SWIM: An Alternative Interface for MSN Messenger

Minh Hong Tran, Yun Yang
Smart Internet Technology CRC, and
CITR—Swinburne University of Technology
PO Box 218, Hawthorn, 3122, Australia
{mtran, yyang}@ict.swin.edu.au

Gitesh K. Raikundalia
ITArl—Victoria University
PO Box 14428, Melbourne City, 8001, Australia
Gitesh.Raikundalia@vu.edu.au

Abstract

The research of the authors investigates an alternative interface for Instant Messaging (IM). This paper presents SWIM (SWinburne Instant Messaging), an IM tool that is developed based on MSN Messenger. SWIM presents an innovative interface design that combines the conventional sequential interface with the adaptive threaded interface. In addition, SWIM supports persistent conversation which facilitates users' participation to a group conversation. The integrated interface and support for persistent conversation allow SWIM to be used both as a convenient tool for social conversation and as an effective tool for task-oriented group discussion. In this paper, we discuss the design approach of SWIM, describe details of our implementation technique, and report a preliminary evaluation of SWIM. The evaluation shows that SWIM holds great promise in supporting group conversation.

Keywords: Sequential interface, adaptive threaded interface, persistent conversation, Instant Messaging, chat.

1 Introduction

Instant Messaging (IM) is a text-based computer-mediated communication tool that allows an instant exchange of messages between people. IM has increasingly become a popular communication means (Isaacs et al., 2002; Muller et al., 2003; Nardi et al., 2000; O'Neill and Martin, 2003). In the early days, IM was used primarily to foster social interaction. Today, the technology has become increasingly popular in business. In particular, IM has been adopted by commercial organisations as a productivity-enhancement tool.

In comparison to the early days of IM, the technology has been used for a wider range of purposes, raging from sole social interaction amongst teenage groups to more task-oriented discussion in the workplace. However, the interface design of IM has not changed much to accommodate conversational styles of different tasks. The sequential interface of IM is limited in supporting coherence of conversation (Cech and Condon, 2004). As a result, stepping outside the zone of social conversation, IM becomes less effective and more difficult to use as a tool for group discussion.

Several alternative interfaces have been investigated to enhance the conventional sequential interface of IM. Some representative examples include Babble (Erickson et al., 1999), Chat Circles (Viegas and Donath, 1999), Coterie (Spiegel, 2001), Threaded Chat (Smith et al., 2000), and so on. Babble presents a list of topics and then displays messages of each topic in the sequential layout. Babble also provides "social proxy" to show the participation of users across topics. Chat Circles represents users as coloured circles. A circle expands when a new message is posted. The interface of Chat Circles aims to provide more information about users' levels of turns in a conversation. Coterie is another alternative interface that visualises users' presence, conversational activity and the structure of a conversation.

This paper presents our approach to designing an alterative interface for IM. The focus of our approach involves utilising the strengths of existing interface models and integrating them to support group discussion. In particular, our approach combines the sequential interface with the adaptive threaded interface. We apply this approach to designing SWIM (**SW**inburne **I**nstant **M**essenger). SWIM is an IM tool that is developed using the MSN Messenger communication protocol. The aim of SWIM is to retain support for social conversation provided by MSN Messenger, and to improve support for group conversation.

This paper is structured as follows. Section 2 reviews alterative design approaches to the conventional sequential interface of IM. Section 3 then describes our unique approach of integrating the sequential interface with the adaptive threaded interface. In particular, we present the design, implementation and preliminary evaluation of SWIM. Finally, Section 4 discusses potential enhancements to SWIM and lessons learnt from the design and implementation of SWIM.

2 Alternative Interfaces

There is a diverse range of interfaces for IM. This section examines three major design models, including the sequential interface, threaded interface, and graphical interface.

2.1 Sequential Interface

The sequential interface has been the most dominant interface model, adopted by almost all popular IM systems, such as MSN Messenger, Yahoo Messenger and AOL Messenger. The sequential interface presents messages in a list, which is often sorted in a chronological order with the latest message at the bottom. This interface model is simple, easy to use, and suitable for the fast pace of online social conversations. However, the design is limited in many ways such as lack of mapping between people and their messages, no listening in progress, poor turn-taking support, obscure visual cues that facilitate communication, and so on (Garcia and Jacobs, 1999; Smith et al., 2000; Vronay et al., 1999).

Currently, IM is used mostly in one-to-one conversations (Tran et al. 2005), and used only occasionally in group conferences where it appears to be inefficient for three main reasons.

