To be published in Proceedings of OZCHI'95, Wollongon, Australia, Nov 27-30, 1995.

COORDINATING, CAPTURING AND PRESENTING WORK CONTEXTS IN CSCW SYSTEMS

John C. Grundy†, Warwick B. Mugridge††, John G. Hosking†† and Mark D. Apperley†

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

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

ABSTRACT

Large Computer Supported Cooperative Work (CSCW) systems require both high level work coordination mechanisms and low level asychronous and synchronous editing capabilities. We describe an architecture supporting flexible, user-defined work coordination mechanisms, fully integrated with work artefacts. Users define and work within task contexts. When artefacts change, descriptions of the changes are automatically annotated with task context-dependent information. This contextual information is presented (at a suitable level of abstraction) to interested users facilitating coordination between collaborative workers. We illustrate the use of this architecture in a collaborative software engineering environment.

KEYWORDS

computer-supported cooperative work, work coordination, work contexts, software engineering environments

INTRODUCTION

The Need for Coordination in CSCW Systems

Much recent research into Computer Supported Cooperative Work (CSCW) systems has focused on providing low-level interaction mechanisms, such as synchronous and asynchronous editing support [13, 4]. For example, C-SPE is a collaborative environment for software development [6]. Figure 1 shows a screen dump from C-SPE during semi-synchronous view editing. Changes made by another user ("rick") are presented semi-synchronously in work artefact views or in a dialog. C-SPE also supports asynchronous development where different versions are merged, and synchronous view editing, where users see and manipulate exactly the same view information.

C-SPE's low-level editing mechanisms, like those of most other such systems, do not support the coordination of work, and only deal with the editing of work artefacts. C-SPE thus illustrates the key problem of systems providing only low-level editing for CSCW work: the lack of information about the context that changes have been carried out in. The collaborating user is not told why rick changed things, only the sequence they were changed in. No support is provided for planning work nor for grouping changes into histories based on particular tasks or subtasks.

Some work has been done on providing higher-level coordination mechanisms for CSCW, such as workflow configuration [12], software process protocols [8], and shared workspace awareness [3]. Usually, however, these systems are separate from the work artefacts or editing tools, or are not utilised to provide work context information when work artefact changes are presented. Thus their usefulness for integrated work coordination is limited.

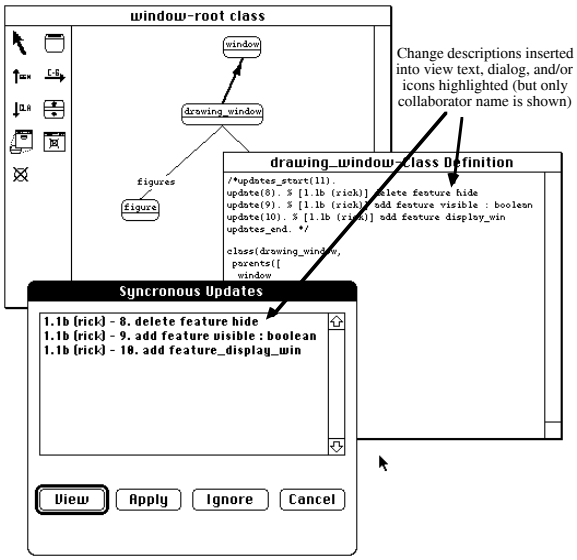

Figure 1. Asynchronous and Semi-synchronous editing in C-SPE

In large CSCW systems, such as collaborative software engineering environments, coordination of cooperative work activities is needed. Users must collaborate in the planning of work activities and be aware of the contexts in which other users' work is carried out. Support is needed for defining activities to be done (plans), coordinating planning itself (meta-plans), and restructuring the history of work done to better convey intent ("rewriting history"). Such a system should encourage users to build good work plans and histories. Users should be able to structure the rationale for their work according to their current work context. Work and plan artefacts, and extra rationale, should be shared. Users should be able to specify the coordination and work artefact changes they are interested in, and how the

work or plan context of these changes is to be captured and presented to them. Capture of work context and rationale should be as unobtrusive as possible [2].

