Working Paper Series
ISSN 1170-487X

Proceedings of the

OzCHI'96 Workshop on the Next Generation of CSCW Systems

Edited by:
John Grundy

Working Paper 96/29
November 1996

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Proceedings of the

OZCHI'96 Workshop on the Next Generation of CSCW Systems

25th November, 1996

Hamilton Gardens, Hamilton, New Zealand

Edited by John Grundy

Preface

This is the Proceedings of the OZCHI'96 Workshop on the Next Generation of CSCW Systems. Thanks must go to Andy Cockburn for inspiring the name of the workshop and thus giving it a (general) theme! The idea for this workshop grew out of discussions with John Venable concerning the Next Generation of CASE Tools workshop which he'd attended in 1995 and 1996. With CSCW research becoming more prominent within the CHI community in Australasia, it seemed a good opportunity to get people together at OZCHI'96 who share this interest. Focusing the workshop on next-generation CSCW system issues produced paper submissions which explored very diverse areas of CSCW, but which all share a common thread of "Where do we go from here?", and, perhaps even more importantly "Why should we be doing this?".

The workshop was intended to be a fairly informal forum for discussion between people coming to OZCHI'96 and interested in CSCW systems, particularly the future of CSCW systems. All of the papers in this proceedings have more or less of a next-generational theme. The papers have been grouped into those offering insights into next-generational theoretical and/or foundational concepts, next -generational CSCW architectures, and new or insightful applications or evaluations of CSCW systems. Many of the papers, of course, cover more than one of these areas.

A great vote of thanks must go to the Workshop Committee for refereeing all of the submissions to the workshop and ensuring the papers in this proceedings are to a high standard. All papers were anonymously reviewed by at least two members of the workshop committee, with several having three reviews. A good deal of interest has been expressed by people around the world of having the papers available on-line (as HTML and/or Postscript). We may investigate this in the near future, if all authors agree.

It is hoped that for OZCHI'97 (incorporated with INTERACT'97 and APCHI'97), a similar workshop focusing on CSCW systems will be proposed and run. I hope you enjoy the papers in this proceedings and they motivate and inspire you in your own work in CSCW, or inform you of advances in the area.

John Grundy
November, 1996

Workshop Committee

Andy Cockburn, University of Canterbury

Simon Gianoutsos, University of Waikato

Saul Greenberg, University of Calgary

John Grundy, University of Waikato

John Hosking, University of Auckland

Steve Jones, University of Waikato

Simon Kaplan, University of Queensland

Chris Marlin, Flinders University

John Venable, University of Waikato

Contents

Towards supporting remembering in organizations by a CSCW tool -- dimensions of "organizational memory"

Kari Kuutti

Center for HCI and CSCW Research
University of Oulu, Dept. Information Processing Science
Linnanmaa, FIN-90570 Oulu, FINLAND
e-mail: kuutti@rieska.oulu.fi

Abstract

Introduction

The idea of "organizational memory" has been around already for a while, and although it normally is conceived as a special kind of CSCW system, it has gained interest widely beyond the CSCW community. Usually, the idea has followed the simple storage metaphor borrowed from standard cognitive science, where human and computer memories are seen to be to a large extent similar, and where "storing" and "recalling" are symmetrical operations which do not lead to any situational complexities.

This, however, is not a very accurate picture of what happens to real people in in real organizations where simple recalling is rarely adequate but where any data has to be reinterpreted -- either with respect to original or recent conditions. So in Bannon & Kuutti 1996 we started to deconstruct the idea of "organizational memory" by reformulating the notion of memory implicit if not explicit in most current views, i.e. the notion of memory as a passive store, arguing instead for an active, constructive view of "remembering" that has a long, if forgotten history within psychology. We also emphasised the need for empirical studies of "memories in use" and common information spaces, and suggested expanding the domain of discourse to include sociological as well as psychological perspectives on concepts such as memory, learning, remembering, talking, etc. in the context of organizations.

The paper was, however, mainly a critical review that did not make much progress in constructing alternative formulations what kind of systems would be needed to move from storage to remembering support. This paper takes one step further by drafting a hypothetical structure of the framework where remembering for acting in situations takes place, and thus also for a structure that a support tool should possess. The framework is constructed by connecting two ideas: the modality of remembering and the structure of activity.

The modality of remembering

One of the most charasteristic features of active remembering is that it is puposeful and situational. Typically, we remember something to facilitate acting in a situation. Whatever facts we have at hand or are able to recall, they have to be reflected against the situation, otherwise they cannot make sense. When this situationality is studied further, an interesting feature can be recognized, namely that we do not remember only the past, but also the present and the future.

In many languages the verb "remember" has also the meaning "to know" or "to be able to keep in mind", and this meaning focuses on what is currently happening: "Have we now remembered everything?" -- in order to do the next action. Furthermore, this kind of remembering extends from the present to the future as well: "Remember to turn left from the next corner!". Only when this kind of remembering fails, a special recalling action has to be engaged. In practical situations remembering past, present and future merges together: to act in situation means that present has to be understood in the light of the past, and that it has to be reflected against what will and should happen in the future. It seems thus plausible that if we want to facilitate acting in situations by supporting remembering, something for this past, present and future feature has to be offered.

The structure of activity

Another "dimension" along which remembering can be situationalized and contextualized is the structure of the context where it is taking place. Several frameworks could be used for this purpose; in this paper the structure of activity from Activity Theory is utilized. During the last years Activity Theory has started to gain more and more popularity within HCI and CSCW communities as one of the theoretically well-founded frameworks to

understand and analyse work situations. A good overview of it can be found in the recent collection of articles published by MIT Press (Nardi 1996).

One of the core principles of Activity Theory is that human actions can be understood and studied only within a contex that gives them sense and meaning. This context is defined through the existence of a common object that is transformed by actions of participants for a purpose. Individual actions can be understood only against the background of this common object and purpose, and thus these have to be included into the unit of analysis, which is called an activity (hence the name of the theory). Y Engeström has proposed (1987) the following structure of such activity:

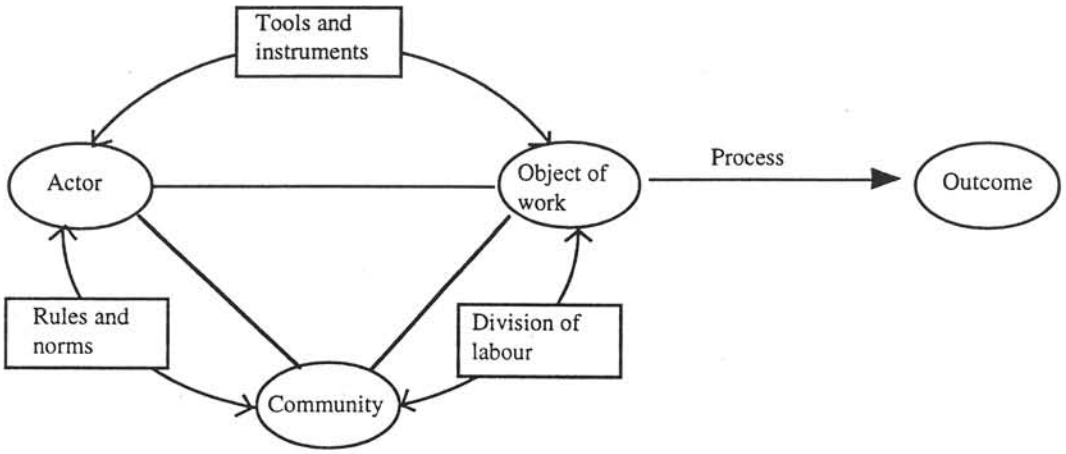

Fig. 1 The overall structure of an activity (modified from Engeström 1987)

This overall structure can be used in defining three levels of observation: the level of actions, the level of the structure and functioning of an activity, and the level of the object of an activity. The focus of observation can be in individual actions that contribute to the transformation process of the object into the outcome; the focus can be at the level of the structure of an activity: who belongs the community sharing the same object, what is the division of labour between them, what are the tools and instruments available, what are the steps in the transformation process etc. Finally, the focus can be at the level of the object and purpose of the whole activity: what is the purpose, why is this activity existing?

The resulting structure

By combining these two "dimensions", a hypothetical "field of situational remembering" can be generated. It is depicted in Fig. 2.

	Past	*Present*	*Future*
Object	"Object-history"	Current conception of object	Changes in the "external world"
Structure and functioning of an activity	Development path	Current structure & functioning	Development potentials
Actions	Records what was done	Current situation	Plans, trajectories

Figure 2. The "field of remembering".

In the proposed "field", each of the "cells" forms a potential reference point against which the content of another cell may be reflected in order to make sense in a situation. The connections are not homogenous, but different patterns emerge in different situations. Thus the "current situation" cell contains all the "facts" and actions to be kept in mind in order to act correctly in that situation. From time to time the significange, correctness etc. of these has to be reflected and judged against what has happened in the past or against their potential influence what is going to happen in the future. The content of the "current situation" cell can also be reflected vertically against the structure of the activity ("Who is going to deliver this piece? What tools we have available for this purpose?"), or ultimately, against the purpose of the whole activity.

In other occasions the results of actions in the past have to be reflected against the structure of the activity within which they were produced, or the purpose they served at that time. Or, the future actions may need judging against the development potential in the structure or perhaps a potential shift in the purpose of an activity.

Conclusion

A hypothesis is made that in situational remembering the current "facts" to be kept in mind have to be reflected against past and future; and that they have to contextualized within the activity where remembering is taking place. If this hypothesis holds, the natural next step is to assume, that a CSCW tool to support such remembering — an organizational memory, or better a "common information space" (Schmidt & Bannon 1992) — should have a similar structure.

This suggests a two-pronged research strategy: validation of the hypothesis by studying remembering in actual situations, and construction of prototype systems to support it along the forementioned lines.

References

Bannon, L. & Kuutti, K. (1996) Shifting Perspectives on Organizational Memory: From Storage to Active Remembering. Proceedings of the HICSS-29, IEEE Computer Press

Engeström, Y., *Learning by expanding.* 1987, Helsinki: Orienta-konsultit.

Nardi, B.A. (Ed.) (1996)*Context and Consciousness: Activity Theory and Human Computer Interaction.*. The MIT Press: Massachusetts, Cambridge.

Schmidt, K. and L. Bannon (1992): Taking CSCW Seriously: Supporting Articulation Work. *Computer-Supported Cooperative Work*, vol. 1, no. 1.

How do you specify a CSCW system?

Steve Reeves
Department of Computer Science
University of Waikato
Hamilton
New Zealand
stever@waikato.ac.nz
http://www.cs.waikato.ac.nz/~stever

Abstract

The rôle of both formal and informal specification is well-established and, for many systems, clearly necessary. However, there appears to have been little work so far on the specification of CSCW systems. In an attempt to ensure that development of such languages keeps pace with the development of ideas in the implementation of CSCW systems we need to start now. Here we argue for specification languages for such systems, make some observations about what such languages should be like and show examples of their use.

1. Introduction

This paper came out of trying to answer the question "How do you specify a CSCW system?". More exactly I asked myself "What does it mean to say that you have specified a CSCW system?" and also "Can you do it formally—so that you can unambiguously and precisely describe it?".

Recall that specifying a system means saying what a system does and not going into the implementation details of how it does it. This allows us to describe the system in as clear a way as possible since we do not risk obscuring the system itself by having to refer to its implementation details. It means that we have a precise way of writing down a design without having to make an early (perhaps too early) commitment to our implementation methods.

Another reason for trying to do our specifying formally is that we may wish to have a description that could, for example, form the basis of a contract between user and developer.

Alternatively, the system may have safety- or security-critical aspects that we want to be sure of communicating correctly and unambiguously. Also, at least one governmental agency (the Australian Department of Defence) now requires that critical systems be developed using formal methods, see [1]; others (for example the British Ministry of Defence, see [3) have enforceable Standards that mandate the use of formal methods.

In any case, software developers who produce systems for use by others are under a general requirement (under laws about negligence, for example) that means products should be developed using the best available means and incorporating the best available techniques. Finally, professional pride and standards should mean we all want to use the best methods to reach our goals.

Interactive systems, which exhibit some of the same problems for the specifier as CSCW systems, have, in the last few years, started to attract the attention of people working in formal methods ([2] is good example of very recent work). However, CSCW systems themselves have not yet been widely studied by the formal specification community, though they have special properties which differentiate them from other systems and which provide a special challenge to the specifier.

2. The essence of CSCW systems

In order to answer our questions we really need to go back to the basic ideas and make seemingly simple-minded observations in response to naïve questions. So, what is a CSCW system (not how does one work)? Essentially it is a system of software that allows its users to collaborate in some task. (A nice example would be the shared editing of a document.) It seems to me that there are two obvious parts of such a system at the level of what it must do. There is the straightforward software part which, in our example, allows a user to edit a document. This is fairly standard as far as the software goes, though there is the added complexity of things like locking people out of editing exactly the same part of the text, keeping track of different versions and allowing amalgamation of drafts. All of these things can be expressed in terms of what the software does.

The other part of such a system, and this is what is really new about a CSCW system, is that both the user and the designer clearly have in mind some model of what other users of the system are aware of about their and others' use of the system and what other users of the system know about the system.

So when you as the designer decide that the presence of people editing the documents shall be indicated in some way, a specification of that system has to say, for instance, that when somebody new joins the editing session their presence must be indicated to the other users of the document so that they can build the presence of this new person into their model of what it means to be co-operatively editing the document.

At the specification level the designer might say that when a user is editing a paragraph all the other users must know that someone is editing it (and perhaps they must know who) and (depending on the requirements for the system) that they must not be able to edit it at the same time.

Another example: if someone is editing a paragraph and some others are trying to edit it then the specification might need to say that the person doing the editing must know that others are trying to edit it.

Finally: each user should be aware that when they perform actions, the other users will be aware that they have performed them.

All these things are examples of the second sort of part of a specification of these systems and they clearly specify what users must be able to tell about each other—their activities and the things that they are aware of. Notice that they do not have to go into detail about how, for example, other users should be aware of the presence of someone (e.g. by using a telepointer or several scrollbars)—this refinement process, towards an actual implementation, can go on later (and again need not be in programming language—our specification language might be able to do this job too). What we are trying to do currently is describe at a high level what users should be aware of.

To recap: CSCW systems seem to fall into two parts. There is the standard part which says what the software does in terms of the usual sorts of constructions. Then there is the part which specifies what a user must be told about other users in the system and their use of the system. In other words, there must be a way of the user building a correct model of the complete system in terms of both the state of the software and what other users are doing.

3. Languages

This two part analysis came about both by reflecting on actual CSCW systems but also by having in mind what techniques and languages we might use to specify such systems.

The first part can be specified by what are becoming fairly standard action-and-agent logics. That is, languages which allow you to write sentences (amongst others) of the form "if these pre-conditions P hold then when any agent A does this action a then this post-condition Q will hold afterwards" so for example

for all A : Agent .
if cursor_over(open_button)
then [after A does push]
highlighted(open_button)

or

for all A, B : Agent; a : Action.
if text_field_contains(B)
and cursor_over(lock_out)
then
[after A does push]not can_do(B,a)

Note that these sentences can be given a completely formal semantics, so we do not have to rely on some informal, suggestive meaning for them (based on what we think the words are probably intended to mean)—a failing that many 'languages' have. This means that not only does the meaning not have to depend on what the reader happens to think the words mean but that we can provide tools to support the language (for parsing, editing, typechecking, for example) and, perhaps more importantly, tools to support exploration of the consequences of these sorts of sentence via proof- and model-builders and checkers.

The second part is the part which says what users must be told about by the system in order to model the co-operation that is going on: it is not at all standard.

We have started developing, by adapting existing logics of knowledge, a language and logic which allows us to talk about and reason about what people should be aware of when they use the system.

This work has three parts itself: firstly we have to come up with a suitable logic which has the right sorts of properties; secondly, having either developed or discovered an appropriate logic, we have to think up general

properties of things like awareness and knowledge of other people, that is we have to axiomatize the ideas of awareness and knowledge that people need when they are using these systems; thirdly, we have to show how typical requirements, which would be specific to the system being developed, can be modelled within that language.

We also have to show how we can reason about those situations.

An example general property could be stated as

for all x; A : Agent.
if aware_of(A,x) then aware_of(A,aware_of(A,x))

and an example typical requirement could be stated as

for all A, B : Agent.
if present(A) and present(B)
then aware_of(A,B) and aware_of(B,A)

There is not room here to give any more detail of this work (which is still going on), but [4] contains more on it. Again, the language for this second part has a well-defined semantics (though this is not easy to achieve and the straightforward adaptations of modal logic used for the first part are not adequate), so mechanized support is possible.

4. Conclusions

When all these parts are brought together, what will we have?: a language and a logic which allows us to specify what our CSCW system will be like. That specification will consist of two parts: one that says what the software does at the level of atomic behaviours and interactions of the people with it; the second part will specify that bit of the system which specifies what a user's model of the system will be, so it will say, for instance, "when a new user joins the system all the users must be made aware of that fact", "when a user does action a all other users should be aware of that action", "when a user does action b then any users who are in the neighbourhood of where that action was carried out should be aware of it".

So the second part will allow all users to build a model of the whole system: both the software part and the people using the software- their interaction with the software and each other.

To sum up: it seems to me that if you do not have a language that does both these things then you cannot claim to have a language which allows specification of a CSCW system. This is because a CSCW system consists not only of the software and the people interacting with the software but also of the model that the people doing the interacting have of the system as a whole, as it changes over time. And for that you need the second part—you need a part that will allow description of the models that people must build of the system.

Acknowledgments

This work is being carried out in the CSCW group in the Department of Computer Science at the University of Waikato, New Zealand under a Foundation for Research, Science and Technology grant.

References

[1] Australian Ordnance Council Pillar Proceeding 223-93.