First, IM displays messages of a conversation in a sequential order, which is highly limited in creating a structured and logical layout of group discussion. Thus, it makes it difficult to connect coherently between questions and answers, and between two consecutive messages of the same person. This issue becomes more problematic when the number of users participating in a discussion increases.

Second, given the nature of a linear sequence of messages, it is impossible for the users to respond explicitly to a particular message that was previously posted by another person. Hence, users often copy-and-paste the contents of the previous message into the message they are currently composing.

Third, the conversational context of group discussion is missing in the sequential structure. Conventional IM systems only provide a limited set of visual cues for who is typing. In addition, they fail to indicate the point in the conversation into which a new message fits.

2.2 Threaded Interface

To tackle the limitations of the sequential interface, the threaded interface has been studied. For example, Threaded Chat (Smith et al., 2000) organises chat messages into threads, and presents the threads in a tree-based layout. The threaded interface is reported as useful in helping users sustain their discussion, supporting turn-taking and producing a more balanced level of participation in a group.

However, the evaluation of Threaded Chat shows that the participants find the system difficult to use, especially in navigating between threads. Additionally, two major usability concerns challenge the threaded interface-based design.

First, the focal point of a conversation is not well supported by the threaded interface. It is difficult to show a node in the message tree at which a new message is located. Unlike the conventional sequential interface where a new message is always pushed to the bottom of the message stack, the threaded interface allows new messages to be displayed anywhere in the message tree. As a result, it becomes difficult to keep track of what new messages there are, who posted the latest messages, etc. The problem of missing the focal point forces the users to scroll up and down frequently in order to track the arrival of new messages. As the size of the message tree (i.e., the number of branches and levels) grows, this becomes conspicuously problematic.

Second, in IM systems that implement the sequential interface, users are not required to think about where they should place their messages, as new messages automatically appear at the bottom of the message stack by default. However, when messages are displayed in the threaded interface, users might carry an extra cognitive load of justifying where to post their messages. This can be problematic when topics of a conversation are not clearly defined.

2.3 Graphical Interface

In recent years, as personal computers have become more powerful and the network bandwidth has increased, it has become feasible to include graphical content into IM. Several studies have examined the graphical interface as an alternative to a conventional text-based model, for example The Palace (The Palace, 2005), Comic Chat (Kurlander et al., 1996), Chat Circles (Viegas and Donath, 1999) and Coterie (Spiegel, 2001). The appealing look-and-feel of the graphical interface is able to enrich online social conversations. For instance, a person is now represented by an avatar instead of merely a text-based username; background images can be included to reflect the social culture of conversational partners.

The Palace is an icon-based tool that provides a unique way of allowing users to portray their social identity and personality. In The Palace, users are represented by cartoon characters. Users can tailor their favourite avatars which are meaningful to them as well as conveying their personalities to others. The Palace is a fun tool for social conversation, but provides little support for maintaining the structure of group conversation. As a result, it is not practical for group discussion.

Comic Chat is another attractive graphical chat tool that creates a stylish visualisation of a conversation. Comic Chat adopts a comic book metaphor to lay out a chat conversation in the form of a story, which is very amusing. Unfortunately, due to space constraints in Comic Chat, only a small portion of a conversation is shown. Moreover, comic avatars and background images used in Comic Chat are loosely related to the conversation itself, which could be misleading.

Chat Circles represents users as coloured circles. A circle expands when a new message is posted, and becomes blurry after a period of idleness. Chat Circles is able to convey the level of turns in a conversation. In Chat Circles, messages are conversant-oriented in structure, thus there is no support for grouping related messages.

Coterie is a graphical interface of an IRC conversation that focuses on visualising users' presence, conversational activity and the structure of a

conversation. In Coterie, users are represented as coloured ovals which bounce and become brighter when they post messages. Coterie defines heuristics to analyse the relationships between messages, and then groups them into threads. However, messages in each thread still appear in a linear order; therefore Coterie has limitations like the conventional sequential interface, as discussed above.

3 Our Design Approach

Our design approach is an innovative integration of the sequential interface and the threaded interface. Although the two individual models of interface are already known, the way that we approach and integrate them is new.

Each of these two interface models offers unique strengths in supporting online conversation, and it is an intention of our approach to utilise their strengths. We realise that the sequential interface is simple and highly effective in presenting the flow of a conversation, whilst the threaded interface is useful in presenting the structure and coherence of a conversation.

Furthermore, our approach also aims to investigate the *adaptive* threaded interface. Unlike the threaded interface implemented in previous tools, such as Threaded Chat, the adaptive threaded interface allows the appearance of message threads to be adjusted automatically to suit users' conversational behaviour and the context of a conversation.