Related Research

Examples of collaborative environments which support low-level editing mechanisms include most groupware systems [4], Mercury [7], Mjølner [11] and C-SPE [6]. As these systems do not facilitate coordination of work, nor the capture and presentation of work contexts, effective collaborative development of large systems is not possible. Some systems support limited group awareness capabilities [13], such as multiple cursors, but these usually only inform collaborators about the work artefacts of immediate interest, not their overall work contexts.

Process-centred software engineering environments utilise information about software processes to enforce or guide software development. Examples include Marvel [1], CPCE [10], and ConversationBuilder [8]. These environments usually provide low-level text-based descriptions of work rationale, and often do not effectively handle restructuring of software processes while in use [14].

Workflow-based systems attempt to coordinate work by describing the flow of documents between collaborators. Examples include Active Workflow [12], and Domino [9]. Workflow approaches have proven inadequate for most real-world coordination activities, as exceptions to workflows usually outnumber cases when they are useful. Workflows also usually need to be modified while in use, and such systems usually do not model nor facilitate collaboration on the coordination (planning) activity itself [14]. Most workflow systems and process-centred environments have only a tenuous integration with the work artefacts being modified. Thus the context of coordination and work artefact changes is difficult or impossible to obtain and present.

Our Approach

We have designed extensions to C-MViews, the low-level CSCW architecture used to build C-SPE [6], which support more effective coordination of collaborative work. Collaborating users design work plans together, and can abstract these into reusable policies, using VPL (Visual Planning Language) [14]. Both plans and policies are very flexible and can be modified before, during or after actioning. Work artefact changes are carried out in association with a current plan stage, the current work context. Descriptions of work and plan artefact changes are augmented with extra information, which captures the current work or planning context, and any extra rationale for the changes. Collaborating users register their interest in plan, policy, or work artefact changes, so that when these artefacts are modified by collaborating users, the user is informed of the change in an appropriate manner. Collaborating users are informed of changes by descriptions of changes made and the context the work was carried out in. This includes grouping change descriptions with their plans or meta-plans, showing collaborators the context of work when displaying change descriptions, and highlighting affected plan and work artefacts in various ways.

WORK CONTEXTS AND CONTEXT SETTING

Defining Work Contexts

Rather than inadequate workflow or process model approaches, we have adapted Visual Planning Language (VPL) [14] to define the context of work and planning activities. VPL allows the definition of plans and subplans for work tasks, and can be used to specify meta-plans for the planning process itself. Useful plans can be abstracted out into policies, which can then be instantiated into further plans. VPL supports flexible restructuring of plans while in use, and partially defined plans can be actioned and later completed. The history of what actually happened when using a plan can be used to restructure the plan to better document the process and to refine it into a policy.

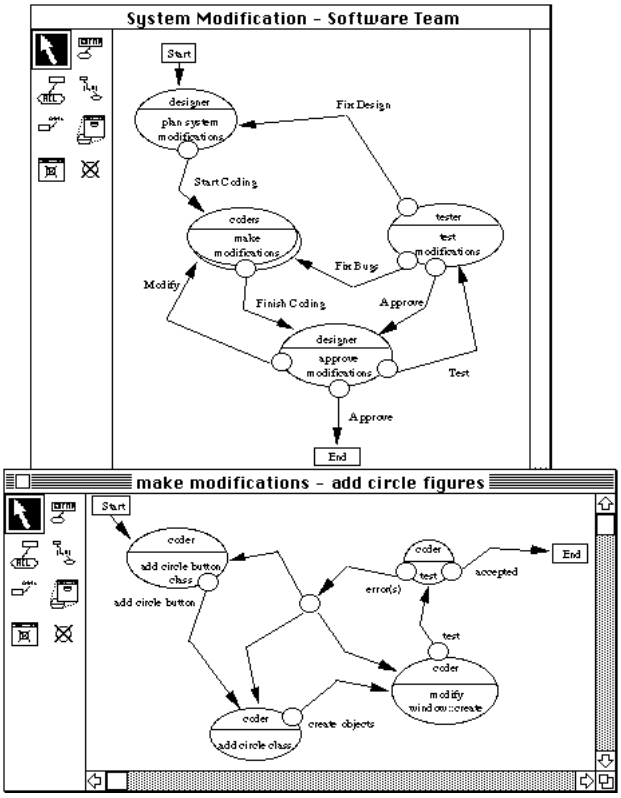

Figure 2. Defining work contexts.