[2] Hussey, A. and Carrington, D. (1996) Using Object-Z to Specify a Web Browser Interface, Proceedings of OzCHI'96, IEEE Computer Society Press, 1996.

[3] Ministry of Defence, The Procurement of Safety Critical Software in Defence Equipment, interim defence standard 00-55, Ministry of Defence, Directorate of Standardization, Kentigern House, 65 Brown Street, Glasgow G2 8EX, April 1991.

[4] Reeves, S.V. (1996) Specifying and Reasoning About CSCW , Proceedings of DSV-IS'96, Springer-Verlag, 1996.

Orbit & Support for Pervasive Collaboration

Simon M. Kaplan[1,2,3], Geraldine Fitzpatrick[1], Tim Mansfield[2]
[1]Department of Computer Science, The University of Queensland
[2]CRC for Distributed Systems Technology, the University of Queensland
[3]Department of Computer Science, University of Illinois at Urbana-Champaign
{simon, ger, timbomb}@dstc.edu.au

Abstract

For the past several years we have been investigating collaboration support frameworks that provide support for the informal, cultural aspects of workaday activities as well as the formal, structured aspects of work traditionally associated with workflow systems and other "groupware" tools. A major issue in this investigation is the identification of theoretical approaches or models which can inform systems development and function as "bridges" to the ethnographic investigations of work situations currently popular within the Computer-Supported Co-operative Work (CSCW) community. Recently, we have focused on Anselm Strauss' notion of social world as a theoretical model which might address this issue. Drawing on our prior experience with developing CSCW systems, we have constructed a CSCW support environment called wOrlds (Work Locales and Distributed Social Worlds) to evaluate our theoretical approaches and investigate technologies for support of collaborative work. In this paper, we sketch the origins of the wOrlds project in its predecessor system, ConversationBuilder, and briefly critique its failings and our move towards the notion of social worlds derived from the work of sociologist Anselm Strauss. We outline our current wOrlds system and the way in which we have interpreted social worlds in the context of computer-based collaboration support. This support is provided by the implementation of locales which support the interactions of social world members. We then critique both the implementation of wOrlds and the concepts used to shape its construction, with a view to identifying some significant open problems in the construction of CSCW technologies and to point the way to possible solutions.

1. The wOrlds system

Our work on wOrlds has proceeded on two parallel tracks that shape and inform each other: a theoretical track developing the notion of *locales* and a practical track building our prototype system.

1.1. Locales

We have learned from Strauss that as people work, social worlds continually form and dissolve, as needed. For us the essential question for wOrlds was how we could support *contexts for work* embedded in the computer such that members of social worlds could then use to accomplish their tasks. When groups are working, they need the following:

- The family of artifacts that make up the "formal" layer of their work activities. Examples include program files, medical records, yellow stickies, stripcharts, etc.
- The tools that are used to manipulate these artifacts, such as compilers, editors, debuggers, pens, ECG machines, etc.
- Resources for "effective communication" which grant members of the social world the ability to communicate appropriately to the task at hand.
- Automation of mundane tasks, such as change notifications, where appropriate.
- The ability to navigate, i.e. to seamlessly switch among multiple ongoing tasks, interrelate them as appropriate, and find tasks and people as needed.

The question then becomes: what is it that we can provide through a computer network that allows the construction of contexts of work such that the needs outlined above are met? Our answer is to introduce the notion of *locale*.

A locale for the purposes of wOrlds is a "virtual space" inside the computer system which is intended to be the vehicle for groups to establish shared contexts. Many locales can exist simultaneously, and context switching

among them should be relatively easy. Further, each locale should provide the following:
Furnishings. Access to shared objects and tools which can be manipulated by users as necessary. Although the specific objects and tools used to furnish a locale will vary widely from locale to locale, depending on its purpose, every locale will provide audio and video conferencing among all the users "in" the locale at a given time.
Participants. Participants are users who may or may not be present in the locale at any given time, but who have ongoing responsibility, defined by roles, inside the locale.
Visitors. Visitors are users who are in the locale at a given moment, although they may or may not have any particular roles there. Entry to locales can be restricted when required.
Trajectory Schemas or Processes. Processes define the flow of information and control of actions. Processes generally *span* locales, and can be seen as a way both of automating standard aspects of workaday activities where appropriate and useful, and as a way of grouping locales together,
Actions. Actions with user-defined semantics can be invoked at appropriate times, subject to satisfaction of guards and other constraints.

Additionally, locales provide ways of browsing and viewing information, establishing Audio/Video (AV) calls among users and navigating through the collection of locales.

Thus, locales in wOrlds are our current way of "affording" social worlds interactions via the computer, by providing a framework for construction and support of work contexts.

2. Critique and future directions

In this section we will critique wOrlds from the perspectives of both theory and technology. Critiques, almost by definition, tend to focus on the negatives but despite the points and issues raised below we consider the wOrlds experiment to have been successful. We have built a useful CSCW system that has been applied in several different domains, we have been able to demonstrate that a non-action-centered approach to CSCW support can indeed support both formal and cultural aspects of work and we have learned an enormous amount about the subtleties of CSCW support that we simply could never have learned without the hard practical experience of building and working with a system such as wOrlds.

Most importantly, the experience of constructing wOrlds has allowed us to envision its successor system: *Orbit*, in which we hope to answer many of our own criticisms. Where appropriate we will counterpoint our critique of wOrlds with elements of the planned design of Orbit.

2.1. Positive aspects of wOrlds

By their very nature critiques tend to focus on the negative; here we briefly point out some of the many positive features of the wOrlds system.
Multimedia Rooms. wOrlds demonstrated the many benefits of multimedia-based interaction in locales, including the extensive and ubiquitous integration of audio and video.
Shared Objects. wOrlds users tend to make extensive use of the underlying shared-distributed-objects-everywhere framework in the system, which supports the ability to create rich tapestries of objects.
Contextual Action. Users found the ways in which wOrlds, especially the Introspect aspects, supported contextual action a useful way of obtaining guidance and having necessary actions at their fingertips.
Accessing People and Locales. The ability to call people and glance into or warp to locales is a continually-used feature of the system.
Integrating Mail & Web. The seamless integration of email and the web, allowing one to, for example, call someone froma web page or glance a locale from an email message (or vice versa), is a very heavily used feature of the system.
External Object Integration. wOrlds users could integrate external objects (files, etc) with their shared ORB-based objects and make extensive use of this facility.
Navigation Facilities. wOrlds provided a rich navigation tool for finding users and locales which was much relied on by wOrlds users.

2.2. Theoretical issues

One way of thinking about wOrlds, and MUDs more generally, is that they are about setting boundaries around

spaces and then populating those spaces. While we have no problem with populating spaces, we believe that the joint focus on setting boundaries and on spaces are misplaced and lead to fundamental problems with collaboration support. The body of this section really critiques wOrlds indirectly, through discussion of several key concepts which are missing from wOrlds and many other MUD-like systems. In many cases we do not have more than the broadest sense of possible solutions to these issues, they are open problems for us at this time.

2.2.1. Idiosyncrasy and intensity. People are players in multiple social worlds at any point in their lives. The combination of personal history and experience, intersecting social world memberships, and personal desires, needs and goals all combine to mean that people tend to have highly idiosyncratic views of the world. While this is well known, collaboration systems tend to take very little account of this, giving only a fixed group view. Thus, our first criticism of wOrlds is that it does not account for the idiosyncrasies of individuals, allowing each user to build his or her individual view of the system, its locales, and the contents of the locales themselves.

Further, while we are all members of different social worlds the *intensity* of that membership is again highly idiosyncratic. In one situation one may be intensely involved, and in another have only a passing interest or be interested only in a particular facet of the activities of members of the social world. Our second criticism of wOrlds, thus, that it does not allow for different individuals to participate in locales at differing degrees of intensity: one is either "in" a locale or one is not.

2.2.2. Spaces and places. Like all MUDs, wOrlds provides a collection of "virtual spaces" in which users "gather" to work together, utilizing whatever resources are available in the space. We believe this emphasis on space is a tactical error on the part of CSCW systems designers. While there are many situations in which modeling the physical in the virtual might have advantages, such as modeling meeting rooms, lectures or conferences, there are many times when trying to simulate the physical in the virtual might be positively debilitating.

The telephone is a fine example of this phenomenon. Telephones do not replace face-to-face communication or provide a perfect substitute for it, instead they provide an alternative means of communication, for which we have developed rich social protocols and which have become a critical part of modern society. But thinking of a telephone as a "space" seems ridiculous and there are many other affordances for communication and collaboration that are not spaces, such as mail, fax machines, mailing lists and radio stations and receivers.

We believe that a shift in focus from the purely spatial to a richer notion of *place* as the basis for supporting collaboration will result in greater power and flexibility in CSCW environments. Places, loosely defined, are collections of affordances with the potential for supporting interaction and communication. Thus, places are a superset of spaces, admitting a wider spectrum of possible bases for collaboration.

One immediate practical upshot of such a shift is that the "fundamental thing" we conceive of collaboration systems as providing becomes much more dynamic. In MUDs, for example, "everything is a room" in the end - the fundamental gathering place for collaboration is the MUD room. In wOrlds, it is the locale, and so forth. While graphical representations of rooms will continue to be useful for many situations, in a place-based system users will be able to use any combination of affordances and representations they see fit for the task at hand. Thus one group may choose to have intermittent AV conferences with a whiteboard while another may choose to use a bulletin board and a third could build an entire "situation room", all to deal with the same problem.

2.2.3. Boundaries, permeability and awareness. People employ a multitude of simultaneous foci in their work. wOrlds and other room-based systems provide rigid rooms with strong boundaries. This makes it very difficult to work on "several things at the same time" or keep one's eye on activities in one situation while concentrating on another.

The essential problem here is that while rooms (and locales) have boundaries, social worlds have *centers*. Rigid walls around rooms do not permit users to be sufficiently aware of the activities in other spaces. Our challenge, therefore, is to build systems that allow users to have several simultaneous, personalized foci of differing intensity that support the seamless awareness and permeability among work activities which characterize the real, workaday world.

2.2.4. History and trajectory. Finally, a major weakness with almost every collaboration system we know of is that they do not deal well (if at all) with the issue of providing context through history (what has already happened) and trajectory (what might be a good idea to do next).

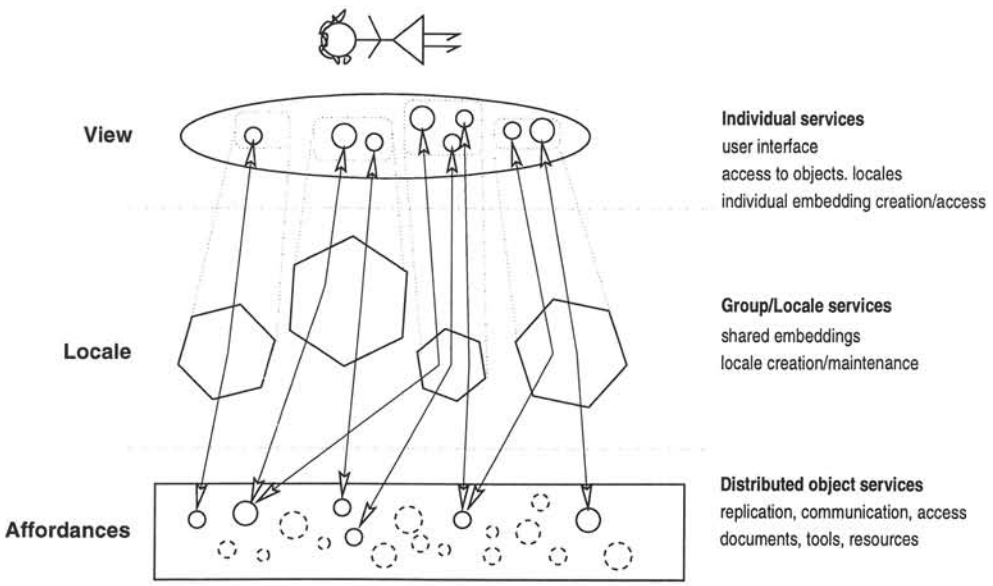

Figure 1. Towards an Orbit Conceptual Architecture

To a very limited extent workflow systems *do* provide this in the sense that one can see where one has been (the parts of the workflow that has been completed) and where one might go next (the uncompleted parts of the flow). Doug Bogia outlines a sophisticated and flexible collaboration model, called Obligations, which provides both for histories and for several different kinds of flexible modifications to workflows. However, no support is provided for capturing the cultural-level history and trajectory of work.

2.3. Systems implications: abstract view

We do not have, as yet, any clear idea of how to build systems that meet this goal, but the approach we believe should be taken is summarized in Figure 1. In our new system, Orbit, we plan to move from the current one-level architecture of wOrlds (objects appear in locales, locales are strictly segmented, users can be in only one locale at a time, and all users see the same view of the locale and its objects) to the more sophisticated three-level architecture sketched in Figure 1. The three levels may be summarized as follows:

2.3.1. Distributed object services. At the lowest level, Orbit should provide a collection of services of various kinds that can be used by workers to accomplish their tasks. Examples include shared, distributed objects, AV conferences, resource discovery agents, tools, etc. This level would also contain the infrastructural services, such as object distribution and replication, which form the technological basis for the system.

2.3.2. Group/locale services. In Orbit, locales are the places where social worlds can build shared collections of distributed objects that are germane to the task they are trying to perform, available network bandwidth, etc. A locale should be thought of as a shared "lens" which brings certain of the facilities at the lower level together. This level will also contain the services for editing and maintaining locale definitions and processes.

2.3.3. Individual services. Where the distributed services level allows the definition of particular services, objects, etc., and the group/locale level allows the definition of shared group views over subsets of the information at the affordance level, shared processes, etc., the individual services level allows users to define the ways in which they will view and interact with the many locales in which they participate. Unlike wOrlds, which imposed strict boundaries among locales, in Orbit the user will be able to hold information from multiple locales in focus simultaneously, with differing degrees of intensity, and switch easily from one to another.

Naturally the question of how exactly to build a system like Orbit remains open. We will report on our progress in future papers.

2.4. Systems implications: technological view

The issues raised here are those which have presented the wOrlds implementation team with *fundamental* diffi-

culties or obstacles, as opposed to problems which are, in one way or another, related to particular implementation choices.

2.4.1. Distributed objects. The ability seamlessly to manipulate local and distributed objects is fundamental to the construction of any complex CSCW system. We chose to investigate ORB technology as the basis for the distributed object model in wOrlds. The choice seemed sound - ORBS are built to a standard model, that support the interoperability of systems, languages and machines, and they are the vehicle of choice for many distributed systems researchers and developers. However, we have identified several significant shortcomings with ORB-based distributed object systems which greatly hinder our ability to build a truly scalable collaboration support environment.

Internet instability and ORB reliability. The Internet provides an extremely unstable networking framework, while ORBs are architected on the assumption that interaction among the various servers providing access to distributed objects is reliable and stable. This creates a fundamental mismatch that makes it extremely difficult to deploy ORB technology over anything other than local area networks (or dedicated wide-area networks with considerable bandwidth). Since the Internet is not likely to become any more stable in the near future, ORB services that support reliable access to objects is absolutely essential.

Replication. One way that the reliability-of-access issue can be mitigated is to replicate information in many sites, systems can then concentrate on accessing local data, or have several alternate paths to data, while the underlying replication service takes care of duplicating objects around the network as required. While this sounds very attractive, ORBs support no replication services, and indeed no general replication solutions exist. We believe that the solution to this problem lies not in developing the perfect replication algorithm as every situation could potentially require different replication strategies, but rather to build a replication framework and associated specification environment (rather like Introspect) which allows the development and evolution of many replication strategies.

2.4.2. Adaptability and quality of service. wOrlds should be capable of adapting to changing resource availabilities and different quality-of-service (QoS) demands. wOrlds is built almost exclusively from off-the-shelf components, which provide no facilities for automatically adapting to changing QoS demands, or adapting to changing resources. The entire area of adaptable interfaces which can deal with changes of these kinds remains open, but solutions are critically required in our domain. Obviously any solutions to these problems are going to be closely related to solutions to the problems outlined in Section 2.4.1.

3. Acknowledgments

This work has been supported by the National Science Foundation under grants CDA-9401124, CCR-9108931 and CCR-9007195, the Defense Advanced Research Projects Agency under grant F30603-94-C-0161, the US Army Corps of Engineers Research Laboratory, Fujitsu Open Systems Solutions Inc., Intel, Bull, Sun Microsystems, Hewlett-Packard, Digital Equipment and the Department of Computer Science of the University of Queensland. The work reported in this paper has also been funded in part by a Co-operative Research Centre Program through the Department of the Prime Minister and Cabinet of Australia. Geraldine Fitzpatrick has received additional support through an Australian Postgraduate Research Award and an Australian Telecommunications and Electronics Research Board scholarship.

The views and conclusions expressed in this document are those of the authors and should not be interpreted as representing the official policies, expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Approaches for Utilising 'Work Contexts' in CSCW Systems

John C. Grundy

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

John G. Hosking

Department of Computer Science
University of Auckland
Private Bag, Auckland
New Zealand
john@cs.auckland.ac.nz

Abstract

When working together using CSCW tools, people have a "work context" within which they perform their work. A work context includes the tools they are using to perform the work, the work artefacts being modified, the tools used to communicate with collaborators, and the other people they are collaborating with. We describe our recent work with trying to represent work context information in a workflow system. We then describe our current work which tries to determine work context information, rather than requiring it to be explicitly specified, by monitoring user activities. We describe how both approaches allow work context information to be communicated between collaborating users, allowing users to better coordinate their work and understand why others have done certain things or are doing certain things. We claim such facilities are necessary in order to enable the next generation of workflow-based CSCW systems to better support work coordination amoung multiple users.