In addition to designing a new effective interface, this project investigates support for persistent conversation in IM. One of the SWIM design objectives is to allow users, who join a group conversation late, to be able to view the entire content of a conversation. Users are also able to save and retrieve a conversation for later use.

Prior to this project, we have applied our design approach to the design of Relaxed Instant Messenger (RIM) (Tran et al. 2005). The design and evaluation of RIM show that the approach is very useful in improving the process and quality of group conversation. Inspired by the successful application of RIM, the present research aims to apply our design approach to an existing commercial IM client.

This paper presents SWIM, an IM client that is developed based on MSN Messenger, one of the most (if not the most) popular commercial IM clients. In addition, MSN Messenger provides several programming options that are useful for the development of SWIM (as discussed in Section 3.3). Three major design aims of SWIM include:

(1) utilising the simplicity of the sequential interface, and the structural usefulness of the threaded interface;

(2) integrating and tailoring the two interface models in order to support task-oriented group conversation; and

(3) supporting persistent conversation.

This section first describes the user interface of SWIM. It then presents SWIM support for persistent conversation. Next, it discusses our implementation strategy, and reports an early preliminary evaluation of SWIM.

3.1 User Interface of SWIM

Figure 1 shows the interface of SWIM, which includes four main panels: *User List*, *Conversation View*, *Tree View* and *Message Entry*.

(A) User List

User List shows presence information about a local user and remote users, including details of users' names, avatars, and typing activity.

In comparison to MSN Messenger and other existing IM clients, SWIM support for users' names and avatars is similar, but SWIM enhances support for conveying awareness information about users' typing activity significantly. SWIM shows *multiple* "who is typing" cues at the same time, in contrast to current IM tools in which a visual "who is typing" cue only shows one user typing at the time. The design aim of providing multiple visual "who is typing" cues is to enhance support for turn-taking in a conversation. In Figure 1(A), two users—Minh and ConDock1—are currently composing new messages.

(B) Conversation View

Conversation View is a panel that adopts the sequential interface to show messages of a conversation. In this panel, messages are displayed in chronological order, with the latest message appearing at the bottom of the panel. Additionally, Conversation View uses coloured icons to enhance coherence of a conversation and to support a smooth shift between Conversation View and Tree View.

A unique colour is assigned to messages of each thread, and SWIM shows a coloured icon in front of each message. Using coloured icons allows conversants to quickly recognise a thread to which a new message belongs, and a set of messages of the same thread. The literature shows that in the sequential interface, unrelated messages often intervene between two adjacently linked messages. Displaying a coloured icon for each message can, therefore, increase coherence of a conversation.

As described below, colours are also used to create visually separated regions of threads in Tree View. Thus, using colours to match messages and threads, users can conveniently shift between Conversation View and Tree View.

Furthermore, Conversation View provides notification messages, such as "New topic" and "Start Sub Topic", as additional information about the structure of a conversation.

(C) Tree View

Tree View uses the adaptive threaded interface to provide a structured and coherent view of messages. Tree View displays all messages in a tree-based layout. Top-level tree nodes refer to main threads of a conversation. A main thread is often a topic of discussion, and a conversation can involve many topics. Sub-threads can also be created if required.

As introduced above, each thread is assigned a unique colour. This colour is used to visually separate different