A VPL plan diagram for a software process is shown in figure 2. VPL elements include: stages (steps in the process), denoted by elipses which include a role and stage description; split stages, which are duplicated for each person involved in the stage; and options, denoted by circles on the stage perimeter, used to specify the the next plan stage. VPL elements are “plan artefacts”, modified in the same way work artefacts are in C-SPE. Plans are defined by users via C-MViews-based synchronous, semi-synchronous or asychronous editors. Plan modifications are coordinated via meta-plans (themselves VPL plans).

The figure 2 plan specifies a software process for modifying a software system. A subplan for the "make modifications" stage specifies particular steps for a particular system enhancement. A plan history view is associated with each actioned plan, containing descriptions of changes made to both plan and software artefacts in accordance with this plan. Textual views describing extra information about the purpose of individual plan stages are also supported.

Setting and Capturing the Context of Work

The same notation used to design plans is used to display and manipulate plans in action. Active stages for an activated plan are highlighted, and to advance plan stages users interact with stage options. Obligations can be specified between plans and plan stages. These ensure users of related plans are informed of changes to a plan.

The plan view specifies the current work context for each member in a collaborating team. A user may have several plan views open to see the status of other plans they are interested in or are working with. Plan views indicate the current work context of other users by their respective roles. When making work artefact or plan artefact changes, information about the context of this work must be captured and presented to interested collaborating users. The information required includes who made a change, what was changed, when it was changed, why it was changed and the work or planning context in which the change was made.

For example, consider a software system for a drawing editor which is being modified by several software developers. This drawing program is being extended to incorporate support for circle figures and draggable figures. Team members are using the plan for software system enhancement from figure 2. Figure 3 shows different kinds of views team member "John" might use while working on adding circle figure support (only some of these are displayed at any one time). Team member "Mark" is adding draggable figures, and team member "Rick" is team leader.

This example shows the kinds of supported views and the ways these views are used to capture work context information when users modify work artefact or plan views:

1. A "current plan" view shows John's current active stage. This captures the current work or plan context for the changes being made.
2. Shared VPL plan artefact views show other plan stages John is currently working on i.e. further contexts for John's work. These also highlight the active stages of John's collaborators, and change when active stages or the plan itself changes.

