of reusable servers independent of client-side implementation language, platform etc. XML is used to encode and store large amounts of hierarchical groupware data e.g. event, email, chat and to-do list histories.

JDBC database connectivity is used to manage user, message and notified configuration data. Our CORBA servers all provide event subscribe-notify to enable notification mechanisms. Third-party client-side JSPs (or other server-side scripting technologies) may include some of our JSPs to provide in-place groupware, or may provide links to them. To-do list items may provide hypertext links to appropriate 3rd party thin-client application pages (stored in the database associated with each to-do list item). Events from 3rd party application server components can be subscribed to and translated into our groupware CORBA object events by "notification wrappers" written using the target application technology.

We have built a range of groupware interfaces using thin-client interfaces, component-based development techniques and that provide groupware facilities with limited adaptability (to different devices) and integration (both among the groupware components and with 3rd party application components). Across these groupware user interfaces we have attempted to provide consistent display and user interaction look-and-feel characteristics. We have performance-tested our groupware to demonstrate the server-side CORBA servers provide efficient management of groupware events and data for a large number of concurrent users. We have carried out some basic usability testing of our groupware using function checklist and common design guidelines, comparing our interfaces to both those of third-party thin-client groupware we have encountered and to facilities provided by thick-client applications. In general our groupware components provide comparable facilities and interaction approaches across multiple devices to custom-built thin-client groupware applications.

We are currently designing a combination of observational and questionnaire-based usability experiments to more precisely gauge the effectives and efficiency of our groupware components. We are integrating each of our set of groupware components providing interfaces for different devices into a single chat, email, note, to-do list, notifier etc JSP which detects the device characteristics and provide appropriate interface (HTML, WML for Phone, WML for PDA, etc) for the device. We are also adding further adaptation support to tailor interfaces to different users (e.g. user display and interaction preferences; facilities based on user e.g. moderator can update/delete messages etc) and possibly tasks (some facilities irrelevant for a particular user task are disabled). We are extending groupware client components to enable further tailoring e.g. colours, fonts, display layouts, images and so on, using component property setting that can be done by developers and sometimes end users.

7 Summary

We have designed and prototyped a range of thin-client groupware using a component-based approach. Our groupware components reuse significant numbers of server-tier abstractions (messages, events, event and message histories, subscribe-notify infrastructure, server organisation) and presentation-tier abstractions (list management, component configuration, message, annotation and event representation). Our groupware provides interfaces for both HTML-based and WML-based client devices. We have provided some basic mechanisms for groupware and 3rd party thin-client application integration. To date our groupware components have proved useful for supporting basic communication and co-ordination needs in thin-client domains. Our groupware provides page-based interaction between users including semi-synchronous chat and task awareness facilities and asynchronous email, notes, to-do lists and so on. Due to its thin-client architecture, fully synchronous exchanges like key-stoke and mouse-movements are not supported. We are working on further support for collaboration (version management and some additional group awareness support, including last-page-accessed information), run-time adaptive components (to devices, users and tasks) and possibly Enterprise JavaBeans-based sever-side components (to further improve reusability, performance and 3rd party application server integration support).

8 Acknowledgments

Support for this research from a New Economy Research Fund grant and the University of Auckland Research Committee is gratefully acknowledged.

9 References

APPELT W. (1999): WWW based collaboration with the BSCW system, *Proc. 26th Conference on Current Trends in Theory and Practice of Informatics*, Lecture Notes in Computer Science 1725, 66-78, Springer-Verlag.

BANDINELLI, S., DINITTO, E., AND FUGGETTA, A. (1996): Supporting cooperation in the SPADE-1 environment, *IEEE Transactions on Software Engineering* **22**(12), 1996.

BEGOLE, J., ROSSON, M.B., SHAFFER, C.A. (1999): Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. *ACM Transactions on Computer-Human Interaction* **6**(2), 95-132.

CHONG, N.S.T., SAKAUCHI, M. (2000): e-CoBrowse: co-navigating the Web with chat-pointers and add-ins - problems and promises, *Parallel and Distributed Computing and Systems* **2**, 803-808, IASTED/ACTA Press.

DEWAN, P. AND CHOUDHARY, R. (1991): Flexible user interface coupling in collaborative systems, *Proc. of ACM Conference on Human Factors in Computing CHI'91*, 41-49, ACM Press.