1. Introduction

When people work collaboratively together, they either formally or informally have a notion of the context of their work, which we refer to as a "work context". This includes the artefacts they use or modify to perform work, the tools used to view or modify these artefacts, the tools and artefacts used to articulate work to others (i.e. describe what a user is doing and why), and the other people they are working with. Work contexts may be fixed i.e. only certain artefacts (or parts of artefacts) and tools are used or modified and only certain communication techniques are used to discuss work with particular collaborators. Examples include specific phases of a software process (e.g. modify this particular code fragment using this particular tool/methodology). At other times a person's work context may be very volatile, with a variety of artefacts, tools and other users being utilised. Examples include aspects of office automation systems, where a person or persons may (or may not) use a word processor, spreadsheet, Information System or pen and paper, and their associated artefacts, to carry out some business process (e.g. processing invoices).

The notion of a "work context" is implicit in many CSCW systems, while made more explicit in others. Many CSCW tools have implicit work contexts depending on what problem domain they are used in. For example, email assumes data to transmit and receivers of this data, collaborative Web browsers assume people use any WWW pages and discuss these [5], and text chats, collaborative editors and whiteboards may be used to discuss or edit specific artefacts, thus their work context depends on what is being discussed or modified at a particular times and the participants of the discussion. Many such tools allow participants to come and go, and allow the focus of discussion or work to change while the tool is in use.

One example of explicit definition or representation of work contexts is the work of Schmidt and Bannon on the articulation of work [11], and their idea of dividing cooperative work into doing the work itself and articulating (or describing) the work done. Most workflow systems provide some manner of codifying the context of work for each process in the workflow. This is either just the roles involved, as in Regatta [12] and TeamFLOW [13], or more explicit artefact, tool and role definitions as in Action Workflow [10] and in process-centred environments, such as SPADE [1] and Oz [2].

Some systems, such as wOrlds [3] and Orbit [9], allow work contexts to be more flexibly specified and represented, with these contexts implicitly defined by the artefacts, tools, actions and collaborators involved, rather than explicitly codified. wOrlds and Orbit make use of the notion of a "locale" [4], where people perform both informal and formal aspects of work, with locales being very dynamic in their artefact, tool and participant make-up. Key problems with the use of work contexts in CSCW systems identified in the wOrlds/Orbit work

include lack of flexibility in defining a context, forcing people to be in only one context at a time, and lack of providing context through history and trajectory (what was done/is to be done next) [9]. Most existing workflow and process-centred environments suffer from such problems.

This paper describes the use of the work context notion in Serendipity, a flexible process modelling/workflow/CSCW system we have been developing. In Serendipity's initial design and implementation [6, 7, 8], work contexts were explicitly codified using a workflow-style notation. This has the disadvantages of inflexibility and focus on formal context specification as outlined above. Here we describe "agents" which can more informally "determine" work contexts as people use artefacts, tools and communicate with others. Mechanisms to help support multiple work context awareness for users, and to allow users to better visualise interrelationships between the work contexts associated with different aspects of their cooperative work are also discussed. We claim that adding such capabilities to workflow and process-centred environments will better support work coordination in such systems.

2. Serendipity Overview

Serendipity is a process modelling, enactment and work planning environment, which also supports event handling, group communication, and group awareness facilities [6]. Serendipity's notations are designed to be high-level and graphical in nature, and its coordination and rule mechanisms are easily extended by users. The notation is based on Swenson's Visual Planning Language [12] but extends it to support artefact, tool and role modelling, and arbitrary event handling.

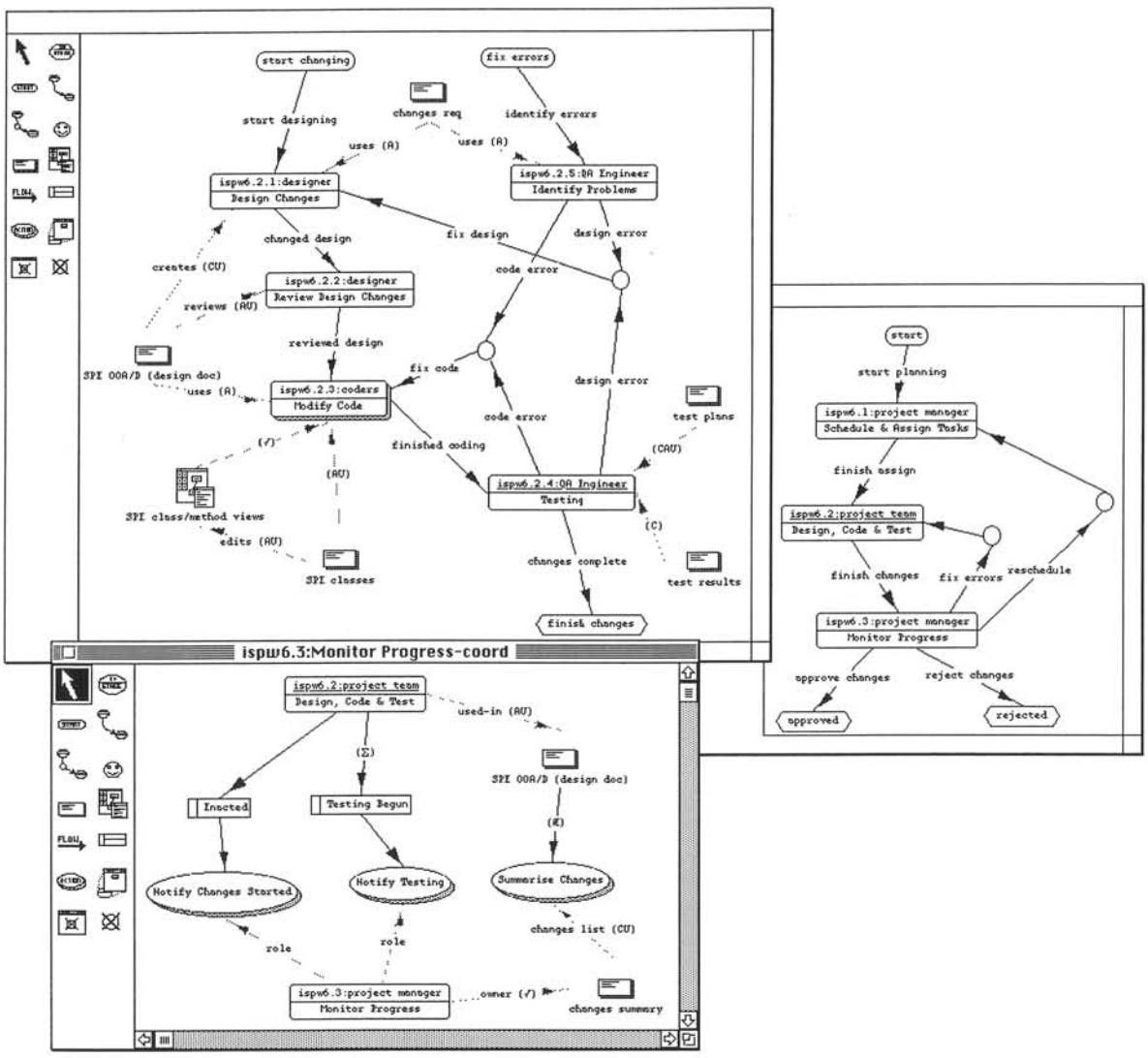

Figure 1. Part of the ISPW6 software process example modelled in Serendipity.

The left and bottom windows shown in Figure 1 are Serendipity views modelling part of the ISPW6 software process example. Stages describe steps in the process of modifying an arbitrary software system, with each stage containing a *unique id*, the *role* which will carry out the stage, and the *name* of the stage. Enactment *event flows* link stages. If labelled, the label is the *finishing state* of the stage the flow is from (e.g. "changed design"). There are a number of specialised types of stage including *start*, *finish*, *AND*, and *OR* stages (empty round circle).

Modularity is provided in the form of hierarchical subprocess models. The window at the left of Figure. 1 is a subprocess model refining the "ispw6.2:Design, Code & Test" stage of the process in the bottom window. Underlined stage IDs/roles mark the presence of a *subprocess* model. The shadowing of the "ispw6.2.3:Modify Code" stage indicates that multiple implementers can work on this stage (i.e. the stage has multiple subprocess enactments).

Serendipity supports artefact, tool and role modelling for processes. *Usage connections* show how stages, artefacts (eg the change requirements document "changes req"), tools (such as SPE's OOA/D class/method views) and roles (such as the project manager) are used. Optional annotations indicate: data is created (C), accessed (A), updated (U), or deleted (D); whether a stage must use only the tools, artefacts or roles defined (√); and whether a stage cannot use a particular tool, artefact or role (¬).

In addition to specifying the static usages and enactment event flows between process model stages, Serendipity supports *filters* (rectangular icons) and *actions* (ovals), which process arbitrary enactment and work artefact modification events.

For example, the coordination of the process model via the "ispw6.3:Monitor Progress" stage is defined by the top-right window of Figure. 1. This uses two filters and three actions to carry out the coordination. The Enacted filter selects only stage enactment events, in this case when the ispw6.2 stage is enacted.

This triggers the "Notify Changes Started action", which notifies its associated role (in this case, the project manager) that changes have commenced. The other filter acts similarly to notify commencement of testing. The other action takes artefact modification events from the OOA/D design document and accumulates them into a changes summary. Serendipity models may be used to guide work or to enforce particular work processes (by defining rules with filters and actions).

3. Defining and Determining Work Contexts

As can be seen from the left-hand side process model in Figure 1, Serendipity allows users to specify and hence codify work context information (artefacts, tools and collaborators) for processes. Users have the ability to restrict the access of a process to particular artefacts, tools and other users. They can also specify more loose work contexts ("this process may make use of so and so..."). This approach works well when codifying work contexts for use with formal process models. For example, we have found this useful for software process modelling and enactment and the coordination of work in such projects.

The approach doesn't work so well for the more volatile "processes" (work contexts) in such systems, nor in domains where such volatile work contexts are the norm (higher-level system design, office automation, etc). In these situatiuons, people cooperating do not usually wish to formally specify exactly what artefacts, tools, collaborators and communication mechanisms they want to use [11, 9]. Most other workflow and PCEs suffer from the same deficiency in this regard as Serendipity - in fact most do not even support the level of work context and work coordinatioon capabilities as outlined above [8].

Serendipity's filter/action visual language allows users to specify a variety of mechanisms for handling notification of changes and "automatic" processes. Currently this has been used in a similar style to the example in Figure 1: to help coordinate work or automate some parts of work processes. However, this language has the potential to support informal determination of work context information.

For example, Figure 2 shows a filter/action to determine the artefacts, tools and collaborators someone is working with for a particular process stage. Whenever the user accesses or modifies an artefact, a tool they are using generates an event, or they communicate with another user with CSCW tools, this information is cached in a simple database. Other filter/actions can be then be defined to extract this information and make use of it.

This may be to formalise the work context/process model the person has been using, or to visualise the dynamic aspects of their work context. This filter/action can be applied to all possible process stages by applying the filter/action to a metaprocess model or to the top-level process stage for a system. More complex filter/actions can be defined and utilised which recognise specific artefacts/tools/people of interest, collate events before storing them, etc.

Using such an approach means Serendipity can dynamically determine the contexts of peoples' work by monitoring their process stages and seeing which tools, artefacts and other people they use or interact with. By storing this information, it can be made available to users or to other filter/actions. The lack of such a capability in existing workflow/PCEs means they can not adapt to situations which require dynamic work contexts which can not be explicitly codified (or that users may not want to explicitly codify) in their workflow models.

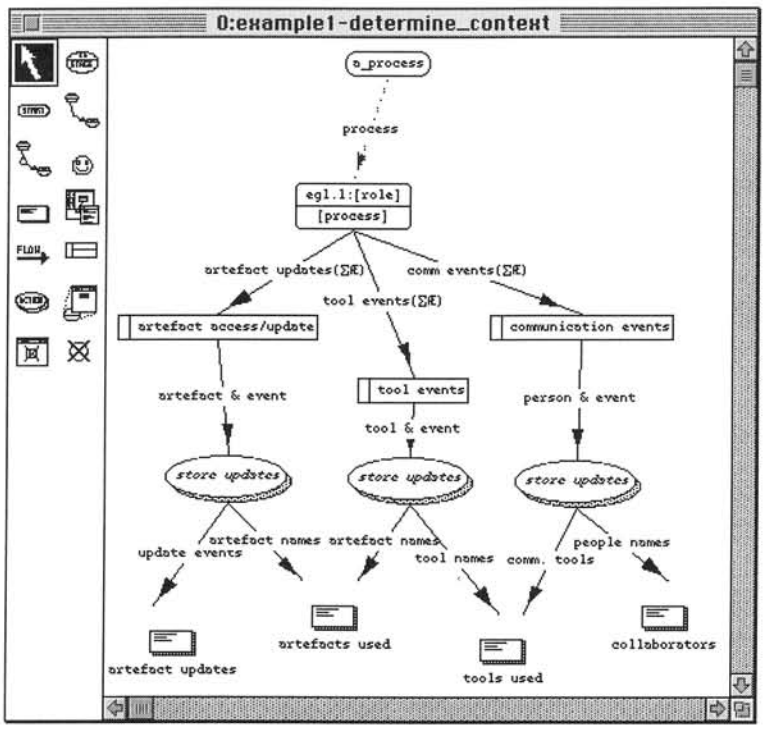

Figure 2. Determination of a work context.

4. Visualising Work Contexts

If we utilise work context determination techniques as outlined above, we also need mechanisms to access this information, visualise it, and use it to make others more aware of their collaborators' work and the reasons for it. Most workflow/PCEs which support multiple users rely on explicitly specified work context information to keep users aware of each other's work (if they even provide any work context awareness information at all).

Serendipity currently annotates events generated by tools integrated with Serendipity (i.e. tools for performing work) with the currently enacted process stage to indicate the work context [6, 7, 8]. This allows people to see not only the changes others have made to artefacts, but the "reason" these updates have been made (i.e. the work context they were made in). Serendipity also highlights the current enacted process stage of cooperating people, allowing a user to see just what aspect of a system other users are working on at the current time. Users can also define filter/actions which inform them of others' work (when they access/update artefacts, use tools etc.) in various ways (open a dialog box , highlight artefacts, send a message, etc.). Figure 3 (a) shows a filter/action to highlight the artefacts being being modified on a process model diagram when an artefact update event is received.

While we have found this approach to work context visualisation useful, it relies on collaborating users to explicitly enact and advance process model stages to keep others informed of the context of their work. Requiring people to explicitly enact or complete stages can interfere with the informal aspscts of cooperative work. In addition, the approach does not, currently, take into account that the artefacts, tools and collaborators for

someones work context may be dynamica, and thus need to be determined (as outlined in the previous section) rather than explicitly specified in process model views or in filter/action models.

We can make use of the "determined" work contexts discussed in the previous section by providing ways to visualise the stored event data, and to utilise this data to e.g. highlight any effected artefacts in tools being collaboratively used. For example, Figure 3 (b) shows a filter/action which highlights class icons in the SPE software development environment when another user modifies part of that class. We are working on other filter/actions which help keep collaborators aware of others' work using awareness widgets in tools which are updated by utilising stored, determined work context information.

Providing work context awareness facilities which utilise dynamically determined work context information allows Serendipity to be used in situations where such dynamic work contexts are utilised, while still providing similar context awareness capabilities to when work contexts are explicitly codified. Such an approach is not supported in most current workflow systems and PCEs.

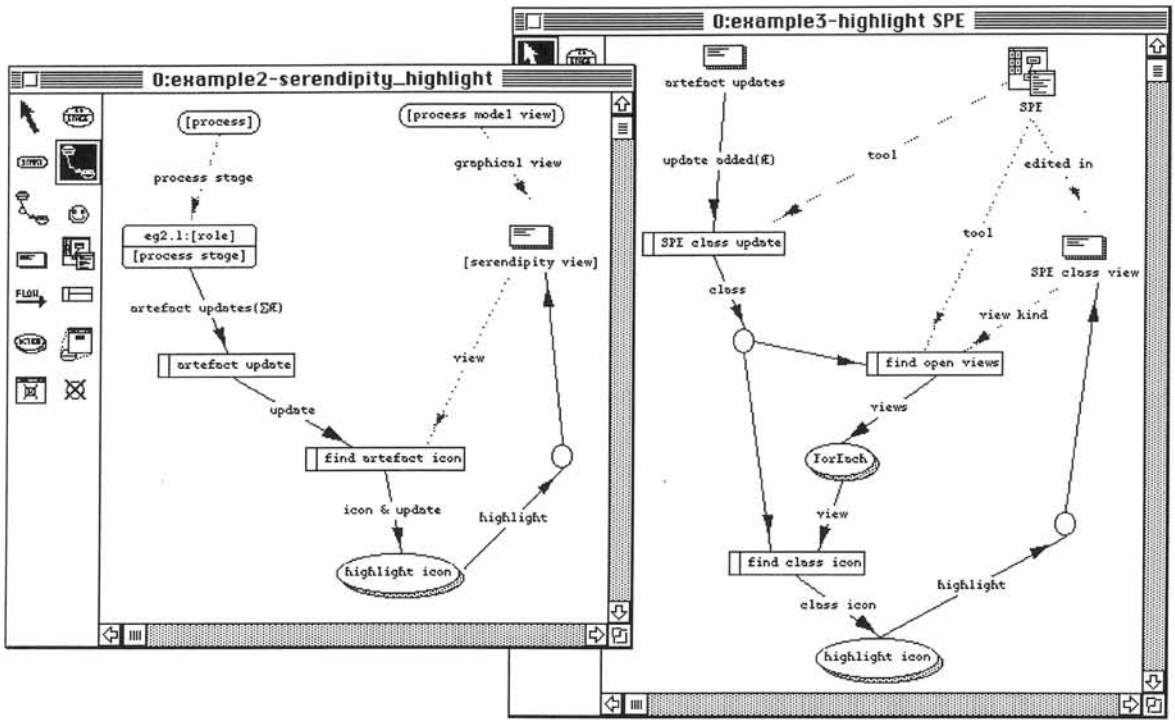

Figure 3. (a) Highlighting modified Serendipity artefacts. (b) Highlighting affected artefacts in tools.