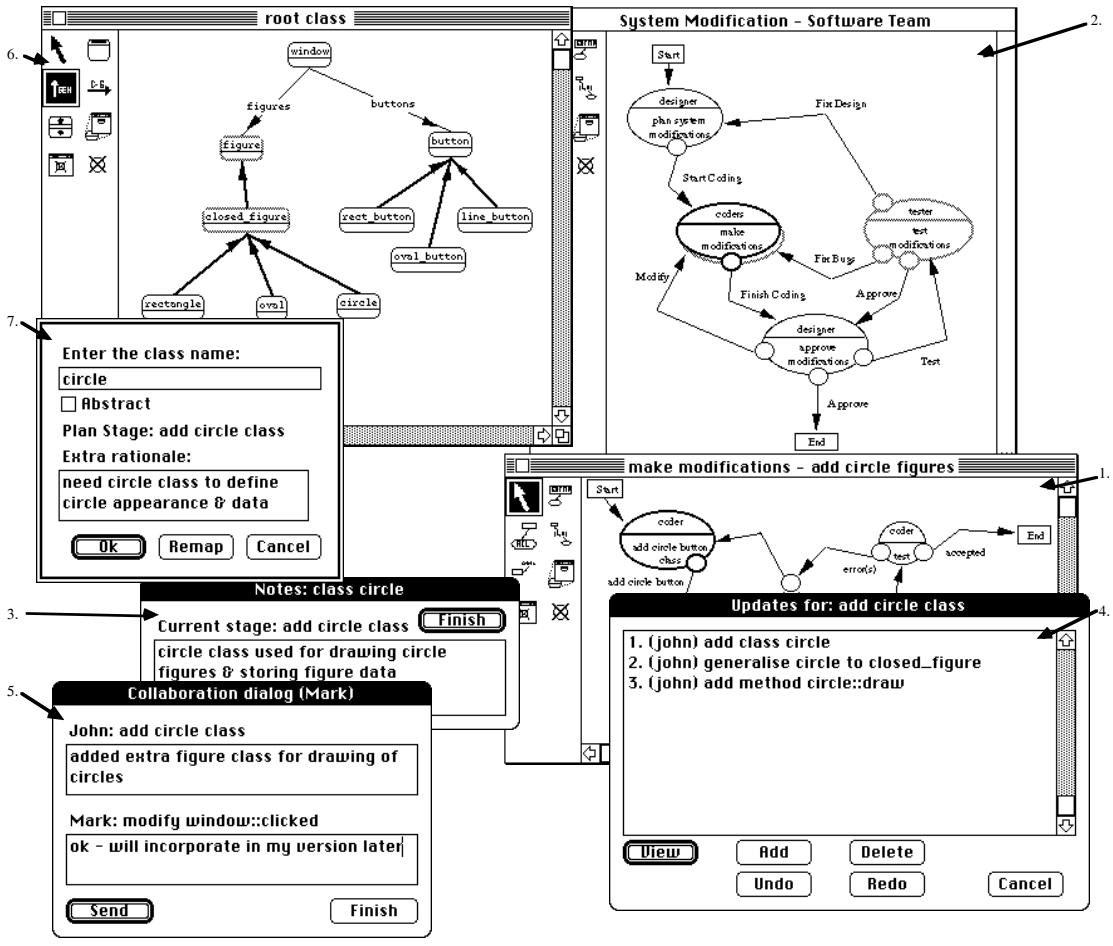

Figure 3. Context awareness.

3. Artefact notes dialogs capture artefact documentation. These, like work and plan artefact views, are shared between all collaborators.
4. Shared modification histories for the current plan stage, or other plan stages John wants to review the history of. These detail the actual changes that have been made for that stage. As plan or work artefact changes are generated, descriptions of these changes are forwarded to the appropriate current plan stage and stored to document the stage's subplan history. These historical changes may also be used to "rewrite history" if the plan is restructured. C-MViews automatically generates objects representing these change descriptions [6].
5. A collaboration dialog for a collaborator (Mark), is used for informal dialog between John and Mark, and changes when Mark's current stage changes.
6. Work artefact views are graphical or textual editors which capture information about changes made to artefacts. Plan artefact views are manipulated in the same manner, co-ordinated by meta-plans.
7. Augmented work artefact/plan artefact dialogs allow John to optionally specify extra rationale for low-level work or plan artefact changes. These reasons are stored with the plan stage history and communicated to collaborators.

The current context of work is modified by advancing VPL active stages using the specified stage options. Incomplete VPL views can be extended as users become more familiar with their work processes, unlike many existing workflow and process-centred systems. Plans can also be restructured while in use, using meta-plans to coordinate this process. Plan history items can be split up if plan stages are split, or moved to other stage histories if plans are deleted.

PROCESSING WORK ARTEFACT CHANGES

When work or plan artefacts are modified, descriptions of these changes must be sent to collaborating users. Collaborators may be informed immediately the change is made, some time after the change is made, or may view the change and reasons/context for the change on request.

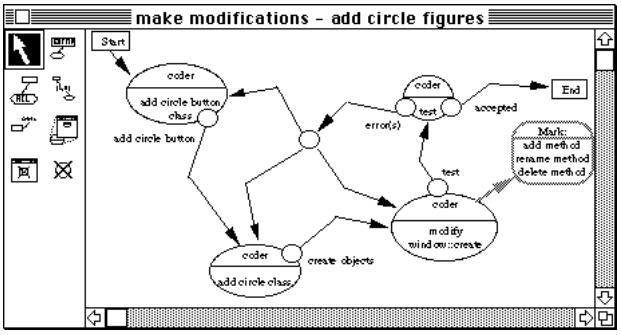

Figure 4. Registering interest in changes.

Collaborators must thus be able to specify which changes they are interested in seeing and when and how these should be presented. We have extended VPL to include annotations for specifying obligations between plan stages, to allow users to register interest in particular kinds of plan or work artefact changes, and for defining extra change description processing. Figure 4 illustrates the basic annotation indicating which changes to detect. Extra information about how to present these changes to interested collaborating users is specified in a dialog. This notation allows users to specify interest in either particular or general kinds of work or plan artefacts. Changes to these artefacts or instances of these artefact types are then communicated to the collaborating users as they occur.