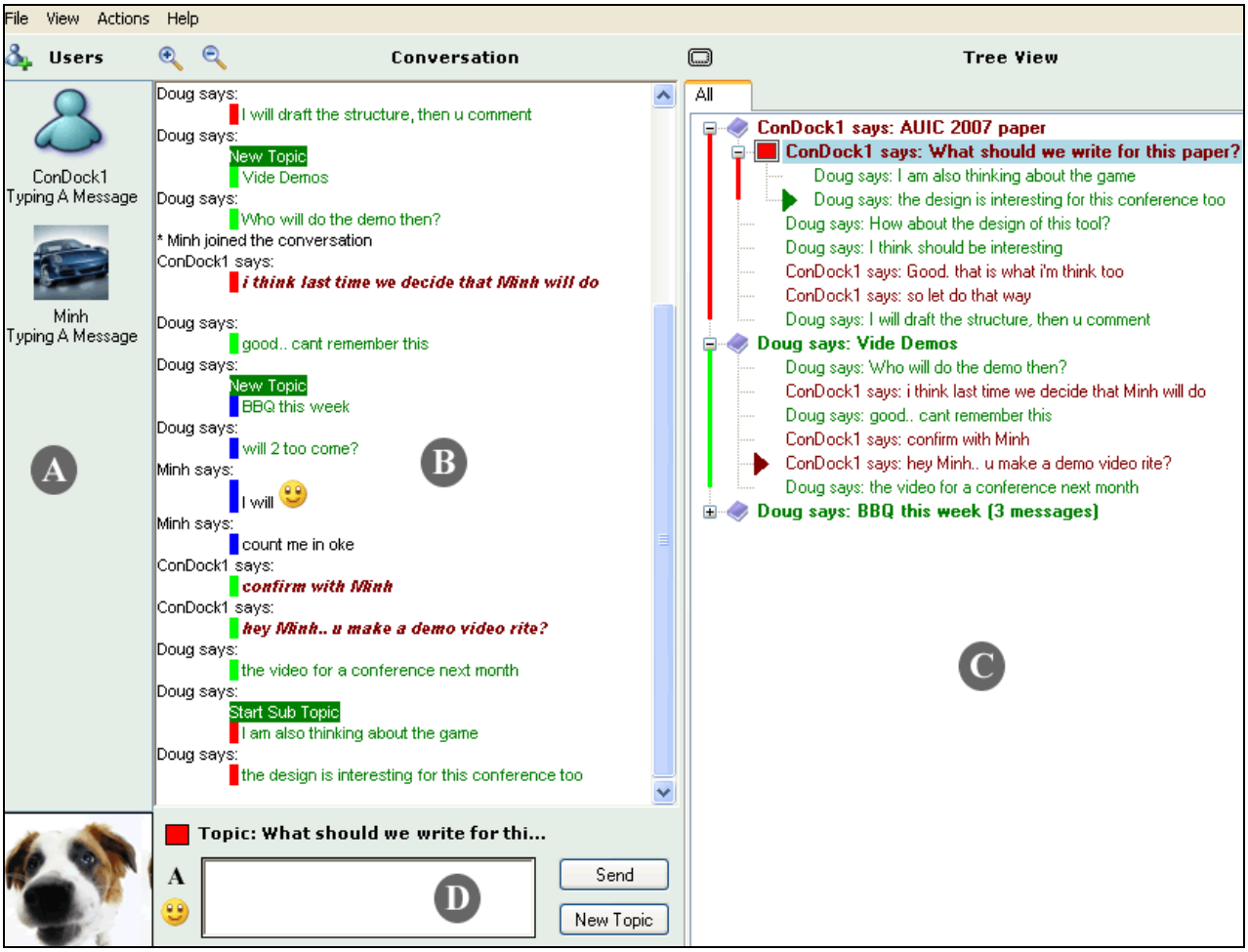

Figure 1: Swinburne Instant Messenger (SWIM).
(A) User List, (B) Conversation View, (C) Tree View, (D) Message Entry.

threads. For example, in Figure 1 (B), messages of the first thread are within the red region, whilst those of the second thread are within the green region. Tree View also indicates a current thread in which a local user is engaged. Furthermore, the last messages posted by all users are marked by arrows, as seen in Figure 1(C).

Messages can be populated in Tree View in two ways. First, users can create a new node in the message tree and type a new message at this new node. Second, users can type a new message at Message Entry (introduced immediately below), and SWIM automatically assigns this message to appropriate threads. In addition, SWIM allows users to re-allocate messages in Tree View by dragging-and-dropping from one node to another.

The appearance of Tree View is adaptive to users' interaction. That is, threads collapse or expand automatically depending on the heuristics of users' participation in the threads. Inactive threads are shown in a collapsed state, whilst active threads are expanded. A thread is considered inactive or active based on a history of messages. By default, SWIM defines that if a thread does not host any of the latest 10 messages, then it is an inactive thread. Users are able to change this default setting and customise their own rules. In Figure 1(C), the last thread, "BBQ this weekend", is collapsed because it is considered as inactive based on our adaptive rule.

(D) Message Entry

Message Entry includes a text-box component, which is used to populate users' messages in Conversation View and Tree View. A message from Message Entry appears at the bottom of Conversation View and at an appropriate thread in Tree View. An allocation of a new message to the bottom of Conversation View is straight forward, whilst assigning the message to Tree View is much more challenging. We developed the Utterance Rule-based Principle[1] (UPR), which analyses the locations of users' previous messages to automatically allocate a next message to an appropriate thread. Users can override UPR by specifying specific threads to which they want their new messages to be assigned.

In addition, Message Entry provides visual and textual cues that inform users of their current threads and the colours of the threads. For example, in Figure 1(D), a local user's current thread is "What should we write for this paper", which is coloured in red. By providing this contextual information, Message Entry helps users stay aware of the state of an ongoing conversation (e.g., to which threads their messages belong even before the message is sent).