5. Changing Work Contexts

Currently Serendipity assumes that either users explicitly enact/advance workflow/process model stages, stages are enacted by enactment events flowing into them from stages they are conencted to, or actions send process stages enactment events in response to some other filtered event(s) which have occured.

If work contexts are specified in process models, we can utilise determined work context information to "automatically" change the work context for a user. This can be achieved by a filter/action determining someone is now using a tool or artefact or communicating with another user, not specified for their current enacted process model stage. The filter/action can attempt to determine an appropriate alternative process stage and then enact that stage. Even if work contexts are not fully specified, if events caused by user actions utilise a tool, artefact or person not before seen when determining work context information, this could be recognised by filter/actions as some form of potential work context switch.

Alternatively, Serendipity could record all events that occur while the user is in a "non-determined" stage, and when the user subsequently enacts a stage or a suitable stage is determined, these events can be processed for the newly determined work context. We envisage that this "optimistic" work context determination could also be

used to predict future actions of the user, and hence make others aware of this, by using the history of actions and work contexts which have been stored.

Once again, the lack of such capabilities in current workflow and PCEs means users are constrained to using codified process models and work contexts. Thus the formal aspects of work are well-supported, but where processes and/or work contexts are more dynamic, these CSCW systems do not adequately support work in these domains. Providing filter/actions which allow people to work in "non-determined" stages but retain a history of work allows process models to be specified incrementally as users develop models while carrying out their work. Automatic context-switching allows users to do their work while Serendipity determines their work context from their historical work. The histories of work allow future events to be anticipated based on previous actions taken during a work context, or previous actions in relation to the artefacts, tools and people a person is currently working with.

6. Summary

We have discussed the current use of work contexts in the Serendipity workflow/process modelling environment. We are trying to utilise Serendipity's built-in event processing language to allow more flexible determination, visualisation and utilisation of work context information. This, we believe, will better support users and application domains where formal codification of work processes and work contexts is not appropriate, or where this codification unduly interferes with cooperative work. Adding such capabilities to the next generation of workflow and process-centred environments will, we believe, enhance the usefulness of these tools in such situations.

References

[1] Bandinelli, S., Fuggetta, A., Ghezzi, C., and Lavazza, L., *SPADE: an environment for software process analysis, design and enactment.* Finkelstein, A., Kramer, J. and Nuseibeh, B. Eds, J. Wiley, 1994.

[2] Ben-Shaul, I.Z. and Kaiser, G.E., "A Paradigm for Decentralized Process Modeling and its Realization in the Oz Environment," in *16th International Conference on Software Engineering,* May 1994, pp. 179-188.

[3] Bogia, D.P. and Kaplan, S.M., "Flexibility and Control for Dynamic Workflows in the wOrlds Environment," Procs *of the Conference on Organisational Computing Systems,* ACM Press, Milpitas, CA, November 1995.

[4] Fitzpatrick, G., Tolone, W.J., and Kaplan, S.M., "Work, Locales and Distributed Social Worlds," in *4th European Conference on Computer-Supported Cooperative Work,* Kluwer Academic Publishers, Stockholm, Sweden, 1995.

[5] Gianoutsos, S. and Grundy, J., "Collaborative work with the World Wide Web: adding CSCW support to a Web browser," Procs *of Oz-CSCW'96,* DSTC Technical Workshop Series, University of Queensland, Brisbane, Australia, August 1996, pp. 14-21.

[6] Grundy, J.C., Hosking, J.G., and Mugridge, W.B., "Low-level and high-level CSCW in the Serendipity process modelling environment," *Procs of OZCHI'96,* IEEE CS Press, Hamilton, New Zealand, Nov, 1996.

[7] Grundy, J.C., Venable, J.R., Hosking, J.G., and Mugridge, W.B., "Coordinating collaborative work in an integrated Information Systems engineering environment," *Procs of the 7th Workshop on the Next Generation of CASE tools,* Crete, 20-21 May 1996.

[8] Grundy, J.C. and Hosking, J.G. "Serendipity: integrated environment support for process modelling, enactment and improvement," Working Paper, Department of Computer Science, University of Waikato, 1996.

[9] Kaplan, S.M., Fitzpatrick, G., Mansfield, T., and Tolone, W.J., "Shooting into Orbit," Procs *of Oz-CSCW'96,* DSTC Technical Workshop Series, University of Queensland, Brisbane, Australia, August 1996, pp. 38-48.

[10] Medina-Mora, R., Winograd, T., Flores, R., and F., F., "The Action Workflow Approach to Workflow Management Technology," Procs *of CSCW'92,* ACM Press, 1992, pp. 281-288.

[11] Schmidt, K. and Bannon, L., "Taking CSCW seriously: Supporting Articulation Work," *Computer Supported Cooperative Work: An International Journal,* vol. 1, no. 1-2, Kluwer Academic Publishers, 7-40, 1992.

[12] Swenson, K.D., Maxwell, R.J., Matsumoto, T., Saghari, B., and Irwin, K., "A Business Process Environment Supporting Collaborative Planning," *Journal of Collaborative Computing*, vol. 1, no. 1.

[13] TeamWARE, Inc., *TeamWARE Flow*, 1996. (http://www.teamware.us.com/products/flow/)

Supporting Cooperative Work for Software Developers with Multiple View Process-Centred Environments

Chris Marlin†

marlin@cs.flinders.edu.au

†Department of Computer Science
Flinders University
Adelaide, South Australia
Australia

David Jacobs†‡

jacobs@cs.flinders.edu.au
djacobs@asc.corp.mot.com

‡Motorola Australia Software Centre
Second Avenue
Technology Park, South Australia
Australia

Abstract

A significant proportion of the risks associated with software development projects appear to be concerned with the interactions between the people involved in the projects. Thus, better coordination of the personnel involved in software development is likely to be a key component of successful approaches to addressing the problems of software development. One approach to the coordination of the activities of software development personnel is embodied in so-called process-centred software development environments. *Previous work on these environments has largely focussed on the technical issues concerned with the definition and enactment of software processes, rather than catering for the differing information needs of the users of these systems. This paper outlines ongoing work on a multiple view process-centred software development environment, covering both the views presented to the users and the means by which the environment is being implemented.*

1. Introduction

Although a human activity of considerable economic importance, software development has been found to be an activity which has a number of inherent risks. These risks take many forms, but a significant proportion of them seem to be concerned with the interactions between the people involved in a software development project. In fact, one can take the view that the software crisis that arose in the latter part of the 1960's, and remains with us today, arose at the point that the software needs of society could no longer be met through the efforts of a relatively small number of lone talented software artisans. As one manifestation of the current difficulties, it has been demonstrated by DeMarco and Lister [3] that the majority of software projects which fail do so due to project management difficulties.

Thus, better coordination of the personnel involved in software development is likely to be a key component of successful approaches to addressing the problems of software development, as attempts are made to improve the quality of the software produced, reduce the time taken to produce the software artefacts, and so on. One framework for thinking about and addressing the coordination of software development personnel has been an increasing emphasis on having software development projects follow a defined *software process*. This software process may, in fact, define how software development personnel performing a range of roles should cooperate to achieve the overall goal. A typical manifestation of a defined software process being followed in an organisation is the production of some kind of process documentation, which is then used by the relevant development teams as the definition of how to carry out various steps (such as module testing) or of the appropriate relationship between subsets of the project personnel (such as the relationship between the designers and the implementors).

Another manifestation of the software process approach has been the development of so-called *process-centred software development environments* (PCSDEs). A number of these environments have been designed and built [2, 4, 10, 11]. There are a number of ways in which such environments may operate, but generally their operation amounts to guiding a development team along a defined software process. As a result, there will then be some assurance that the defined process is being followed; however, this assurance will always be limited by the certitude with which the environment can know what is happening outside the scope of the environment (e.g., if a software engineer carrying out module testing reports that a module has been tested sufficiently, then the environment must record that testing of that module is complete).

A PCSDE will have a range of users. Typically, these will include those defining appropriate software processes (and refining them based on information collected as to their effectiveness), software engineers engaged in the projects, and project managers monitoring the progress of their projects.

Most of the past work on PCSDEs has focussed on technical issues concerned with the definition and enactment of software processes, with little attention being paid to the information needs of the various users. However, it is immediately clear that the various users of a PCSDE will have different information needs; for example, an individual software engineer working on a project will most usually require a list of tasks to be performed and a quality manager may well want a summary of the quality assurance procedures which have been followed since the beginning of the project.

Thus, what is required is a PCSDE which presents multiple views of the software process, both with respect to process definition and process enactment (i.e., as it is being used to guide software developers in a project). These views would present information about the software process in a way which was most convenient for the particular intended user and for their circumstances or role at the time.

The remainder of this paper briefly indicates the views of a software process which might be supported by a multiple view PCSDE, and then presents an overview of the implementation of such an environment. The paper concludes with a brief discussion of ongoing work on our environment.

2. Multiple Views of Software Process

Figure 1 shows three of the views proposed for our multiple view process-centred software development environment. A number of views are planned, as an indication of the ability of our implementation architecture to support a range of such views.

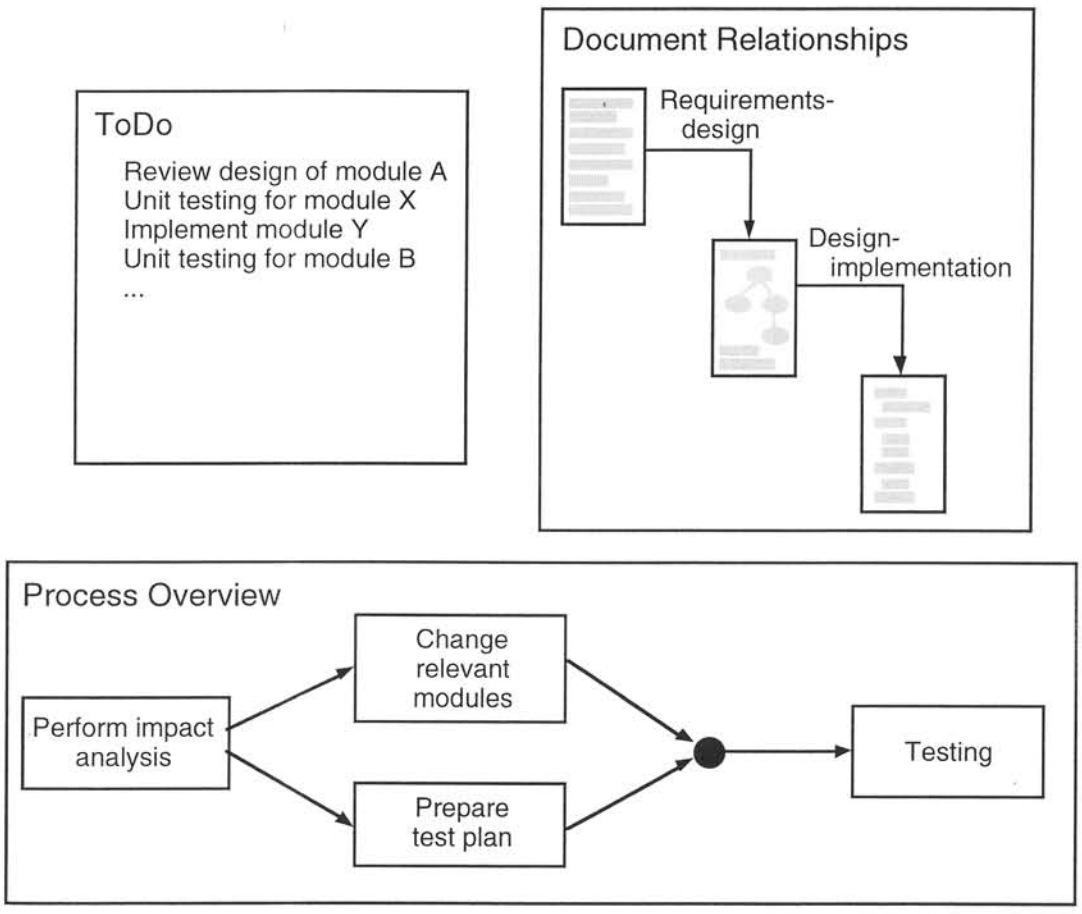

Figure 1. Sample proposed views of the software process.

The views in Figure 1 are:

- a ToDo list which lists all the jobs assigned to a particular person on the project team,
- a Document Relationships view which illustrates the relationships between various artefacts produced by the project, and
- a Process Overview view which shows a high level view of the software development process defined for the project.

In a multiple view PCSDE such as this, it is important that all the views are consistent. To ensure this consistency, it is most convenient if all the views are generated from a single representation of the software development process. In our PCSDE, a canonical representation of the software process [6] being enacted is stored in the process database. The following section outlines the essential details of the organisation of the environment and how the views are maintained.

3. Implementation

As mentioned in the previous section, the views of the software process in our environment are generated from a canonical representation of the software process. This organisation, which is described in more detail in [5], is a specific application of an architecture for supporting multiple views which was initially developed in the context of the MultiView programming environment [1, 7], an environment intended only to support the work of a single software developer. This architecture is described in general terms in [8].

When applied to the multiple view PCSDE, the architecture is as shown in Figure 2. The *database process* contains the canonical structured representation of the software process. Each *view process*, which will be notified by the database process of any changes to the process representation that are relevant to this view type, maintains a view-specific projection of those aspects of the process representation which are related to this view type. Note that the database process has no direct user interface, as all interaction with the users is expected to be carried out using view processes.

A view may be used to make some change to the process representation (e.g., a manager assigns a task to an individual software engineer); in this case, edits in the user interface of the view concerned are interpreted in terms of changes in the view-specific projection of aspects of the process representation and this view-specific projection is updated. These changes are then transmitted to the process engine, where they are used to update the software process representation residing there. Finally, the changes are broadcast to other view processes of types which relate to the parts of the process representation which have changed. The view processes need not be running on the same machine as the *process engine* (which executes the process description and thus guides the progress of the project along the defined path), as they communicate via internet sockets. This is important, as the PCSDE is intended to be used by teams of software engineers who need not necessarily work on the same machine or even in the same physical location.

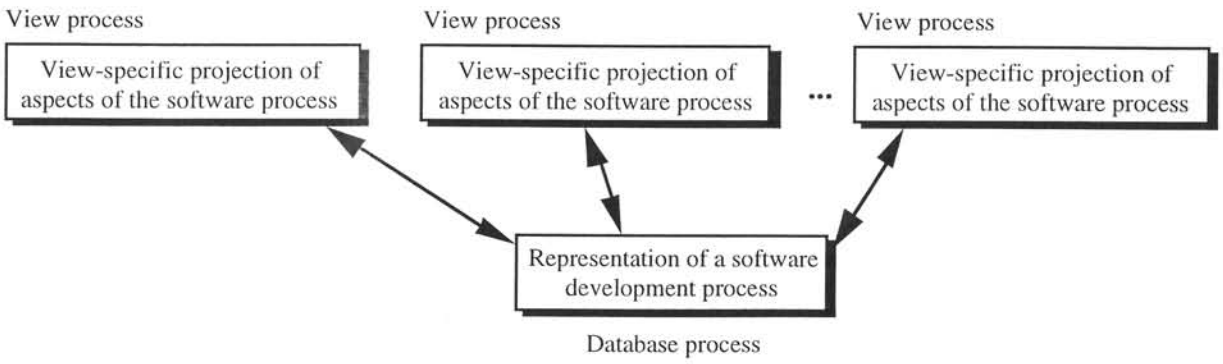

Figure 2. The MultiView architecture applied to a process-centred software development environment.

Each of the software development personnel cooperating on a particular project will have some collection of views on their workstation, implemented by a corresponding collection of view processes. The entire collection of view processes in existence communicates with the database process and the cooperation between individuals (say, in terms of a design document passing from the designer to the designated implementors) is achieved via the database process and its associated process engine.

4. Conclusions and Ongoing Work

This paper has presented an overview of a multiple view process-centred software development environment, which is intended to coordinate the work of groups of software developers working on software development projects. A key feature of this environment is the way in which the multiple views can be used to present information about the software development process in a manner which is tailored to the needs of the different classes of user of the environment.

Current work in this project centres on the implementation of the environment. As with the MultiView multiple view programming environment [1, 7], this implementation will use Ada and the X Window System. As a result, the environment will not only take advantage of the MultiView architecture and protocol compiler [9], but will also reuse many other special purpose Ada packages developed for the MultiView programming environment.

Once the implementation of the environment is complete, it will be evaluated through its use in software development projects following defined software processes. A particular focus of this test and evaluation phase will be to assess the utility of the initial collection of view types; the results of this evaluation will then provide input for the refinement of these view types and the incorporation of additional view types in later work.

Acknowledgments

The work described in this forms part of a long-term collaborative software engineering research programme involving the Department of Computer Science at Flinders University and the CSIRO-Macquarie University Joint Research Centre for Advanced Systems Engineering; funding for this latter work from the CSIRO Institute of Information Science and Engineering is gratefully acknowledged.

The assistance of the Motorola Australia Software Centre is also acknowledged, especially in terms of hosting the first author during his sabbatical leave from Flinders University in the second half of 1996.

References

[1] R. A. Altmann, A. N. Hawke and C. D. Marlin, "An integrated programming environment based on multiple concurrent views", *Australian Computer Journal,* Vol. 20, No. 2 (May 1988), pp.65-72.

[2] W. Deiters and V. Gruhn, "Managing software processes in the environment Melmac", in *Proc. 4th ACM SIGSOFT Symposium on Software Development Environments {SIGSOFT Software Engineering Notes, Vol. 15}*, R. N. Taylor (Ed.), pp.193-205 (ACM Press, New York, New York, 1990).