Automatically updating plan artefacts in response to changes or change sequences allows an environment to automate some coordination tasks. For example, Rick may need to approve certain kinds of changes or sequences of changes made by John. The collaborative environment can detect such changes and update plan artefacts or stage histories to inform Rick of changes he must approve. Aggregating related change descriptions into higher-level change descriptions, and storing or forwarding these, is useful for gaining a higher-level overview of changes.

PRESENTING THE CONTEXT OF WORK

Collaborating users must be informed of changes to work and plan artefacts they are interested in. In most CSCW environments this amounts to presenting only artefact-level information to collaborators, either directly updating their work artefact views or using version control facilities to indicate changes made by other users. Our approach provides collaborating users not only with C-SPE change descriptions describing actual work or plan artefact changes, but also with extra information about the work context the changes were carried out in.

For example, Mark is collaborating with John to extend the drawing program, and thus is sharing many work and plan artefact views with John. Some changes John makes are directly relevant to Mark and Mark should be informed of these immediately, such as renaming of classes or methods Mark is using. Note that Mark may have a different version of the artefact view, and hence may not need or want to act on the change immediately, but delay incorporating the change until later. Other changes, such as the addition of new classes or methods can be sent to Mark for later perusal, as they have more limited effects on Mark's work. Low-level changes, such as the implementation of methods or classes Mark is not affected by, need not be presented to Mark. Mark can, however, see from John's plan histories and various active stages the kinds of activities John is doing, and may choose to view these changes or modified artefacts on-demand.

Mark is informed of changes in various ways. Figure 5 illustrates how Mark's views carry this out:

1. Mark's current plan view specifies Mark's current planning or work context.
2. The shared VPL plan view indicate's John's current work context, and any change to this context (new stage, changes made, change to shared plan(s)).