[1] A detailed discussion of UPR is beyond the scope of this paper.

3.2 SWIM Support for Persistent Conversation

Previous research shows that persistent conversation is highly useful to group conversation and collaboration (Erickson et al., 1999). Persistent conversation helps provide the global context of a conversation. SWIM supports persistent conversation in two aspects.

First, SWIM allows users to save the content of a conversation. A conversation is saved in the XML (eXtensible Markup Language) format, which can be used for other purposes, such as data mining, querying, and so on. This XML file can also be re-loaded into SWIM afterwards if users wish to continue the conversation.

Second, SWIM supports automatic loading of the content of an ongoing conversation. If users join an ongoing conversation, the content of a conversation is automatically loaded into the users' clients. This loading feature is achieved by using a peer-to-peer (P2P) protocol. That is, when a new participant joins a conversation, a request is sent to all current participants of the conversation. One of the participants accepts this request, and responds to it by sending to the new participant the updated content of the conversation. This solution is developed due to the fact that SWIM does not have access to messages in the MSN Messenger server.

Table 1 summarises the major differences between the interface of conventional IM and that of SWIM.

	Conventional IM	**SWIM**
Interface design	Sole sequential interface or sole threaded interface	Integration of the sequential interface and threaded interface
		SWIM threaded interface is adaptive to users' interaction
Message layout	Messages are displayed in a chronological order with the most current message at the bottom	Messages are displayed in a chronological order in Conversation View, and grouped into threads in Tree View
	The order of messages cannot be altered	SWIM allows messages in Tree View to be re-allocated to maintain the structured of conversation
	No interface support for coherence	SWIM uses colours to support smooth shift between Conversation View and Tree View. Messages are coded in unique colours to support coherence (e.g., recognise related messages)
	Conversation can be saved, but cannot be reloaded for further discussion. Late comers cannot see what happens prior to their entry	Conversation is persistent, can be saved and reloaded. Late comers can see the entire conversation
User activity	A single textual "who is typing" cue	Multiple textual and visual "who is typing" cues
		Textual and visual cue informs users of their current threads

Table 1: Comparison of conventional IM and SWIM.

3.3 Implementation of SWIM

This research aims to port our interface design into commercial IM systems. We chose MSN Messenger as our targeted system for two main reasons. First, MSN Messenger is one of the most popular IM clients. This allows us to conduct a field trial with real-world users in a later stage of the project. Second, MSN Messenger provides several implementation options as discussed below.

There are two main techniques for developing an IM client using MSN Messenger. The first approach is to develop a plug-in, using the Messenger APIs (Application Programming Interfaces) to manage the interaction between the plug-in and MSN Messenger. We refer to this technique as "API-based Technique". The second approach is to develop an independent IM application using which clients are able to communicate with one another via the MSN Messenger communication protocol. We refer to this technique as "Communication Protocol-based Technique". In what follows, we briefly discuss these two techniques.

API-based Technique

The MSN Messenger APIs provide a set of interfaces for objects and events (e.g., `Messenger`, `MessengerWindow`, `OnSignIn`, `OnSignOut`, etc.), which conform to Component Object Model (COM) automation. Via the interfaces of those objects, we can access and manipulate objects of MSN Messenger. Figure 2 illustrates the architecture of the API-based approach.

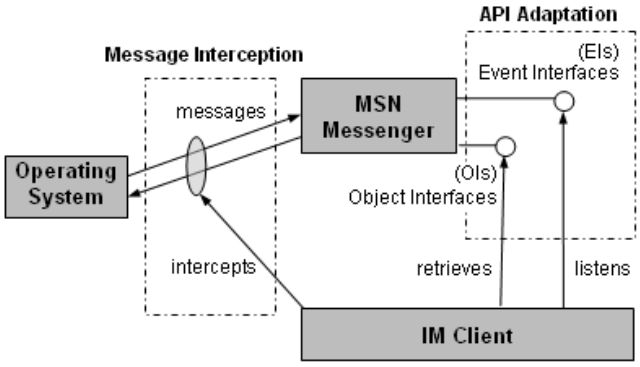

Figure 2: Architecture of the API-based Technique.

In this approach, an IM client needs to interact with MSN Messenger using object interfaces and event interfaces provided by the Messenger APIs (i.e., API Adaptation). Furthermore, the IM client needs to intercept appropriate message calls sent between the Operating System (OS) and MSN Messenger (i.e., Message Interception).