[3] T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams* (Dorsett House Publishing Company, New York, New York, 1987).

[4] A. Finkelstein, J. Kramer and B. Nuseibeh, "Viewpoints: a framework for integrating multiple perspectives in system development", *International Journal of Software Engineering and Knowledge Engineering,* Vol. 2, No. 1, pp.31-57.

[5] D. A. Jacobs and C. D. Marlin, "Multiple view support using the MultiView architecture", *Proc. International Workshop on Multiple Perspectives in Software Development (Viewpoints'96)* (San Francisco, California, 1996), pp.217-221.

[6] D. A. Jacobs and C. D. Marlin, "Software process representation to support multiple views", *Proc. First Asia-Pacific Software Engineering Conference* (Tokyo, Japan, 1994), pp.197-205.

[7] C. D. Marlin, "A distributed implementation of a multiple view integrated software development environment", *Proc. Fifth Conference on Knowledge-Based Software Assistant* (Syracuse, New York, 1990), pp.388-402.

[8] C. D. Marlin, "Multiple views based on unparsing canonical representations – the MultiView architecture", *Proc. International Workshop on Multiple Perspectives in Software Development (Viewpoints '96)* (San Francisco, California, 1996), pp.222-226.

[9] M. J. McCarthy and C. D. Marlin, "Interprocess communication protocol support in a distributed integrated software development environment", *Australian Computer Science Communications,* Vol. 16, No. 1 (Part B), pp.363-371.

[10] B. Peuschel, W. Schäfer and S. Wolf, "A consistency model for team cooperation", *Proc. 7th International Software Process Workshop* (1991), pp.114-116.

[11] The GOODSTEP Team, "The GOODSTEP project: General object-oriented database for software engineering processes", *Proc. First Asia-Pacific Software Engineering Conference (APSEC'94)* (Tokyo, Japan, 1994), pp.410-419.

A Panacea for Distributed CSCW Infrastructure?

Andrew Berry
Department of Computer Science
The University of Queensland
<andyb@dstc.edu.au>

1 Introduction

One of the major difficulties of building and deploying a distributed CSCW system is choosing or building an appropriate distributed infrastructure. While there are a number of solid infrastructures for building distributed information systems, most of these fall a long way short of providing the flexible and dynamic support required by CSCW systems.

Research and commercial development of distributed infrastructure has typically focused on support for building reliable information systems, for example CORBA, DCE and Isis/Horus[14]. These infrastructures provide programmer-oriented functionality in the form of remote procedure call or multicast and unicast messaging. These interaction mechanisms are relatively low-level and static, requiring significant additional programming to support the higher-level interaction mechanisms found in CSCW systems. The infrastructures also tend to be closed, in that all application components[1] must be written specifically for the distributed infrastructure being used.

Existing CSCW systems, for example wOrlds[5] or TeamRooms[6] have resorted to building their own distributed infrastructure services by extending existing services and building new infrastructure to suit their immediate needs. While these approaches have been effective, considerable developer effort has been expended in developing and configuring necessary infrastructure. Since these are prototypes whose primary goal is to demonstrate and support CSCW research, the resulting distributed infrastructure is often inflexible and can be difficult to deploy or extend.

My thesis is that the effort required to build, configure and deploy distributed infrastructure for CSCW systems can be significantly reduced by the use of an executable *architecture description language*. Such a language would support specification of the configuration of distributed application components and the interactions of these components in a manner independent of the underlying network and protocols. By providing an execution engine for the language, CSCW application developers need only provide a description of the configuration and interactions of their application components. The development of an architecture description language and underlying distributed infrastructure support is currently being pursued at the University of Queensland in conjunction with the CRC for Distributed Systems Technology (DSTC). This position statement outlines the plan and progress to date of this work.

2 Plan and Progress

2.1 An Architecture Description Language

Recent research in the software engineering discipline has found that in large software systems, the configuration and interactions of the components is as complex as the software components themselves. This aspect of a software system is traditionally called the *architecture* of the system and has been treated

[1] The term "components" in this paper refers to executable software components. The term "object" is avoided because it tends to be ambiguous across disciplines.

informally or semi-formally in the design process. Software developers typically sketch the architecture with boxes and lines, relying on the discussion around the sketch to define the semantics. When it comes to coding the software components, the architecture is usually embedded in the components, which makes them less flexible and more difficult to re-use. To address these problems, researchers have begun modelling the architecture of a software system in a rigorous and semantically well-defined manner. A comprehensive review of work in this area is provided by Shaw and Garlan in [13].

A number of approaches are being used to define software architecture, including graphical tools with well-defined semantics and architecture description languages (for software, as opposed to hardware architecture description languages). These approaches are intended to remove configuration and interaction code from components, both supporting greater re-use of components, and capturing the architecture of a solution for documentation and potential re-use.

In this work, we have chosen to implement an *executable* architecture description language as a basis for description of component configuration and interactions. By *executable*, we mean that it will be compiled or interpreted and run over a suitable distributed systems infrastructure. The key to the success of the proposed work is development of a practical, programmer-friendly language for describing the architecture of distributed CSCW systems and applications. This work is well underway, and is driven by two areas of research:

1. Work on software architecture and connection languages, in particular the Rapide system[9] and the work of the Software Engineering Institute at Carnegie Mellon University[1][12].
2. Ongoing work on distributed systems architecture at the DSTC, and in particular, the A1! Architecture Model[4] and the Reference Model of Open Distributed Processing (RM-ODP)[7], an ISO standard reference model for distributed systems architecture.

These aspects are largely complementary, with the first aimed at providing flexible support for describing software architectures and the second aimed at mapping such software architectures onto a distributed infrastructure. Our recent work has been the development of a semantic model for architecture description languages suitable for distributed infrastructure[10] based on the following fundamental constructs:

interface: software components have interfaces through which they interact with their environment (via bindings).

binding: a binding is an infrastructure-provided configuration of network connections and behaviour. A binding specification describes a configuration of components and their allowed or expected interactions. Components connect to a binding through their interfaces and must satisfy the expectations of the binding specification.

events: components participate in a binding (interact) by executing events at their interface. Events have parameters and direction (in or out). Events may be non-atomic, that is, they can be built from a configuration of lower-level events.

event relationships: event relationships specify the behaviour and interactions of a binding by describing the relationships between events occurring at component interfaces. An event relationship specification describes:

- the ordering of events, which is a partial order because the components are distributed and operate in parallel,
- the relationships of event parameters, with arbitrary functional relationships permitted between related events, and
- real-time constraints, for example, specification of the maximum allowable delay between the emission of an event and a corresponding reception of a related event.

Interactions such as remote procedure call and multicast are described as an event relationship. In our architecture description language, abstraction and encapsulation facilities will allow pre-defined descriptions of these and other common interactions.

The arbitrary functional relationships between parameters are extremely powerful, allowing the description and implementation of bindings that can, for example, connect DCE components to CORBA components. The potential for unbounded complexity resulting from arbitrary functional relationships is dealt with by allowing a binding implementation to restrict the available functions.

The semantic model also provides highly-flexible facilities for abstraction, allowing us to hide low-level, protocol-oriented behaviour and provide a concise, high-level view of (abstract) events that are relevant to the application or interaction being described. This could be used, for example, to describe video communication in a manner that is independent of its implementation, yet retain compatibility with a particular implementation.

Work is currently underway to use this semantic model as the basis of an architecture description language. This language will describe binding and interface behaviour using events and event relationships. In addition to the core elements of the semantic model described above, it is intended that the language provide encapsulation, implementation inheritance, and dynamic modification of behaviour through access to a meta-level. These facilities are intended to give a structured language with the flexibility necessary to describe CSCW applications.

2.2 Investigation and Description of CSCW Systems

Our work on architectural models for distributed systems has been in progress for several years. It has become clear that for the work to be useful, it must adequately support a wide variety of applications and interaction styles. To validate and refine the model and infrastructure, CSCW systems have been chosen as a target application area due to the inherent complexity of interactions and the need for flexible and dynamic infrastructure support.

In parallel with the architecture description language work, this author has begun to gather the of CSCW systems requirements through participation in an evaluation of wOrlds[5]. The following key requirements have been identified:

- It should be possible to incorporate existing application components and interaction mechanisms in a cohesive and flexible manner;
- The infrastructure should support dynamically configurable replication with a variety of mechanisms to deal with varying bandwidth, latency and coordination requirements in Internet-scale networks;
- It should be possible for users to dynamically vary their participation in and awareness of collaborative activities. This is necessary to support the ever-changing work focus of individuals and the varying quality of service provided by the network.

It is intended that other CSCW systems, literature and infrastructure toolkits, for example the relevant COMIC work[3] and GroupKit[11], will also be evaluated to develop a well-rounded set of requirements for distributed systems infrastructure. The author is also participating in the design of a successor to wOrlds known as Orbit[8], which will be the source of additional requirements.

As an evaluation of the language work described in 2.1, the interaction mechanisms provided or required by these systems will be specified in the resulting language. This will both ensure that the language is capable of describing such systems, and provide a basis for duplicating the infrastructure of those systems once the prototype execution environment has been completed.

2.3 Prototyping

The binding language described in section 2.1 requires an environment to support its execution. Given such an environment, a CSCW system developer could, for example, specify a suitable replication strategy in

a high-level language and have it implemented directly by the execution environment. This binding specification can be re-used in higher-level bindings that describe application-oriented features, for example, shared document editing.

A prototype distributed infrastructure supporting the principles of the A1! Architecture Model[4] has been developed at the DSTC. The Hector[2] prototype aims to provide support for connection between distributed components using arbitrary protocols based on a binding description. The focus at present is on the establishment of such connections in an open environment, including negotiation of binding specifications and parameters. In its current form, Hector provides only a relatively low-level API for software developers.

The goal of this author's work is that an interpreter for the binding language described in section 2.1 will execute within the Hector prototype. The researchers working on Hector are in close contact with the language work, ensuring that our respective goals and requirements remain synchronised. The result should be a language and environment for describing and implementing complex CSCW systems over a distributed infrastructure.

3 Discussion

The preceding sections outline the plan and progress of work intended to provide an infrastructure and language environment to support distributed CSCW applications. This work is dependent on the progress of the various pieces that make the puzzle, and leaves us with a number of interesting questions:

- Can a feasible distributed infrastructure adequately support the flexible and dynamic nature of CSCW applications?
- Is it possible to describe the necessary mechanisms of interaction using the semantic model that has been developed?
- Are architecture description languages acceptable to developers building CSCW applications and systems?

It is intended that these questions and others will be answered by the continuation of this work. A successful outcome will mean that many of the distributed infrastructure problems currently faced by CSCW researchers will cease to exist.

References

[1] R. Allen and D. Garlan. Formalizing architectural connection. In *Proceedings 16th International Conference on Software Engineering.* IEEE, May 1994.

[2] D. Arnold and A. Bond. An interaction glue for middleware. DSTC Internal Report, Mar. 1995.

[3] S. Benford and J. Mariani, editors. *Requirements and Metaphors of Shared Interaction, COMIC Deliverable 4.1.* Esprit Basic Research Action 6225, Oct. 1993.

[4] A. Berry and K. Raymond. The A1! architecture model. In *Open Distributed Processing: Experiences with distributed environments.* IFIP, Chapman and Hall, Feb. 1995.

[5] G. Fitzpatrick, W. J. Tolone, and S. M. Kaplan. Work, locales and distributed social wOrlds. In *Proc. of the 4th European Conference on CSCW.* Kluwer Academic Publishers, 1995.

[6] S. Greenberg, C. Gutwin, M. Roseman, and A. Cockburn. From awareness to TeamRooms, GroupWeb and TurboTurtle: Eight snapshots of recent work in the GroupLab project. Technical Report 95/580/32, Dept. of Computer Science, University of Calgary, Dec. 1995.

[7] 10746-1 10756-2 10746-3 Basic Reference Model for Open Distributed Processing.

[8] S. Kaplan, G. Fitzpatrick, T. Mansfield, and W. J. Tolone. MUDdling through. To appear in IEEE Proceedings HICSS'97, 1996.

[9] D. C. Luckham and J. Vera. An event based architecture definition language. *IEEE Transactions on Software Engineering*, Sept. 1995.

[10] A. Rakotonirainy, A. Berry, S. Crawley, and Z. Milosevic. Describing open distributed systems: A foundation. To appear in IEEE Proceedings HICSS'97, 1996.

[11] M. Roseman and S. Greenberg. GroupKit: a groupware toolkit for building real-time conferencing application. In *Proc. 4rd Int. Conf. on CSCW*. ACM Press, Nov. 1992.

[12] M. Shaw. Procedure calls are the assembly language of software interconnection: Connectors deserve first-class status. Technical Report CMU-CS-94-107, Software Engineering Institute, Carnegie Mellon Univerity, Jan. 1994.

[13] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an emerging discipline.* Prentice Hall, 1996.

[14] R. van Renesse, K. P. Birman, and S. Maffeis. Horus, a flexible group communication system. *Communications of the ACM*, Apr. 1996.

A CORBA Platform for Component Groupware

G. Henri ter Hofte, Hermen J. van der Lugt, Harm Bakker
Telematics Research Centre, the Netherlands
E-mail: {H.terHofte, H.vanderLugt, H.Bakker}@trc.nl

Abstract

The next generation of CSCW systems should be component-based. Such systems consist of independently developed groupware components (i.e. CSCW software components) which users can pick and mix to obtain a groupware application environment tailored to their needs. To support developers in achieving this, groupware platforms should be based on CORBA, an emerging industry standard for distributed object computing. In this paper, we describe the rationale for a platform for component groupware and the extensions CORBA requires for such a platform. We briefly describe our experiences with the design and implementation of a prototype of such a platform.

1. Problems with current groupware

Many cooperative work situations require comprehensive and flexible technological support. It ranges from support for synchronous collaboration to support for asynchronous collaboration [2, 8]. Moreover, a large variety of media types should be supported, varying from discrete media such as text and graphics, to continuous media such as audio and video.

Compared to development of single-user applications, development of groupware applications involves many additional technical issues from distributed systems development, such as replication, consistency, concurrency, and communication protocols.

Due to these extensive requirements and the complexity of groupware development, up to now, users often end up with support for very limited collaborative tasks and have to deal with a lack of integration of groupware applications [3], if more complex collaborations are to be supported.

2. Component Groupware

Groupware systems that consist of independently developed groupware components provide an important solution to these problems. These component-based systems can both provide the power of isolated tools and the virtues of integration and migration.

Component groupware enables users to pick their favourite groupware components and mix them into a suitable groupware application environment. This requires well-defined program interfaces between different groupware components, enabling *component interworking* (possibly between components of different vendors). Each groupware component typically consists of interworking component entities located at different interconnected computer systems. This requires well-defined groupware component protocols. Preferably, the interfaces and protocols are open and standardized.

Moreover, component groupware enables *component-by-component migration*. Users could start with a groupware application environment where existing single-user applications are combined with collaborative platform features to become groupware components that support primitive forms of collaboration (such as file sharing and user interface sharing). Then, the environment can migrate, component-by-component, towards a groupware application environment that consists of groupware components that support more media types, various modes of collaboration, and transitions between these modes.

3. Platform

In order to realize the full potential of component groupware, we need a platform to support development of groupware components based on a high-level groupware software architecture.

The high-level groupware software architecture should distinguish different classes of groupware components, their combinatory possibilities and their interworking. We propose, inspired by the Co^4 model of groupware functionality [10], an architecture which distinguishes four classes of groupware components:

- conference management components;
- shared workspace components;
- conversation channel components;
- coordination support components.

A *platform for component groupware* allows developers to abstract from generic issues, by providing reuse of generic functionality that is embodied in the platform. It allows developers to concentrate on the important details of their particular groupware component. The platform should support construction and interworking of the identified component classes. Ideally, such a platform should be based on standards for distributed systems and groupware.

4. Using CORBA to Support Groupware

The Common Object Request Broker Architecture (CORBA) is one of the most widely supported emerging standards for Distributed Object Computing [15]. It can be of great value to a platform for component groupware:

- CORBA supports basic interworking between objects implemented in different programming languages. For this purpose, developers specify object interfaces in the standardized CORBA Interface Definition Language (IDL). Method invocations between objects are mediated by an Object Request Broker (ORB). CORBA allows for the construction of software components from (a number of) objects and facilitates interworking of such components.
- CORBA also supports interworking between objects across network boundaries by providing transparent invocation of methods on remote objects, while hiding, if necessary, low-level details such as communication protocols, transport encodings, concurrency, and bridging heterogeneity in operating systems and programming languages.

However, CORBA (like other contemporary platforms for Distributed Object Computing) lacks adequate support for groupware issues. Groupware platforms, such as GroupKit, Rendezvous, MEAD, IBM Lakes and Prospero (for overviews, we refer to [7, 20]), on the other hand, do address these issues, but rarely provide support for component software and are rarely based on standards.

Recent research on CORBA-based groupware [4, 12] and group communication-based groupware [18, 19, 9], and research into combining CORBA and group communication [13] indicates that a combination of CORBA, group communication and groupware is a very promising approach for platforms for component groupware.

5. Extending CORBA for Groupware

In the Platinum[1] project [16] we designed and built a prototype of a CORBA-based platform for component groupware. Based on our experiences, we propose that CORBA should be extended with the concept Object Group (§5.1), that ORB support for object groups should be provided (§5.2) and that CORBA facilities should be created for conference management components (§5.3), shared workspace components (§5.4), conversation space components (§5.5) and coordination support components (the latter is left for further study).

---

[1] A joint research project on multimedia groupware over broadband networks (http://www.trc.nl/projects/platinum.htm), in which Lucent Technologies, the Telematics Research Centre, the University of Twente and Deutsche Telekom participated.