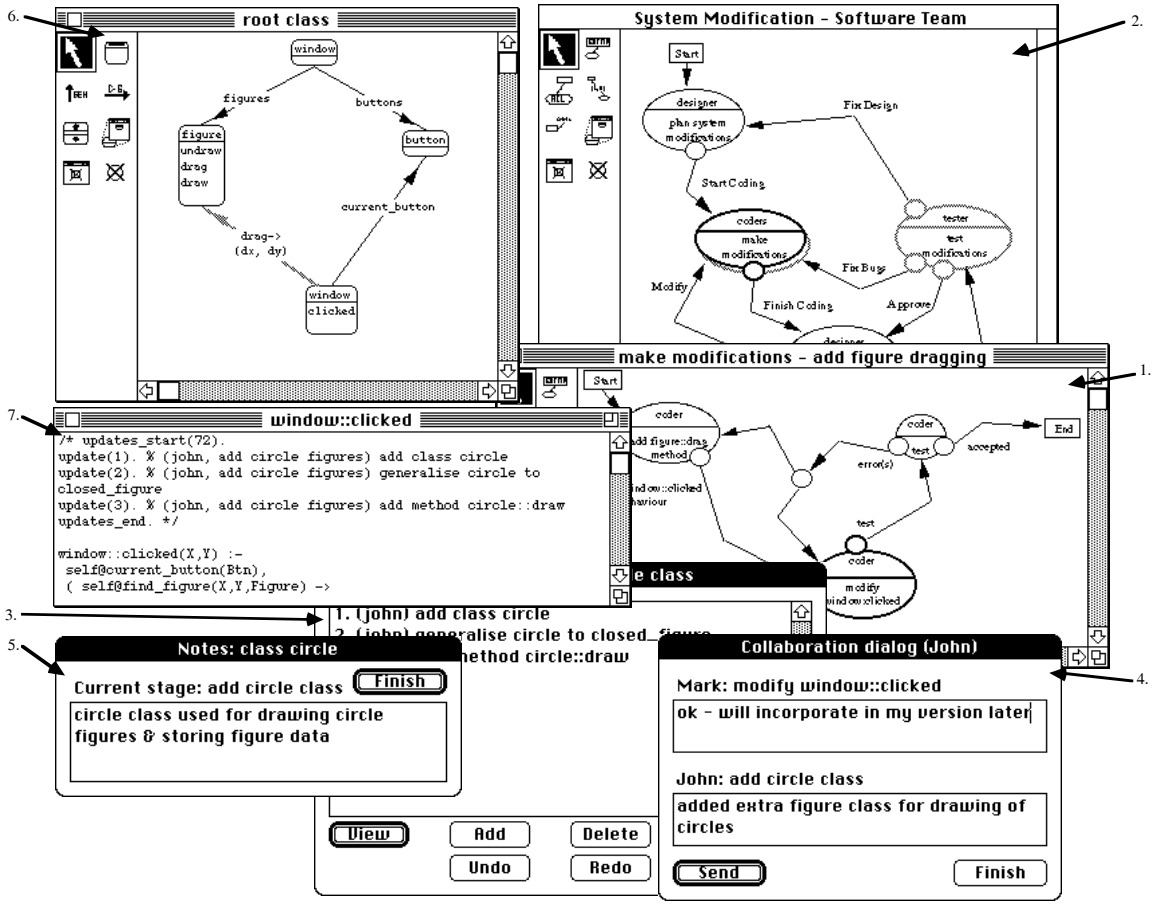

Figure 5. Presenting context information.

3. Mark monitors John's changes, displayed semi-synchronously in a plan history dialog.
4. Informal messages from John are displayed to Mark in a collaboration dialog. Mark can respond to these as they are received.
5. Changes to shared work and plan artefact notes are presented synchronously to Mark. Mark can add extra comments, which are sent to John. Changes to notes are documented in plan histories, and thus can also be viewed asynchronously on-demand.
6. Work artefact views are edited synchronously, semi-synchronously or asynchronously, as in C-SPE. Changed view items can be highlighted, and Mark can see the view modification history or its work contexts' plan histories on-demand.
7. Additional information is presented with change descriptions in work artefact views, including the work context (plan stage) the changes were made in. Hypertext links allow quick access to plan histories and VPL views. Modification histories for individual work or plan artefacts also display the work context each change was made in. Users review change histories grouped by artefact changes (artefact histories) or work context (plan histories).

Not all of the above presentation techniques would be used by Mark at once. Mark determines these by opening views or adding interest/presentation annotations to VPL views.

A HIGHER LEVEL CSCW ARCHITECTURE

C-SPE is built using the C-MViews architecture [6]. C-MViews provides facilities for building collaborative, multi-view editing environments, and extends the MViews architecture [5]. Whenever an artefact is modified, an object documenting the change, called a change description, is generated. For example, renaming an attribute generates a change description of the form update(Artefact, Name, OldValue, NewValue). Change descriptions are propagated to all other artefacts dependent on an updated artefact's state, and these dependents update their own state. C-MViews broadcasts these change descriptions to support semi-synchronous and synchronous editing, and stores them for use in version merging to support asynchronous editing.

We have extended the design of C-MViews to support higher-level coordination of work by adding VPL artefacts and views, and routing all change descriptions to a "current context" (active plan stage) artefact. This stores the change description in its plan history and forwards it to interested collaborating users. Figure 6 illustrates this architecture:

1. A work (or plan) artefact is updated
2. Generated change description(s) are augmented with the work context then stored in an artefact history
3. Change description(s) are forwarded to the current work context (current active plan stage) for the user
4. The plan stage stores change description(s) in its plan history, then forwards them to interested users. Interest in the change is determined by extra VPL annotations. Interest is hierarchical, so interest in a plan includes interest in any subplans. Extra processing, such as composition of changes, is carried out at this stage. Related plan artefacts may be modified in response to these changes.
5. Change descriptions are presented to collaborating users. The extra work context information is also presented in various ways. Change descriptions may be stored in a plan or work artefact history for on-demand viewing by the collaborating user.

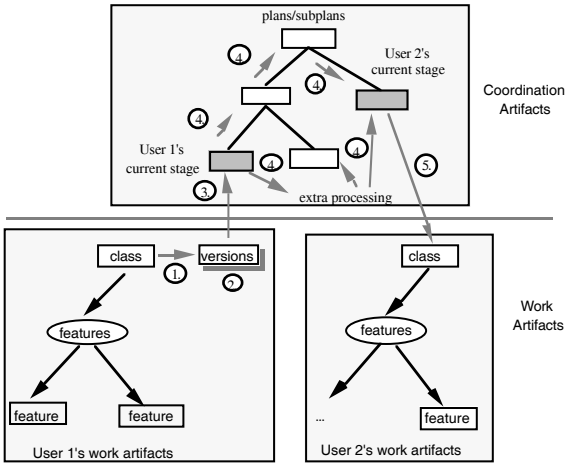

Figure 6. Supporting coordination of work.