We have already applied this technique in the implementation of ConDock (Tran et al. 2006). Based on our experience, although the MSN Messenger APIs provide options for accessing and manipulating objects of MSN Messenger, the APIs are still limited, especially many events of MSN Messenger are not accessible. Furthermore, in order to implement the adaptive threaded interface, we need to manipulate text messages sent between MSN Messenger clients. This task is difficult to

accomplish using the MSN Messenger APIs due to the incompleteness of the APIs.

Moreover, the current version of MSN Messenger adopts the windowless technology, and this makes difficult to intercept messages sent between OS and MSN Messenger.

Communication Protocol-based Technique

MSN Messenger allows other third-party applications to interact with the Messenger communication protocol (http://www.hypothetic.org/docs/msn/). Adopting this protocol, we can develop an IM system of which clients are able to communicate with one another via the Messenger server and switchboard. Popular multi-protocol IM clients, such as Trillian (Cerulean Studios, 2006) and Gaim (Gaim, 2006), are representative examples of how to utilise the MSN Messenger communication protocol.

In comparison to the API-based Technique, the Communication Protocol-based Technique offers developers:

(1) more control on how they want to manipulate messages sent between IM clients, and

(2) more options of capturing events at local and remote clients.

However, these beneficial features do not come for free. This technique requires an understanding of the MSN Messenger communication protocol, and involves developing a corresponding software library to interact with the protocol. Fortunately, there are some open-source libraries, such as DotMSN (Xih Solutions, 2006), that can be used to communicate with the MSN Messenger protocol.

Figure 3 shows a general software architecture of the Communication Protocol-based Technique. Two major software components of the architecture are Adaptation Library and IM Client.

The Adaptation Library component plays the role of a middle software layer, sitting between an IM client and the MSN Messenger communication protocol. Adaptation Library is responsible for interacting with the MSN Messenger communication protocol, such as establishing a connection and authentication with the MSN Messenger server, and sending and receiving messages via the MSN Messenger switchboard.

The IM Client component listens to users' interaction and passes captured events and messages onto Adaptation Library. In addition, the client is responsible for manipulating the contents of messages sent from other clients for presenting the adaptive threaded interface.

In this research, we adopt the Communication Protocol-based Technique to develop SWIM. We use and enhance DotMSN library to communicate with the MSN Messenger communication protocol. SWIM is written in C#, using Microsoft .NET Framework (http://msdn.microsoft.com/netframework/).

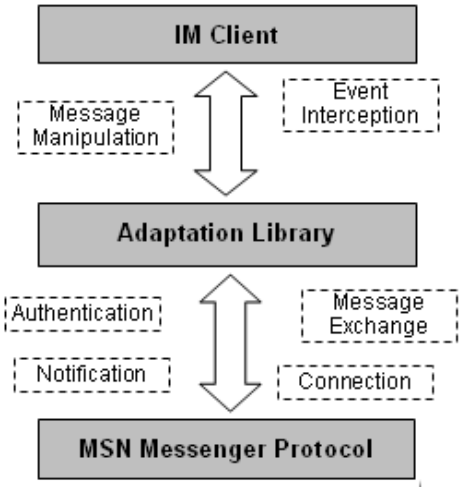

Figure 3: Architecture of the Communication Protocol-based Technique.

3.4 Preliminary Evaluation of SWIM

We are planning for a field trial of SWIM. So far, we have evaluated SWIM by using it amongst our research group. In most cases, SWIM was used by a group of two and three people, and in a few cases SWIM was used by a group of four people. This preliminary evaluation was useful because it helped us identify observable issues related to the design and implementation of SWIM. In addition, this preliminary evaluation was useful for designing materials of the field trial (e.g., tasks, data collection technique, etc.). This section presents some initial observation and feedback from the preliminary evaluation of SWIM.

We have used SWIM to discuss various topics. The system particularly was found useful in presenting a structured and coherent conversation. For example, we used SWIM to carry out a discussion about the design of SWIM itself. The discussion covered various aspects of software and interface development, such as programming issues, testing bugs, interface layout, and so on. We found that SWIM was really useful in supporting our ongoing discussion in the sense that SWIM assisted us significantly in sustaining our discussion in a structured and logical manner. Using SWIM, we could easily organise related messages into a thread, and monitor how the thread is evolving.

In addition, we found that having a persistent conversation was highly useful. It often occurred that not every person of a group was able to join a conversation at the same time: some people joined a conversation late. SWIM allowed those late comers to view the entire content of a conversation. Furthermore, a history of messages was organised in a structured way, and this assisted the later comers significantly in understanding the conversation.