5.1 Object Groups in Groupware

Typical patterns of interaction occurring in groupware systems are easily implemented with object groups [11], a concept originating from fault tolerant computing. An Object Group consists of a number of replicas: the member objects. To the world outside the group, an object group behaves as if it were a single object. A method invocation on an object group is invoked on all replicas. This ensures state changes are applied to all object replicas. Thus, a high availability of an object's state can be achieved, even if some members are not available due to faults such as system or network crashes.

A groupware system can be modelled as a distributed computer system with multiple user interfaces, each providing representation and manipulation of some shared artefact, be it a shared document, a textual discussion, or a workflow. Each user interface reflects the actions of its user. Due to different actions by different users, the interfaces *diverge*, and sooner or later, the system will *synchronize* the user interfaces so they not only reflect a user's own actions (feedback), but also the actions of other users (feedthrough).

A typical pattern of interaction in such a groupware system, as illustrated in Figure 1, is an invocation, triggered by a user action, that needs to be invoked "simultaneously" on a number of objects, each containing a replica of the shared artefact located at a users' site, in order to change the state of the shared artefact globally. Here, an object group can be used for the objects containing the shared artefact.

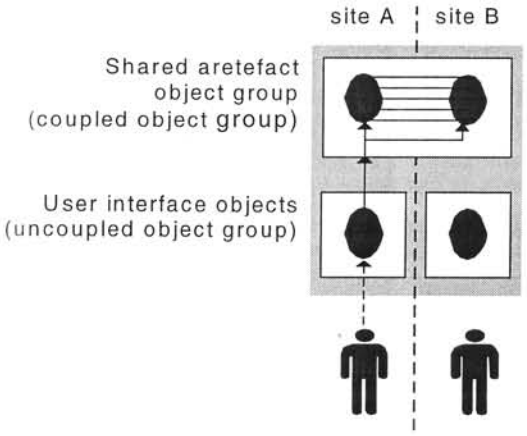

Figure 1 Typical use of Object Groups

5.2 Multicast Object Request Broker

Platform support for such Object Groups should be provided by a Multicast Object Request Broker (MORB). It should provide generic support for multicasting object invocations, multipoint ordering services, response collation, object group membership management and state transfer for newcomers. In addition to these services from the fault-tolerant domain, it should support services such as coupling and uncoupling object groups (see also §5.4), parallel versions support (intentionally diverged replicas) and version merging support (synchronizing diverged replicas). MORBs could evolve from current ORBs by replacing the transport layer with one that supports multicast RPCs, as done in Orbix+ISIS and Electra [13], or by connecting two ORBs with an inter-ORB multicasting bridge, an approach we followed in the Platinum project.

5.3 Conference Management Facility

Conferences consist of a number of participants (users), media (shared workspaces and conversation spaces) and coordination policies (such as a floor control policy). Conference Management functions, which manage associations between a number of users, shared workspaces, conversation spaces and coordination policies, are the multi-user CSCW equivalent of single-user desktop and window management functions, which manage associations between one user, applications and files. Conference Management functions include functions such as create, terminate, join, leave, add medium, delete medium, add coordination policy, delete coordination

policy. More advanced functions are conference suspend and resume, conference split and merge and creation of conference hierarchies in which a conference can contain subconferences.

Conference Management functions should be provided to users in a component separate from shared workspace and conversation space components, allowing reuse and uniformity of conference management functions for a wide array of conferences. However, it is unlikely that one conference management component can be used for all styles of conferences, ranging from long-lasting conferences with open participation and many participants, to short-lived conferences with closed participation and few participants.

A Conference Management facility should provide reuse of generic functionality of conference management components and should ensure interworking between conference management components and other components in a conference, including other conference management components of subconferences.

5.4 Collaborative Compound Document Facility

Collaborative Compound Document Editing represents an important enhancement over compound document editing. The latter was recently standardized in CORBA's Distributed Document Component Facility (DDCF) [1], which is based on OpenDoc [15].

In compound document editing, different parts of a document, such as picture, spreadsheet and text, are edited in-place, each by its own component editor, which together act as a seamlessly integrated editor. Users can pick and mix these different components to create the editing environment suitable for their needs. New content types can easily be added to documents by adding their component editor to the environment.

Whereas compound document editing already supports step-wise migration towards editing new media types, *Collaborative* Compound Document Editing in addition enables a step-wise migration from existing editing components (e.g., OLE applications) with limited collaborative facilities (e.g., only file and interface sharing), towards more advanced collaborative component editors.

More advanced collaborative component editors typically require multiple object groups which do not always need to be "coupled". If a level is coupled, state changes in objects at that level occur "simultaneously" for all users; if a level is not coupled, state changes in objects at that level occur in isolation, i.e. states of each member of an object group are unrelated. To enable users to selectively suppress feedthrough of actions with less profound impact, such as moving a mouse, a number of levels can be distinguished. We use four levels (from high to low: file, edit, view and user interface). With these levels, users can decide to "couple" all object groups at a certain level and all higher levels and to "uncouple" all lower levels (this is also known as the "zipper architecture" [6, 17]). Thus, users can be allowed control —per document part— over the "tightness" of collaboration support simply by coupling and uncoupling object groups.

Developing collaborative component editors requires platform support in the form of a Collaborative Compound Document Facility, which extends the existing DDCF with object groups and user controllable coupling in line with the zipper architecture. The facility should also provide interworking with other kinds of groupware components, such as conference management.

5.5 Multiflow Multipoint Channel Facility

CORBA currently primarily supports a request-response style of interaction, which is not suitable for support of continuous media flows with isochronous characteristics such as audio and video. In groupware, the continuous media streams typically have a multipoint character. Recently, OMG issued a Request For Proposal (RFP) on "Control and Management of A/V Streams" [14], seeking interface specifications for the control (i.e. setup, QoS (re)negotiation) of multipoint (i.e. with multiple sources and sinks) streams, consisting of multiple flows. We contend that such generic support should be provided by a multiflow multipoint channel facility, which provides interworking with other groupware components, such as conference management. The ReTINA approach [5] is an interesting starting point in this respect.

6. Implementation

During the Platinum project, we implemented a multicast ORB according to the inter-ORB multicasting bridge approach, based on IBM's SOM ORB and the SOM Metaclass Framework, which provides (non-CORBA-compliant) hooks for changing a class' method invocation mechanisms. For conference management, we implemented a (non-CORBA-compliant) conference management application.

On top of the MORB, we implemented CoCoDoc, a prototype of a framework for collaborative compound document editing, which supports four document-wide levels of coupling (file, edit, view, user interface) and allows collaborative part editors to offer additional coupling levels to users. CoCoDoc is implemented as an extension of OpenDoc.

Our implementation of CoCoTree, a simple collaborative compound outline editor on top of the CoCoDoc platform, confirmed that using CoCoDoc requires only minimal learning effort from OpenDoc programmers.

7. Conclusion and Open Issues

The next generation of CSCW systems needs to be component based. This requires a CORBA-based object oriented groupware platform, which supports Object Groups with a Multicast Object Request Broker (MORB), and provides CORBAfacilities for Conference Management, Collaborative Compound Documents (CoCoDoc) and Multiflow Multipoint Channels as important extensions to CORBA.

Further research is needed by the CSCW and CORBA community to standardize these extensions. Some important open issues we intend to address in future research are:

- What kind of platform support should be provided for coordination components?
- How should existing and emerging groupware protocol standards be used, such as the ITU-T T.120 multipoint data conferencing standards?

Acknowledgements

The authors wish to thank all Platinum project participants, in particular, Raymond Otte, Ferial Moelaert, Wouter Teeuw and Maurice Houtsma.

References

[1] Apple Computer, Component Integration Laboratories, IBM and Novell, *OMG RFP submission : Compound presentation and compound interchange facilities*. Object Management Group, Framingham, MA, USA, 1995, http://www.omg.org/docs/1995/95-12-30.ps. OMG document, 95-12-30.

[2] Beck, E.E. and V.M.E. Belotti, 'Informed opportunism as strategy : Supporting coordination in distributed collaborative writing'. In G. de Michelis, C. Simone and K. Schmidt (eds.), *Proceedings of the third European conference on computer-supported cooperative work*. Kluwer Academic, Dordrecht, NL, 1993, p. 233-248.

[3] Bullen, C.V. and J.L. Bennett, 'Groupware in practice : An interpretation of work experiences'. In R.M. Baecker (ed.), *Readings in groupware and computer-supported cooperative work : Assisting human-human collaboration*. Morgan Kaufmann, San Mateo, CA, USA, 1993, p. 69-84.

[4] Costa, F.M. and E.R.M. Madeirea, 'An object model and its implementation to support cooperative applications on CORBA'. In A. Schill, C. Mittasch, O. Spaniol, et al. (eds.), *Distributed Platforms : Proceedings of IFIP/IEEE international conference on distributed platforms*. Chapman & Hall, London, UK, 1996, p. 213-228.

[5] Dang Tran, F., V. Perebaskine, J.N. Stefani, B. Crawford, A. Kramer and D. Otway, 'Binding and streams: The ReTINA approach'. In *The convergence of telecommunications and distributed computing technologies : Proceedings of TINA'96 conference in Heidelberg, Germany, September 3-5, 1996*. VDE, German Association of Electrical Engineers, Berlin, D, 1996, p. 101-113.

[6] Dewan, P. 'Multiuser architectures' In C. Unger and L.J. Bass (eds.), *Engineering for HCI (Proceedings of the IFIP WG2.7 Working Conference on Engineering for Human-Computer Interaction in Grand Targhee Resort, Wyoming, USA, August 14-18, 1995*. Chapman & Hall, London, UK, 1996, p. 247-270, ftp://ftp.cs.unc.edu/pub/users/dewan/papers/arch.ps.Z

[7] Dourish, P., *Open implementation and flexibility in CSCW toolkits*. Ph.D. Thesis, University College London, 1996, ftp://cs.ucl.ac.uk/darpa/jpd/dourish-thesis.ps.gz.

[8] Dourish, P. and V.M.E. Belotti, 'Awareness and coordination in shared workspaces'. In J. Turner and R.E. Kraut (eds.), *CSCW'92 : Proceedings of the conference on computer-supported cooperative work, Oct. 31 to Nov. 4 1992, Toronto, Canada*. ACM, NY, 1992, p. 107-114.

[9] Farooqui, K., *Group communication models*, in press [to be published in Computer Communications Journal, 1997; also available via author's web homepage], http://www.csi.uottawa.ca/~farooqui.

[10] ter Hofte, G.H., H.J. van der Lugt and M.A.W. Houtsma, *Co[4], a comprehensive model for groupware functionality*, 1996. TRC Scientific Series, SS/96002 [to be published in Proceedings of the APTEC conference of Euromedia'96, held December 19-21, 1996 in London, UK. The Society for Computer Simulation, in press].

[11] ISIS Distributed Systems, Inc. *Object groups : A response to the ORB 2.0 RFI*. Object Management Group, Framingham, MA, USA, 1993, OMG document, 93-04-01, http://www.omg.org/docs/1993/93-04-11.ps.

[12] von Lukas, U. and U. Dietrich, 'CSCW in einer CORBA-basierten CA-umgebung' [CSCW in a CORBA-based CA environment (in German)]. In H. Krcmar, H. Lewe and G. Schwabe (eds.), *Herausforderung Telekooperation : Einsatzererfahrungen und Lösungsansätze für ökonomische und ökologische, technische und soziale Fragen undserer Gesellschaft : Proceedings of Deutsche Computer Supported Cooperative Work 1996, DCSCW'96 in Stuttgatt-Hohenheim, D, September 30-October 2, 1996*. Springer-Verlag, Berlin, D, 1996, p. 225-242.

[13] Maffeis, S., 'Adding group communication and fault-tolerance to CORBA'. In *Proceedings of USENIX conference on object-oriented technologies (COOTS) in Monterey, CA, USA, June 26-29, 1995*. USENIX association, Berkely, CA, USA, 1995, p. 136-145, ftp://ftp.cs.cornell.edu/pub/maffeis/electra/electra_corba.ps.gz.

[14] Object Management Group. *Control and management of A/V streams : RFP*. Object Management Group, Framingham, USA, 1996, OMG doc. telecom/96-08-01, http://www.omg.org/docs/telecom/96-08-01.ps.

[15] Orfali, R., D. Harkey and J. Edwards, *The essential distributed objects survival guide*. Wiley, NY, 1996.

[16] Ouibrahim, H. and J. Schot, 'Tele-teaching and the electronic superhighway : Towards a vertical approach'. In *Proceedings of IDC'95 : First international distributed conference on high-perforance networking for teleteaching in Madeira, Portugal; Madrid, Spain; Sophia Antipolis, France; Brussels, Belgium, November 16-17, 1995*, 1995.

[17] Patterson, J.F., 'A taxonomy of architectures for synchronous groupware applications'. *SIGOIS Bulletin*, 15 (April 1995), 3, p. 27-29.

[18] Powell, D., 'Group communication (special issue)'. *Communications of the ACM*, 29 (1996), 4, p. 50-97.

[19] Trevor, J.J., *Infrastructure support for CSCW*. Ph.D. Thesis, Lancaster University, UK, 1994, ftp://ftp.comp.lancs.ac.uk/pub/reports/ThesisGS.ps.Z.

[20] Urnes, T. and R. Nejabi, *Tools for implementing groupware : Survey and evaluation*. Department of Computer Science, York University, North York, Ontario, Canada, 1994, Technical Report No. CS-94-03, ftp://ftp.cs.yorku.ca/pub/clock-papers/CS-94-03.ps.Z.

Collaborative Filtering via PICS: A Case Study in Categorising and Evaluating Asia Information

Mimi M. Recker
Tim Beal
Victoria University of Wellington
P.O. Box 600
Wellington, New Zealand

Abstract

It is a well-recognised problem that the explosive increase in the number of World-Wide Web resources has seen a corresponding growth in the problem of finding relevant, quality, and validated information. Within information systems, there have been several promising approaches to the problem of labelling, categorising, and filtering information. In this research, we are exploring and evaluating the use of the Platform for Internet Content Selection (PICS), developed by the international W3 consortium, as an easily-integrated infrastructure for social information filtering of Web resources within a particular community of use (Asia-Pacific Law and Business). In particular, we are developing a Web-accessible database that will store third-party ratings of documents. Users can use our database in conjunction with a PICS-compliant browser to access documents and filter based on their ratings and/or categories. This research will lead to the design and evaluation of a PICS-based system for supporting the integration of user and community knowledge into documents, thereby creating a richer repository of Web information for our particular domain.

Introduction

It is a well-recognised problem that the explosive increase in the number of World-Wide Web [2] resources has seen a corresponding growth in the problem of finding relevant, quality, and validated information. The Web lacks the structure and strong typing found in more closed database system [6]. Moreover, its distributed nature precludes the implementation of filtering and reviewing conventions typically provided by libraries and publishers.

Within information systems, there have been several promising approaches to the problem of labelling, categorising, and filtering information. Malone et. al [4] describe three types of information filtering activities: cognitive, economic, and social.

Cognitive filtering is based on content. Content-based filtering depends on a machine-readable and parseable format. Unfortunately, this can be difficult to implement in a multimedia environment.

Economic filtering is based on a cost-benefit analysis of searching activities. While a powerful approach in large information repositories, it generally prevents serendipitous discovery of information.

Social information filtering is based on word-of-mouth and recommendations from trusted sources [7]. If you wanted to try a new restaurant, how would you decide where to go? You would probably ask friends or look at a restaurant guide. This same premise applies to social information filtering and, as proposed by others [3] holds promise for the Web.

Approach

For Web applications, a system supporting social filtering has several important pre-requisites [5]. First, the system must easily integrate into the existing infrastructure. Second, the system must make it easy for users to add ratings of Web documents. Third, a critical mass of users must participate to ensure rating reliability. Fourth, it must be easy for information seekers to see the ratings from other users. Moreover, the ratings must be presented in an easy-to-understand format. Finally, the system must support the creation of user profiles. These profiles, summarising user interests, can be used to form clusters of users based on shared information goals. This will enable documents of mutual interest to be spread among user clusters [3].

The Platform for Internet Content Selection (PICS), developed by the international W3 consortium, offers an easily-integrated infrastructure for information filtering. Originally developed in response to the threat by government groups of censorship of objectionable content on the Web, PICS is a generic specification imposing no restrictions on the kinds of labels and filters applied to Web documents [1].

In this research, we are exploring and evaluating the use of PICS as a vehicle for rating, categorising, and filtering Web resources within a particular community of users. Specifically, we are focusing on Web information related to Asia-Pacific Law and Business. This domain and its members are well-known to the second author.

In addition, our categorising scheme is based on several existing sources. We utilise the two series of keyword indexes (subject and geographical) developed by Tim Beal for the NZASIA Directory of Asian Studies (`http://www.vuw.ac.nz/~caplab/directy.htm`). The categorisation scheme is complemented by bibliographic research being done on Asia information on the Web, in particular the Asian Studies WWW Virtual Library developed by Dr T Matthew Coliek at ANU (`http://coombs.anu.edu.au/WWWVL-AsianStudies.html`) and the survey of Asian databases being conducted by Susan McDougall and George Miller of the Asia and Pacific Special Interest Group of the Australian Library and Information Association (`http://www.alia.org.au/~apsig/questionnaire.html`).

Research Activities

While our research effort is still in its infancy, our plan is as follows.

First, we are developing a PICS-compatible categorisation scheme for our domain. As previously discussed, the scheme is based on previously-developed indexes.

Second, we are developing a simple, easy-to-use tool that enables the insertion of classifiers into Web documents. As specified by PICS, the classifiers are inserted into the HEAD of HTML documents, within the META tag. The tool is used by a researcher to classify a small set of pre-selected documents. Other users in the community will be invited to use the tool to classify their own documents. Once completed, this exercise will enable a refinement of our classification scheme.

Third, we are developing a Web-accessible database that will store third-party ratings of documents. Using a Web client, interested users in the community can provide their ratings of documents (as specified by the URL) that they encounter on the Web. These ratings, of course, will use our classification scheme, and will be stored on our master database.