Single-user MViews environments become collaborative environments using these extended C-MViews facilities. No changes need be made to work artefact or view implementations to support our coordinated, collaborative approach, as change descriptions are intercepted using standard MViews facilities. High performance synchronous and semi-sychronous editing facilities are unaffected by the coordination layer, as C-MViews provides these directly [6].

CONCLUSIONS AND FUTURE RESEARCH

We have described a new approach to providing coordination of work and work planning for CSCW systems. This utilises extended Visual Planning Language views to define the context of work, and this work context is associated with descriptions of all changes made to work or plan artefacts. Collaborating users register their interest in these changes, and descriptions of the changes, including the work context they were made in, are presented in various ways to these users. The main advantages of our approach over other CSCW systems include its support for work coordination, informing collaborators about the context work artefact changes are made in, and the ability to add this coordination on top of any MViews environment.

We are currently implementing these extensions to C-MViews and thus extending C-SPE to support coordination of software development. We are further developing notations to define interest in changes, extra processing of change descriptions, and how to present change descriptions and their work contexts to collaborating users.

REFERENCES

1. Barghouti, N.S. Supporting Cooperation in the Marvel Process-Centred SDE. In *Proceedings of the 1992 ACM Symposium on Software Development Environments,* ACM Press, 1992, pp. 21-31.

2. Cockburn, A. and Jones, S. Four Principles for Groupware Design. In *Proceedings of OZCHI'94,* 1994, pp. 21-26.

3. Dourish, P. and Bellotti, V. Awareness and Coordination in Shared Workspaces. In *Proceedings of CSCW'92,* ACM Press, 1992, pp. 107-114.

4. Ellis, C.A., Gibbs, S.J., and Rein, G.L. Groupware: Some Issues and Experiences. *Communications of the ACM 34*, 1 (January 1991), 38-58.

5. Grundy, J.C. and Hosking, J.G. A framework for building visusal programming environments. In *Proceedings of the1993 IEEE Symposium on Visual Languages,* IEEE CS Press, 1993, pp. 220-224.

6. Grundy, J.C., Mugridge, W.B., Hosking, J.G., and Amor, R. Support for Collaborative, Integrated Software Development. In *Proceedings of the 7th Conference on Software Engineering Environments,* Netherlands, IEEE CS Press, April 5-7, 1995, pp. 84-94.

7. Kaiser, G.E., Kaplan, S.M., and Micallef, J., Multiuser, Distributed Language-Based Environments. *IEEE Software* (November 1987), 58-67.

8. Kaplan, S.M., Tolone, W.J., Carroll, A.M., Bogia, D.P., and Bignoli, C. Supporting Collaborative Software Development with ConversationBuilder. In *Proceedings of the 1992 ACM Symposium on Software Development Environments,* ACM Press, 1992, pp. 11-20.

9. Kreifelts, T., Hinrichs, E., and Klein, H.K. Experiences with the Domino Office Procedure System. In *Proceedings of the Second European Conference on Computer Supported Cooperative Work (ECSCW'91),* 1991, pp. 117-130.

10. Lonchamp, J. CPCE: A Kernel for Building Flexible Collaborative Process-Centred Environments. In *Procs of the 7th Conference on Software Engineering Environments,* IEEE CS Press, 1995, pp. 95-105.

11. Magnusson, B., Asklund, U., and Minör, S. Fine-grained Revision Control for Collaborative Software Development. In *Proceedings of the 1993 ACM SIGSOFT Conference on Foundations of Software Engineering,* Los Angeles CA, December 1993, pp. 7-10.

12. Medina-Mora, R., Winograd, T., Flores, R., and F., F. The Action Workflow Approach to Workflow Management Technology. In *Proceedings of CSCW'92,* ACM Press, 1992, pp. 281-288.

13. Roseman, M. and Greenberg, S. Groupkit: A groupware toolkit for building real-time conferencing applications. *Proceedings of CSCW'92,* ACM Press, 1992, pp. 43-50.

14. Swenson, K.D. A Visual Language to Describe Collaborative Work. *Procs of the 1993 IEEE Symposium on Visual Languages,* IEEE CS Press, 1993, pp. 298-303.