An analysis of the structure of SWIM conversations showed that users often started their conversation with social greetings (e.g., "hello", "how are you", etc.). At this point, there was no clear topic or thread defined, hereby SWIM functioned like MSN Messenger. After this

greeting period, users then moved into a main conversation. We observed that in order to utilise SWIM, users needed to agree on what topics they would discuss. This helped identify the top-level threads of a conversation. In some cases where top-level threads were not identified clearly, disrupted turn adjacency occurred more often in SWIM (e.g., related messages interweave with unrelated messages).

When a message happened to be posted in an unrelated thread, SWIM allowed users to re-position the message easily simply by dragging the message to a relevant thread. This drag-and-drop feature was useful in monitoring the structure of a conversation.

Although no quantitative data were collected from this preliminary evaluation, the evaluation did give us positive feedback on the usefulness of SWIM. In the upcoming field trial, we will focus on both the usefulness and usability of SWIM. We will also examine to what extent SWIM improves group discussion, in comparison to MSN Messenger.

4 Discussion

This section discusses some potential enhancements to SWIM, and lessons that we have learnt from the design and implementation of SWIM.

4.1 Potential Enhancements to SWIM

SWIM currently supports the adaptive threaded interface, based on users' interaction in a conversation. SWIM defines a "degree of interest" variable to calculate users' involvement in a particular thread, which then leads to a visibility of the thread (e.g., collapsed or expanded). From the evaluation of SWIM, this concept is found useful and it could be extended to model coupling[2] in a group. A visualisation of coupling is able to provide users with contextual information about social interaction within a group.

SWIM combines the sequential interface and the adaptive threaded interface to facilitate group conversation. The former interface is used to track the chronological order of messages, and the latter interface is used to form the structure of messages. Although SWIM already provides several mechanisms assisting users in shifting between the two interfaces, we realise that it is useful to support *multiple focal points* in the threaded interface.

Multiple focal points refer to a visualisation technique of presenting all users' the latest messages in a single window without scrolling. From our experience in developing SWIM, showing all users' the latest messages in the threaded interface is challenging because these new messages can be posted at any node of a tree. For instance, one message could be in the first thread while another message could be in the last thread, and the two threads are not visible simultaneously in a local user's viewport. One possible solution for showing multiple focal points is by adopting a focus+context visualisation technique. For example, a fisheye view (Weir and Cockburn, 1998) can be used to render all threads in a single window. The fisheye view magnifies selected messages around focal points whilst de-magnifies less recent messages.

Another potential enhancement to SWIM involves a further development of URP. Currently, URP analyses a user's history of message to predict the location of the user's new message. We find that URP can be enhanced by incorporating its current approach with a semantic analysis approach. That is, in addition to analysing users' message history, UPR could examine the content of a new message and consider how the new message relates to available threads semantically.

4.2 Lessons Learnt

To present the adaptive threaded interface, SWIM maintains a data structure and maps this data structure with the MSN Messenger sequential interface. Although this data structure is manageable in SWIM, we find that the mapping between the data structure and the sequential messages is quite unnatural and somewhat inefficient. Currently, the MSN Messenger communication protocol supports formatting properties of a message (e.g., font, colour and font style), but it does not support structural properties of a message (e.g., information about the hierarchical position of a message). As a result, SWIM needs to attach a special header to a message and extracts this header at remote sites. By manipulating this header, SWIM constructs the threaded interface of messages.

From our prior research on integration of the sequential interface with the threaded interface, and from the preliminary evaluation of SWIM, we find that this integrated interface offers great potential in facilitating structured and coherent group conversation. For implementing this model efficiently, the MSN Messenger communication protocol should be extended in such a way that it allows users to be able to specify structural properties of a message.

5 Conclusions and Future Work

This paper presents SWIM (SWinburne Instant Messenger), an innovative Instant Messaging tool that is able to support structured, coherent group conversation. The interface of SWIM involves integrating the conventional sequential interface with the adaptive threaded interface. The sequential interface is used by SWIM to provide the flow of a conversation (e.g., a chronological order of messages). The adaptive threaded interface is used to maintain the structure and coherence of a conversation. The appearance of this structured layout is able to adapt automatically to users' participation in a conversation. In addition, SWIM maintains persistent conversation, which is used to support users' interaction in group discussion.

We report some findings from the preliminary evaluation of SWIM. The preliminary evaluation shows that the interface of SWIM is useful in facilitating structured

[2] In our research, the term "coupling" is used to refer to the extent to which two users interact with one another in a conversation.

group conversation. By maintaining both the flow of messages and the structural layout of messages, SWIM allows users to follow a conversation easily and keep track of messages from different topics conveniently. SWIM also helps users easily recognise a set of related messages. Furthermore, the adaptive threaded interface of SWIM is also found useful in helping users organising the presentation of threads.