Users can then use our database in conjunction with a PICS-compliant browser (such as Microsoft Internet Explorer 3.0) to access documents and filter these based on their ratings and/or categories. If our database has several relevant ratings, it can present a composite picture of the document. Rating information can also be augmented with contextual information, such as title of the document, author of the rating, and usage history. As appropriate, we can use other meta-information to sharpen document context [6]. Research has shown that incorporating such contextual information is critical in filtering systems [5].

Finally, we plan to evaluate both the effectiveness of our tools and the PICS-based collaborative filtering approach. Our evaluation will be performed via studies of users, both by observing usage and by soliciting feedback.

This work serves to evaluate the effectiveness of PICS as a social information filtering system within a special-purpose community. At the same time, our research can lead to systems for supporting the integration of user and community knowledge into documents, thereby creating a richer repository of Web information for our particular domain.

Acknowledgments

We are grateful to the Internal Grants Committee, Victoria University of Wellington, for their support.

References

[1] K. Amaral. Promoting the PICS fix. *World-Wide Web Journal*, 1(3), Summer 1996.

[2] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Nielsen, and A.Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76--82, August 1994.

[3] W. Hill, L.Stead, M. Rosenstein, and G.Furnas. Recommending and evaluating choices in a virtual community of use. In *ACM Conference on Human Factors in Computing Systems*, New York, 1995. ACM.

[4] T. Malone, K. Grant, F. Turbak, S. Brobst, and M. Cohen. Intelligent information sharing systems. *Communications of the ACM*, 30(5), 1987.

[5] D. Maltz and K. Ehrlich. Pointing the way: Active collaborative filtering. In *ACM Conference on Human Factors in Computing Systems*, New York, 1995. ACM.

[6] P. Pirolli, P. Pitkow, and R. Rao. Silk from a sow's ear: extracting usable structures from the web. In *ACM Conference on Human Factors in Computing Systems,* New York, 1996. ACM.

[7] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word-of-mouth. In *ACM Conference on Human Factors in Computing Systems*, New York, 1995. ACM.

Qualitative Evaluation of a Collaborative Web Browser

Simon Gianoutsos and John Grundy
Department of Computer Science, University of Waikato
Private Bag 3105, Hamilton, New Zealand

Abstract

Many tools have been developed to support cooperative work, but many of these have not been evaluated. This paper describes various experiments conducted to derive a qualitative evaluation of a WWW browser with CSCW support. These experiments are not restricted to this particular domain but can also be applied to other types of CSCW systems.

1. Introduction

With the growing number of tools to support cooperative work, evaluation of these tools is required to increase our knowledge of user requirements. Many of the tools created react differently and offer differing levels of usefulness under certain circumstances, and modifications may be necessary to make a tool more generic. But how generic can a tool be before it is too basic to be useful? Is it useful for interactions between two users, more than two users, and larger groups? This paper provides the results of experimentation with W4, a WWW browser with CSCW support (described further in [2]), and attempts to answer some of these questions.

2. Related Work

Evaluation of CSCW systems has been noted as being especially difficult due to the different backgrounds of group members, the administrative or personality dynamics within a group, and by the difficulty with trying to emulate realistic groups within a lab by Jonathan Grudin [3]. He also points out that groupware evaluation "in the field" is difficult due to group composition, and a range of environmental factors that may play a role in determining user acceptance, such as training, management buy-in and vendor follow-through. He sees this lack of suitable evaluation as a contributory factor as to why CSCW systems fail to deliver the benefits intended.

Magnus Ramage argues that existing CSCW evaluation techniques are mostly inadequate, because people have spent a lot of time developing methods that are designed to be the one best way to evaluate or design computer systems, but are often based in a particularly disciplinary background and only consider a certain part of a particular situation [4]. His research suggests that evaluation methods need to take into account issues of individual, group and organisational effects as well as questions of useability.

Evaluation of the MEAD prototype [6], a multi-user interface generator tool for use in the context of Air Traffic Control, has provided an insight into many of the problems of evaluation. This evaluation points out that various people have different views about what evaluation actually is, and the multitude of techniques that can be used to perform evaluation. The researchers concluded that their informal evaluation procedures were a powerful, cost-effective means evaluation, yet raised the question of whether systems for use in cooperative work environments could indeed be evaluated for validity in isolation from the work.

Our work provides another case study from which lessons may be learned. The following sections overview W4, a WWW browser with CSCW support, and contain details of what evaluation was performed, how it was performed and lessons learned from the experience.

3. W4 Overview

W4 (World Wide Web for Workgroups) is a collaborative tool developed to allow users to add a variety of annotations to Web pages, which include simple URL links, notes, text chats, a brainstorming tool, and a shared whiteboard. The HTML source of Web pages is not modified in any way – the annotations are stored centrally by a GroupKit [5] W4 conference. All cooperating users join this W4 conference, enabling all annotations to be made persistent and allowing users to join and leave conferences, preserving annotations and their contents. In addition to annotation capabilities, W4 supports various group awareness and work coordination facilities, including telepointers, multiple scroll bars, shared page histories and bookmarks, and the ability to "follow" other users' page visitations.

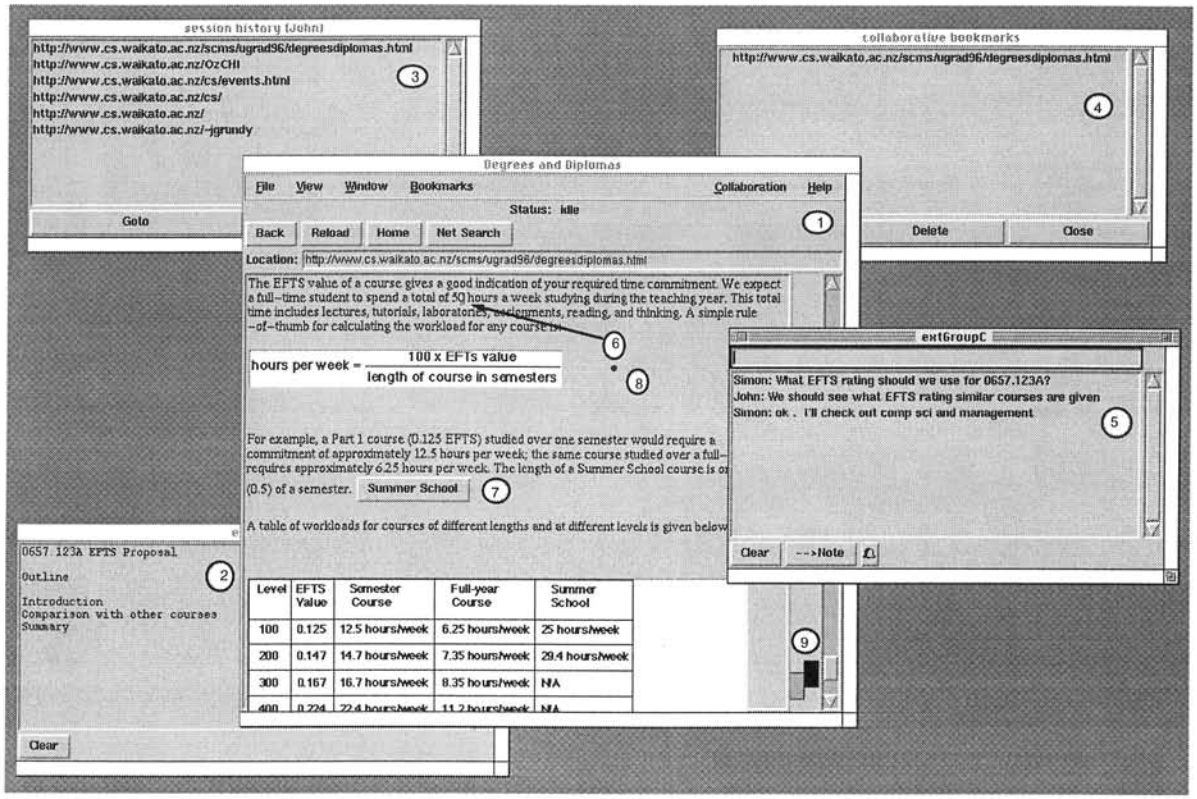

Figure 1: Example of annotated Web page, with collaborative note, text editor, page history and shared bookmarks

Figure 1 shows a screen dump from W4 in use for a cooperative, possibly geographically distributed, task. In this example two (or possibly more) users are collaborating to determine an EFTS (Effective Full-Time Student) rating proposal for a Part I University course, "0657.123 The Computing Experience". In order to perform this task effectively, the participants need to be able to view the same WWW information as each other, be able to annotate pages of interest with notes or URL links to related pages, be able to collaboratively edit text and diagrams either embedded in the page or separate from it, and be able to send email-like messages to each other or communicate in real time. In addition, they need ways to remain aware of each other's work, including pages visited, bookmarked pages of interest, and seeing the focus of attention of their collaborators.

W4 provides a range of facilities to allow collaborators to work together in these ways. As shown in Figure 1, window (1) is the collaborative browser window provided by W4, showing a WWW page from the University of Waikato's Computer Science Department WWW server. The person using this browser is user Simon. The collaborators in this W4 conference also have a text document, in window (2), which they are writing together and which contains the EFTS proposal for the 123 course. This is a collaborative text editor which provides WYSIWIS text editing capabilities. Window (3) shows the session history of Simon's collaborator, John i.e. the WWW pages John has visited while in this conference, and window (4) shows shared bookmarks accessible to all members of the conference. These windows are updated each time John moves to a new page or one conference participant adds a new shared bookmark. Window (5) is a collaborative text chat in which Simon and John have been informally exchanging ideas and discussing the work they are doing. The WWW page has been annotated with a yellow square (a "sticky note" representation), at the position indicated by (6), which when clicked on will display the text associated with this note. Users can also reply to the note, creating further notes, or send context-dependent email-like messages to each other using this notes facility (described further in [1]). A URL link annotation (7) has also been added to the WWW page being viewed, which when clicked on by a collaborator will open the "Summer School" WWW page. Any annotation added to a WWW page is visible by all users, and is shown in other users' browsers when they select the appropriate WWW page, or if already selected will appear when they have been added by a collaborator.

Additional group awareness capabilities provided by the browser include telepointers (8), showing the position of collaborators' cursors. Multiple scrollbars (9) indicate the position of other users on the same WWW page.

Telepointers and multiple scroll bars are only shown for collaborators who are viewing the same Web page as the user. Users can also click on the scrollbar of another user and request to follow their page browsing.

4. What we tested

At the commencement of our useability experiments with W4 we were not entirely sure what information we would obtain. Initially, we focussed on trying to obtain a qualitative measure of usefulness of particular applets under different conditions. This we hoped would guide us in developing tools that users would find useful. Not only did we want our software to provide useful tools, but also tools that were easy for users to learn how to use, and that facilitated cooperative work.

A variety of projects were offered for users to undertake including planning trips, discussing updates to Web pages, and collaboratively obtaining information on a certain topic. By giving users a choice of projects to undertake, users were not restricted to some abstract topic that they knew nothing about. This also gave users the feeling that they could govern what direction the project should take, and that they could adventure out on a tangent if desired.

To ascertain the usefulness of W4 applets, these tools had to be used for different projects, that lasted for different periods of time, with different numbers of users of varying expertise. Experiments were also conducted with users using the tool at the same time, and at different times.

We wanted the opinions of users with various CSCW and WWW experience, to get a broad perspective of the usefulness of W4 applets. This meant users would have quite different mindsets at looking at a problem, and varying degrees of computer, WWW and CSCW experience.

Tests were conducted with different numbers of users, group members of varying expertise, and different genders. This helped to ensure W4 was not being directed at a certain group of users and that it was evaluated for single-user browsing and for small group browsing.

5. How we tested

Qualitative techniques were mainly used to obtain the information we required. Questionnaires before and after W4 tests, observing users at work, and verbal discussion with users provided useful qualitative information about W4 applets and W4 as an environment to work. It was also noted during the post-questionnaire that several users found it easier to communicate their opinions, problems, and suggestions verbally, rather than attempting to put it into written words.

The most useful information was derived from observing users at work, and conversing with them prior, during and after the experiment. A questionnaire was given to all users prior to using W4 to ascertain how familiar they were with the WWW, CSCW systems, and what their expectations were of W4. At the conclusion of the experiment users filled in another questionnaire, and this was focussed on how useful they found the various tools provided for their particular task, and how useful they thought the tools could potentially be (i.e. for other tasks than the one they performed). Users were also asked to comment on their experiences with W4. We did this in order to qualify our judgements of W4 applets and to better understand the responses they had given.

Quantitative techniques built into W4 were used during the early tests of W4. A record was kept of artefact events, when they were made, and by whom. Many different events were recorded, including reading a note, adding text to a whiteboard, viewing a user's session history, and even ringing of the bell. At the conclusion of each experiment these results were analysed. In one experiment that lasted an hour the bell was shown to be used 75 times, which may lead users to thinking it was being heavily abused. However, further analysis of the event log shows that the bell was often used several times in succession (19 times in succession, in one instance) to try and grab a user's attention. Given the extreme "heavy handed" use of the bell by one user, the bell being run 75 times does not give conclusive evidence that users needed better ways of getting a user's attention. This example, although an extreme case, provides an insight into why we found quantitative analysis ineffective. A more complex quantitative analysis would have possibly provided useful information, but this has been left as future work. Since preliminary quantitative analysis results showed nothing evident, future quantitative tests were abandoned.

6. Results

After a few experiments it became clear that the number of users using a tool greatly influenced how the tool was used and how effective the tool was. It was also apparent that certain tasks had different requirements for tools that aided communication.

With groups consisting of two users the embedded text chats and whiteboards were not found to be useful. This was due to the fact that users found that communication (for two users at least) was easy enough via an external text chat, a simple tool that has proven very useful for groups of varying sizes. Context-sensitive Notes were very seldomly used within these small groups. If the users were intending to use W4 for much longer periods of time (e.g. a couple of months) then the notes would possibly be used a lot more, largely as a reminder of things that have occurred previously. Longer-term experiments are currently being conducted to validate this. Directed messages were used when users were working at different times and was the main source of communication during this type of asynchronous interaction, yet the external text chat "took over" as soon as users were working simultaneously.

Groups of three or four users utilised many more of the communication applets, but the predominant applet for synchronous communication was still the simple external textchat. Context-sensitive notes also proved to be a lot more useful, since there was a lot more chance that another user might actually stumble across them.

A collaborative text editor was commonly required by users to compile information retrieved from the WWW, but due to a number of bugs inherent within the text editor provided with W4, it was deemed unusable by users. Users desired a text editor (or even better, a word processor) with group support that they could safely write to, knowing that their text would not accidentally get deleted! Users also found it beneficial that URL links and notes could be embedded within the text editor, since it made the text editor suited to its environment, W4.

Whiteboards were not used commonly, although consultation with the users revealed that it was not because they did not need it, but because it did not provide enough functionality. As with the text editor, applets are required that are robust and bundled with features.

A problem that was observed with the experiments was that users with very minimal WWW experience tended to wander off and look at other things on the WWW, and thereby abandoning the project.

7. Summary

The most basic tools often prove to be the most useful, and from our investigations a simple textchat is useful for groups of varying sizes. Context-sensitive notes and messages are yet another simple idea, yet can also aid users to work collaboratively. Their advantage over conventional email is that they can be associated with work artefacts and are available within the context they are describing.

Collaborative text editors and word processors are important in terms of users compiling information together. They are also useful if they can have links to the context in which they were created (c.f. URL links in W4). Our evaluation of W4 shows that if applets do not provide enough functionality or are unusable due to bugs, collaborating workers will not use them. Applets need to be robust and provided lots of appropriate functionality, in order to suit workers' requirements.

References

1. Apperley, M.D., Gianoutsos, S., Grundy, J.C., Paynter, G., Reeves, S., and Venable, J.R., "A generic, light-weight collaborative notes and messaging facility for groupware applications" Working Paper, Department of Computer Science, University of Waikato, 1996.

2. Gianoutsos, S. and Grundy, J., Collaborative work with the World Wide Web: Adding CSCW support to a Web Browser. In *Proceedings of Oz-CSCW96,* Brisbane, Australia, August 1996.

3. Grudin, J., Why CSCW applications fail: problems in the design and evaluation of organisational interfaces, in *Proceedings of CSCW'88,* Portland, Sept. 1988, pp 85-93.

4. Ramage, M., "Evaluation of Cooperative Systems" First Year PhD Report, Computing Department, Lancaster University, 1995.
(http://www.comp.lancs.ac.uk/computing/research/cseg/projects/evaluation/1YR_contents.html)

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

6. Twidale, M., Randall, D., and Bentley, R., Situated evaluation for Cooperative Systems, in *Proceedings of CSCW'94, Chapel Hill*, Oct. 1994, pp 441-452.

CSCW to Support Participative Systems Development

John R. Venable[†*] and Julie Travis[‡]

† Department of Computer Science
University of Waikato
Hamilton, New Zealand
jvenable@cs.waikato.ac.nz
Telephone: +64 7 838 4401
Fax: +64 7 838 4155

‡ School of Information Systems
Curtin University of Technology
Perth, Western Australia
travisj@cbs.curtin.edu.au
Telephone: +61 9 351 7198
Fax: +61 9 351 3076

Abstract

In this paper, we consider aspects of CSCW needed to effectively support information systems development (ISD). In particular, we are interested in supporting the full and effective participation of all the various stakeholders in the ISD process, not just the technical developers (e.g. analysts, programmers, testers, and project managers). We explore some of the issues and requirements in encouraging full and effective participation, then discuss ways in which CSCW technologies might be better brought to bear to address these issues and requirements.