DOSSICK, S.E. AND KAISER, G.E. (1999); CHIME: A Metadata-Based Distributed Software Development Environment, *Proc. Joint Seventh European Software Engineering Conference and Seventh ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 464-475, ACM Press.

DRUMMOND, S., BOLDYREFF, C., RAMAGE, M. (2001) Evaluating groupware support for software engineering students. *Computer Science Education* **11**(1), 33-54, Swets & Zeitlinger, Netherlands.

ELLIS, C.A. (1998): A framework and mathematical model for collaboration technology, *Coordination Technology for Collaborative Applications: Organization, Processes, and Agents*, 121-144, Springer-Verlag.

ELLIS, C.A., GIBBS, S.J. AND REIN, G. (1991): Groupware: some issues and experiences, *Communications of the ACM* **34**(1), 39-58.

GREENBERG, S. (1991): Computer-supported cooperative work and groupware: an introduction to the special issues, *International Journal of Man-Machine Studies* **34**(2), 133-141.

GRUNDY, J.C., MUGRIDGE, W.B., HOSKING, J.G. AND APPERLEY, M.D. (1998): Tool integration, collaborative work and user interaction issues in component-based software architectures, *Proc. TOOLS Pacific '98,* Melbourne, Australia, 24-26 November 1998, IEEE CS Press.

HAN, R., FERRET, V., NAGHSHINEH, M. (2000): WebSplitter: a unified XML framework for multi-device collaborative Web browsing. *Proc. ACM 2000 Conference on Computer Supported Cooperative Work*, 221-230, ACM Press.

HARTMANN S, DIRKSEN V. (2001): Optimization of internal business processes through integration of mobile commerce components. *Information Management & Consulting* **16**(2), pp.16-19, IMC GmbH, Germany.

HUMPHREYS, G., HANRAHAN, P. (1999): A. distributed graphics system far large tiled displays. In Proceedings ofVisualization '99, IEEE CS Press, 1999, pp.215-527.

KAISER, G.E., DOSSICK, S.E., JIANG, W., YANG, J.J. AND YE, S.X. (1998): WWW-based Collaboration Environments with Distributed Tool Services. *World Wide Web* **1**, Baltzer Science Publishers.

KURASHIMA, A., MAENO, K., ICHIMURA, S., TAGASHIRA, S., TAKETSUGU, M., NAGATA, Y. (1999): A mobile groupware system "Nakayoshi" supporting local area collaboration. *Trans. Information Processing Society of Japan* **40**(5), 2487-2496.

MAURER F, DELLEN B, BENDECK F, GOLDMANN S, HOLZ H, KOTTING B, SCHAAF M. (2000): Merging project planning and Web enabled dynamic workflow technologies. *IEEE Internet Computing* **4**(3), 65-74, IEEE CS Press.

PACULL, F., SANDOZ, A., SCHIPER, A. (1994): DUPLEX: a distributed collaborative editing

environment in large scale. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pp.165-173.

ROSEMAN, M. AND GREENBERG, S. (1996): Building real-time groupware with GroupKit, a groupware toolkit, *ACM Transactions on Computer-Human Interaction* **3** (1), 66-106, ACM Press.

ROSEMAN, M. AND GREENBERG, S. (1996): TeamRooms: network places for collaboration, *Proc. ACM 1996 conference on Computer supported cooperative work*, 325 – 333, ACM Press.

RYLEY, S. (2001): Corporate portal development: a practical approach ensures real business benefits. *Business Information Review* 18(2), 28-34.

SHUCKMAN, C., KIRCHNER, L., SCHUMMER, J. AND HAAKE, J.M. (1996): Designing object-oriented synchronous groupware with COAST, *Proc. ACM Conference on Computer Supported Cooperative Work*, 21-29, ACM Press.

SZYPERSKI, C.A. (1997): *Component Software: Beyond OO Programming*, Addison-Wesley.

TER HOFTE, G.H. AND VAN DER LUGT, H.J. (1998): CoCoDoc: A framework for collaborative compound document editing based on OpenDoc and CORBA. *Proc. IFIP/IEEE international conference on open distributed processing and distributed platforms*, Toronto, Canada, May 26-30, 1997, 15-33, Chapman & Hall.