As future work, we will improve the adaptive threaded interface of SWIM to accommodate multiple focal points. We also need to examine social coupling between members of a group based on their conversational behaviour. Another important aspect of the future work is to evaluate SWIM by conducting a field trial. The evaluation will focus on the usability of SWIM and comparing SWIM with current IM tools, such as MSN Messenger, in supporting group discussion.

6 Acknowledgement

This project is supported by Smart Internet Technology CRC (SITCRC).

7 References

Cech, C. G. and Condon, S. L. (2004). Temporal Properties of Turn-Taking and Turn-Packaging in Synchronous Computer-Mediated Communication. *Proceedings of the 37th Hawaii International Conference on System Sciences*, Big Island, Hawaii, IEEE Computer Society Press, pp. 107-116.

Cerulean Studios (2006). http://www.ceruleanstudios.com, Accessed 10 August 2006.

Erickson, T., Smith, D. N., Kellogg, W. A., Laff, M., Richards, J. T., and Bradner, E. (1999). Socially Translucent Systems: Social Proxies, Persistent Conversation, and the Design of "Babble". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI '99*, Pittsburgh, Pennsylvania, ACM Press, New York, pp. 72-79.

Gaim (2006). http://gaim.sourceforge.net, Accessed 10 August 2006

Garcia, A. and Jacobs, J. (1999). The Eyes of the Beholder: Understanding the Turn-Taking System in Quasi-Synchronous Computer-Mediated Communication. *Research on Language and Social Interaction*, 32(4), pp. 337-367.

Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D. J., and Kamm, C. (2002). The Character, Functions, and Styles of Instant Messaging in the Workplace. *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work CSCW'02*, New Orleans, Louisiana, USA, ACM Press, New York, pp. 11-20.

Kurlander, D., Skelly, T., and Salesin, D. (1996). Comic Chat. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH'96*, New Orleans, Louisiana, ACM Press, New York, pp. 225-236.

Muller, M. J., Raven, M. E., Kogan, S., Millen, D. R., and Carey, K. (2003). Introducing Chat into Business Organizations: Toward an Instant Messaging Maturity Model. *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work GROUP'03*, Sanibel Island, Florida, USA, ACM Press, New York, pp. 50-57.

Nardi, B. A., Whittaker, S., and Bradner, E. (2000). Interaction and Outeraction: Instant Messaging in Action. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work CSCW'00*, Philadelphia, Pennsylvania, US, ACM Press, New York, pp. 79-88.

O'Neill, J. and Martin, D. (2003). Text Chat in Action. *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work GROUP'03*, Sanibel Island, Florida, ACM Press, New York, pp. 40-49.

Smith, M., Cadiz, J. J., and Burkhalter, B. (2000). Conversation Trees and Threaded Chats. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work CSCW'00*, Philadelphia, Pennsylvania, ACM Press, New York, pp. 97-105.

Spiegel, D. (2001). *Coterie: A visualization of the conversational dynamics within IRC*. Master's thesis, MIT.

The Palace. http://www.palacetools.com, Accessed on 10 August 2006.

Tran, M. H., Yang, Y., and Raikundalia, G. K. (2005). Supporting Awareness in Instant Messaging: An Empirical Study and Mechanism Design. *Proceedings of the Australian Conference on Computer Human Interaction OzCHI'05*, Canberra, Australia, ACM Press, ISBN number: 1-59593-222-4.

Tran, M. H., Yang, Y., and Raikundalia, G. K. (2006). The Transparent Adaptation Approach to the Development of Awareness Mechanisms for Groupware. *Proceedings of the Australian Software Engineering Conference ASWEC'06*, Sydney, IEEE Computer Society, pp. 142-151.

Viegas, F. B. and Donath, J. S. (1999). Chat Circles. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, ACM Press, New York, pp. 9-16.

Vronay, D., Smith, M., and Drucker, S. (1999). Alternative Interfaces for Chat. *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology UIST'99*, Asheville, North Carolina, ACM Press, New York, pp. 19-26.

Weir, P. and Cockburn, A. (1998). Distortion-Oriented Workspace Awareness in DOME. *British Computer Society Conference on Human-Computer Interaction*, Sheffield Hallam University, Sheffield, Springer-Verlag, pp. 239-252.

Xih Solutions (2006). http://www.xihsolutions.net/dotmsn/, Accessed 10 August 2006.