1. Introduction

Information Systems Development (ISD) is the process of creating a new, or modifying an existing, information system in order to address a perceived problem, or to make an improvement in an organisation. It encompasses problem formulation and conceptualisation, requirements determination, system design and specification, system construction, system installation, system operation, and maintenance. The ISD process involves a very complex mix of technical tasks and social processes. Research in CSCW support for ISD thus far has largely concentrated on later/technical stages of the development and supporting cooperation among the technical participants.

In this paper we will concentrate on CSCW support for the social processes in ISD. Many of the most significant problems of ISD relate to these social issues. As Henry Lucas [4] pointed out, information systems often fail because people resist using them, and that they resist because the systems are not seen by the potential users to address their needs and because the potential users feel that the system is being imposed on them. This occurs primarily because they are not involved in the development process. Involvement in the process increases the likelihood of identifying the correct requirements and getting the users to "buy into" the resulting system. The main lessons seem to be that better participation of the stakeholders facilitates identifying the correct requirements and legitimising the ISD process, resulting in better, more accepted information systems. While advances in prototyping have helped address these problems, it is not always used and it often doesn't cover the full nature of the system, nor does development of a prototype ensure that the system will meet the characteristics identified in prototyping. Better facilities to support the involvement of users and other stakeholders are needed.

2. Requirements for Effectively Supporting Participation in ISD

In addition to Lucas's seminal work, Mumford [7] has pointed out that there are varying degrees of participation, and that the best form of participation is where participants have full decision making power, or even take responsibility for the development, rather than being only consulted early on in the beginning. Mumford [7] further points out that it is the users' right to participate in the decision making about their information systems as the information systems affect how they conduct their work. Dumdum [3] has further analysed the aspects of participation supported by various methodologies (e.g. SSM [1,2] and ETHICS [6]) in his thesis addressing the difficult topic of improving problem formulation.

* Dr. Venable's research is partially funded by the New Zealand Foundation for Research in Science and Technology, Grant number UOW 502.

Drawing on all the above literature, in summary, we propose that CSCW systems to support systems development should include features and functionality to facilitate:

1. Discussion of the problem situation, which includes identification of
   - stakeholder interests
   - assumptions
   - goals and objectives

2. A process of reaching agreement or consensus on goals and objectives

3. Carrying the details of these agreed goals and objectives into subsequent (more technical) stages of the ISD process

4. Continuing stakeholder participation throughout later (technical) stages of the ISD process, in order to reach agreement on and gain approval of artefacts of the technical development process which could affect the stakeholders, including such things as:
   - the formal system specification
   - the system design
   - user job designs and associated (manual) procedures
   - the user interface
   - an appropriate training programme
   - a conversion/installation process

In order to support these tasks, the CSCW support should also embody the following characteristics:

1. Universal participation - *Any* potential stakeholder must be freely able to participate.

2. Accountability and quality assurance - Design decisions need to be accountable with respect to their ability to fulfill agreed requirements. The quality of the design (conformance to requirements) must be assurable.

3. Ability to question - Stakeholders must be allowed to seek explanations from technical developers about how or why design decisions were taken, how they fulfill system goals/objectives, and/or to propose other design alternatives.

4. Continuing involvement - Following on from (3), as the requirements manifest themselves in the details of the implementation, stakeholders must be able to continue their involvement and be able to visualise how the decisions impact on their work tasks, responsibilities, and relationships to other stakeholders. The CSCW system should encourage such continuing involvement.

3. CSCW Features Needed

In order to address the above requirements, the CSCW aspects or features of systems to support ISD need to be improved. Current efforts at supporting ISD with CSCW fall into two categories. The first category includes systems designed to support the technical development process, i.e. they are an outgrowth of CASE tools, which include modifications to support multiple users in a cooperative (but still technical) development process. The second category includes ordinary GSS systems utilised to support the requirements elicitation process. This latter category addresses some of the problems above, but is only utilised as a one-time, front end to the ISD process. The results of the requirements determination are then input to the technical ISD process. These two categories of CSCW support for ISD have not hitherto been integrated and used in a comprehensive fashion to support the entire ISD process.

A CSCW System to fulfill the above requirements needs the following (minimum) features (in addition to those addressing the technical development):

1. A discussion - oriented GSS to which *any* potential user can have access. The stakeholders to be included in the discussion should not be artificially limited. This will address requirement (1) above. The discussion should also be structured in such a way that it addresses the issues included within requirement (1) above, i.e. identification of stakeholders and their interests, assumptions, and potential goals of the system

2. A GDSS to support decision making about the goals to be addressed by the system. This system should facilitate rank prioritisation of objectives and consensus seeking about them. This addresses requirement (2) above.

3. The CSCW system needs to include features to structure the requirements into the documentation capabilities of the CASE part of the system, i.e., they need to be available in a straightforward fashion as input work artefacts to the technical developers of the system.

4. Continuing from point/feature number 3, the CSCW/CASE tools must be able to link design and/or requirements specification artefacts (created by technical ISD personnel) to the structured documentation of the results of the decision making process. Each link should be both forward- and backward-traceable, so that when looking at any technical design artefact created by technical developers, one can follow back to the requirement from which it originated. Similarly, when looking at the structured documentation of the group-decided requirements, one should be able to follow forward to determine which features of the design (or requirements specification) address the requirements. Processes and tools need to be in place to insure that these links are created. This feature addresses requirement (3) above.

5. CSCW features also need to be in place to support the items listed under requirement (4) above. Each of the items listed there (system design, user job design, manual procedure, user interface, training programme, and conversion/installation process) is an artefact which must be designed (although some ISD methodologies treat one or more of them implicitly or not at all). Typically, each of the design artefacts undergo some sort of review when they are complete, either a formal walkthrough or a testing programme. The stakeholders should have the opportunity to participate in such a quality review. They should be notified that it is going to take place. They should be able to trace from their original interest in the requirements to see how they are to be implemented in the system or the installation process. They should also be able to contribute to the quality assessment process and any decision making about whether the work artefact is of acceptable quality. For example, if GDSS facilities are utilised to support the formal review process, stakeholders need to be able to participate in that formal process and be users of the GDSS.

6. Stakeholders should be able to view the state of the development at any time, and to view any work artefacts (except strictly working copies). This includes any design rationale artefacts or other discussion artefacts created during the technical development process. Stakeholders should be able to determine who is or was working on or created a particular technical work artefact and contact and engage in a discussion with that person. This improves accountability and the stakeholder's opportunity for on-going participation (requirement (4) above).

7. It should be possible to capture such dialogs as in (6) in order to document past discussions and review them at a later time if necessary. It should also be possible to reference the discussion and continue it at a later point in time. This further improves the stakeholders' opportunity to participate in an ongoing fashion as in requirement (4) above.

4. A Preliminary Project

We have recently been engaged in creating a system to address part of this set of requirements. In particular, we are working on a system to support the front end of the process, participative requirements determination. We have structured a GSS to support the identification of stakeholders and issues in accordance with SSM. This involves structuring a particular discussion with topics drawn from the constructs in SSM's CATWOE criteria and rich pictures. We have chosen to base our system on a GSS called Discussion*Web* [5], because it is web-based and thus nearly universally available. We are also confronting issues in how to involve people in the discussion, such as providing on-line tutorials on SSM to facilitate stakeholders' ability to participate effectively and ways to feed SSM representations of the current state of the discussion of the problem domain, such as rich pictures, back to the stakeholders. The system as it is implemented so far can be viewed [8] and its objectives, requirements, and design rationale are documented in [9].

5. Further Research

Our immediate plans are to put the existing system into operation on a live project investigating the requirements for an Information Systems Digital Library (ISDL, see [9]). The next step is to refine this system based on our

experiences with it during this project. Following that, we hope to extend it to interface to technical cooperative CASE systems that are being developed within a CSCW research project at the University of Waikato. Finally, we hope to also extend the system to incorporate group-accessible editors for working directly with the rich pictures and CATWOE criteria of SSM.

Bibliography

[1] Checkland, Peter (1981). Systems Thinking, Systems Practice. London: Wiley.

[2] Checkland, Peter & Scholes, J. (1990). Soft Systems Methodology in Action. London: Wiley.

[3] Dumdum, Uldarico Rex (1993). An Approach to Problem Formulation in Ill-Structured Situations in Information Systems Development. PhD thesis, Binghamton, New York: Binghamton University.

[4] Lucas, Henry (1975). Why Information Systems Fail. New York: Wiley.

[5] McQueen, Robert J. & Peeters, Ross (1996). Discussion *Web* home page, http://www.mngt.waikato.ac.nz/dw/.

[6] Mumford, Enid (1983). Designing Human Systems for New Technology: The ETHICS Method. Manchester, England: The Manchester Business School.

[7] Mumford, Enid (1984). Participation - From Aristotle to Today. In Bemelmans, T. (ed.), Beyond Productivity. New York: North-Holland.

[8] Sanson, Marc D. and Venable, John R. (1996). Information Systems Digital Library Requirements Investigation Home Page (draft), http://www.cs.waikato.ac.nz/~msanson/Dev/.

[9] Venable, John R., Julie Travis, and Mark D. Sanson (1996). Requirements Determination for an Information Systems Digital Library. Accepted for publication, 7th annual conference of the International Information Management Association (IIMA'96), 4-6 December, Estes Park, Colorado, proceedings forthcoming.

POSITION PAPER
CSCW in Command Support Systems Group of the Defence Science and Technology Organisation.

Christine Wood
DSTO C3 Research Centre
Defence Science and Technology Organisation
PO Box 783 JAMISON ACT 2614
AUSTRALIA
Phone: # (616) 265 8034
Fax: # (616) 265 8080
email: christine.wood@dsto.defence.gov.au

Abstract

Command Support systems group has developed propotype meeting support tools to support planning in the Australian Defence Force. The research conducted in CSSG on CSCW over the past few years has identified a number of issues and given some insight into future CSCW requirements. This position paper is designed to introduce topics for discussion on issues relating to CSCW and the likely future of this technology.

1. COMMAND SUPPORT SYSTEMS GROUP

The Command Support Systems Group (CSSG) within the Defence Science and Technology Organisation (DSTO) focuses on providing Australian Defence Force (ADF) commanders with the necessary information to optimise their planning and decision-making activities. These are essentially group activities and tools to support these activities will be vital in future command support systems.

CSSG has identified the following areas as key activities or research areas:

1. **Command centre concepts.** This utilises an Experimental Command Centre (ECC) as a command, control, communications and intelligence (C3I) technology demonstration facility. The ECC is equipped with advanced tools and facilities for use in exercise and training in an effectiveness assessment environment.

2. **C2 (Command and Control) Prototypes.** Is involved in prototype development and evaluation in technical concepts for Command Support Systems.

3. **C2 Human Factors.** This comprises human factors specialists in CSSG. The focus of Human Factors is on visualisation, human-computer interaction, organisational structures and behaviour, and the decision making processes of individuals and groups.

4. **C2 Project Support.** CSSG offers impartial advice on Australian Defence Force C2 projects, assists in the development of C2 policy, concepts and requirements, and C2 systems evaluation.

2. PROTOTYPES DEVELOPED BY CSSG

CSSG has been working with the HQADF Directorate of Joint Planning and has developed a prototype meeting support system to aid their work. The system is based on lotus Notes and consists of 3 parts:

- An information catalogue which can group databases together and allows free text searches of all or selected databases for particular topics.

- An agenda database for structuring meetings.

- A brainstorming or idea generation database which allows restructuring and elaboration of these ideas. The information in this database can then seed other databases for other meetings.

CSSG has also given advice on the proposed hardware architecture to be installed for the meeting support system. Training and usability testing was carried out and users took away with them a considerable amount of data gathered at these sessions and gave us useful feedback which has enabled CSSG to fine tune the system for delivery. A system has now been installed and further development work in the areas of linking and visualisation of ideas is planned.

Another prototype which has been designed and developed by CSSG using the Participatory Analysis and Design Assessment is a meeting support system used for staff meetings in CSSG. This system has similar features to those described above. With the aid of templates users are able to create edit and compile agendas, reports and other documents required for a group meeting. The coordinator is also able to notify participants of a meeting. Various reports, minutes and action items are also recorded and distributed electronically to participants for comment and action. The participatory design process was carried out over a period of 2 days, the prototype was built over a period of a week and the prototype has been trialed successfully at staff meetings.

3. MAIN AREAS OF INTEREST IN CSCW

Within the key activities and research areas outlined above CSSG focuses on investigating support to planning processes at strategic and operational levels of the Australian Defence Force commands.

We view the Command Centre as a heterogeneous team. This team consists of a number of specialists from a variety of fields who need to work together to achieve a goal.

Some of the aspects of CSCW which are of interest to CSSG include:

- Electronic meeting support for meetings distributed in time and space.

- Capturing and reusing corporate knowledge in collaborative work spaces.

- Providing tools to enable the linking and manipulation of shared knowledge.

- Visualisation of information, ideas and knowledge.

We are primarily interested in how Command Centres and Headquarters plan Australian Defence Force deployment and are exploring the following issues:

3.1 Types of meetings which use CSCW

Among other tasks Australian Defence Force commanders are involved in a wide variety of meetings, some of which are supported by CSCW. The types of meetings which are currently supported by CSCW include information exchange, brainstorming, continuous meetings, multiple concurrent meetings (on the same or related topics), and subgroup meetings (same time same place. There is at present no one tool which fully supports all of these kinds of meetings.

3.2 How does CSCW interact with other tools?

Collaborative work is more productive than traditional methods. Although CSCW was not originally part of our current C3I system projects it has now been recognised as important. We have to address how best to integrate CSCW into overall C3I systems.

The Australian Defence Force needs to draw on knowledge from meetings and a variety of other sources including computer based systems, and to link information - both text and graphics. This requires visualisation of ideas and information with the emphasis on group work, not just on the individual. This might include mind mapping, organising ideas (flow charts etc), white boards, drawing tools and other visualisation devices.

3.3 Should collaboration be imposed?

Not everything is suited to collaboration and we have found that users feel it is important to maintain a personal workspace (not all systems provide this). This private workspace may later be shared with others as required.

3.4 CSCW changes the way users carry out their work.

CSCW has not only changed the way meetings are conducted but is impacting on organisational culture in relation to information sharing and requiring different skills from their employees. People in senior and managerial positions have a particular skill set and an important skill for a manager is managing meetings. CSCW has given collaborative benefits to groups and organisations but in the process has changed the meeting skills managers require. This creates a problem where not only is the new technology doubted but it also threatens those who profit from the old way of conducting business. This may require managers and senior personnel to learn a new set of skills which ultimately leads to user resistance.

Users have commented that the introduction of CSCW has changed the way they do their work. This has unforeseen impacts on the organisation and often requires a change in organisational culture to permit greater information sharing.

3.5 Will CSCW work in the Defence environment?

The Defence environment requires a high degree of accuracy, and the timely delivery of accurate information and decisions to the right people. Meeting skills and practices have been developed over thousands of years. Is CSCW too immature? Does CSCW need time to develop and mature? At present

CSCW does not focus enough on the social psychology aspects of collaboration and meetings. There are a number of different views on CSCW including:

1. CSCW facilitates communication, ie shared information and may lead to information overload. Often users report that they need more information, until suddenly they realise they have too much information and don't know how to deal with it. CSSG is investigating ways of dealing with this problem.

2. Another view is that CSCW must present structured information. This then creates the problem of how to fit this structure into another structure within an organisation. The Department of Defence has a very hierarchical structure which may impede the implementation of a structure required for efficient use of CSCW tools.

3.6 Does the success of CSCW depend on the skill of the facilitator?

As CSCW is relatively immature we see the role of the facilitator as additional to that of the chairperson and secretary of the meeting. At present the facilitator guides the meeting and coordinates responses. As users become more familiar with this technology it is likely that the role of the facilitator will be taken over by the chairperson or the secretary. Some of the situations in which we have investigated the use of CSCW show there is often disagreement over where "things" should be located. This involves reaching consensus on the outcome of the meeting and making decisions about the structure of information. Often all group members are involved in reading and discussion, then a decision is made regarding the structure of that information. The meeting then continues and further discussion takes place. The chairperson may suggest that one person be responsible for categorising ideas and adjourn the meeting. The meeting is then reconvened. This facilitates the discussion of broader issues and does not limit the participants to discussing the location of each individual item. We need to ascertain how CSCW copes with this.

3.7 What must meeting support tools do?

(a) Meeting support tools must be incorporated into the wider organisational processes and culture.

(b) Over 'hype' is a problem. Guidance and system use helps to align user expectations with what is deliverable.

(c) Often it is necessary to bolt the new CSCW onto an old system (paper or computer system).

(d) Permit access to digital libraries, multimodal sources of information, and virtual libraries. Researchers are currently looking for solutions in this area.

(e) Allow participants to have a private workspace and for participants to have a variety of levels of participation.

(f) Incorporate ad hoc and informal communication.

(g) Facilitate work on ill defined tasks involved in the planning process which change in response to real world situations and to changes in group membership.

(h) Facilitate the organisation and reorganisation of ideas and information.

4. THE FUTURE

In the future anything an individual can do will be able to be done in CSCW. Everything will be "group enabled". Our work with the Australian Defence Force (our customer) has identified a number of issues which are not all obviously CSCW but have CSCW implications. The Australian Defence Force has identified the need for tools to facilitate the following:

- What if scenarios
- Predicting other's intentions
- Distributed planning
- AI decision aids
- Situation analysis and assessment
- Enhancing reuse of information at the corporate level
- Situation awareness display
- Dynamic planning rather than static planning
- Conflict resolution aids
- Capability templating
- Multilingual language translation
- Access to multimodal sources of information
- Knowledge interrogation and filtering
- Dynamic configurable filtering of information, and searching
- Distributed cognition
- Strategic to tactical level control
- Visualisation which aids in conveying ideas
- Different time, different place meetings
- Intuitive use by users from different cultures

Many of these tools have yet to be developed, but they are a good indication of the likely future use of CSCW in the Australian Defence Force.