In Proceedings of OZCHI'96, Nov 24-27 Hamilton, New Zealand, IEEE CS Press.

Low-level and High-level CSCW Support in the Serendipity Process Modelling Environment

John C. Grundy†, John G. Hosking††, Warwick B. Mugridge††

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

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

Abstract

In order to effectively collaborate using large cooperative work systems, both low-level and high-level CSCW facilities are required. Low-level mechanisms, including collaborative editing, messaging, annotations and communication, are needed. For large-scale cooperation, they should be augmented with higher-level process modelling, work coordination and work context awareness facilities. We describe the integration of both low-level and high-level support for cooperative work in the Serendipity process modelling environment, and discuss our experience of using these facilities.

1. Introduction

Much recent work has focused on integrating CSCW technologies, process-centred environments and existing tools for doing various kinds of work, such as software development and general business activities. Examples of such efforts include those of Di Nitto and Fuggetta [25], wOrlds [4], Oz [32], and those discussed in [20]. However, much work still needs to be done to achieve true integration of these technologies [20, 25]. Problems with existing approaches include the difficulty of adding CSCW capabilities to existing tools, finding a balance of responsibilities between the process modelling and CSCW parts of an environment, and achieving true, high-level support for cooperative work in a large environment.

The Serendipity process modelling environment supports the definition, enactment, reuse and improvement of work processes via high-level, graphical process model views [12]. It also integrates both low-level and high-level CSCW capabilities. Low-level cooperative work facilities include collaborative text chats, notes, messages, view editing, and version merging. Higher-level work coordination capabilities include group awareness via enacted process model animation, annotation of deltas with work context information, and work context definition, determination and visualisation. These capabilities have been added to the environment using a variety of techniques, including integration of existing CSCW tools, addition of CSCW capabilities to the framework used to construct Serendipity, and use of Serendipity's own graphical event processing notation to define actions to assist in supporting cooperative work.

This paper presents a case study of using the CSCW capabilities of Serendipity integrated with SPE, an Integrated Software Development Environment (ISDE) for object-oriented software development [8]. This case study uses the ISPW6 software process model, which focuses on designing, coding, unit testing and managing a localised change to an object-oriented video rentals system [19]. We also briefly describe the architecture and implementation of Serendipity and our future research directions.

2. The Serendipity Environment

Serendipity is a process modelling, enactment and work planning environment, which also supports event handling, group communication, and group awareness facilities [12]. 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 [29] but extends it to support artefact, tool and role modelling, and arbitrary event handling.

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).

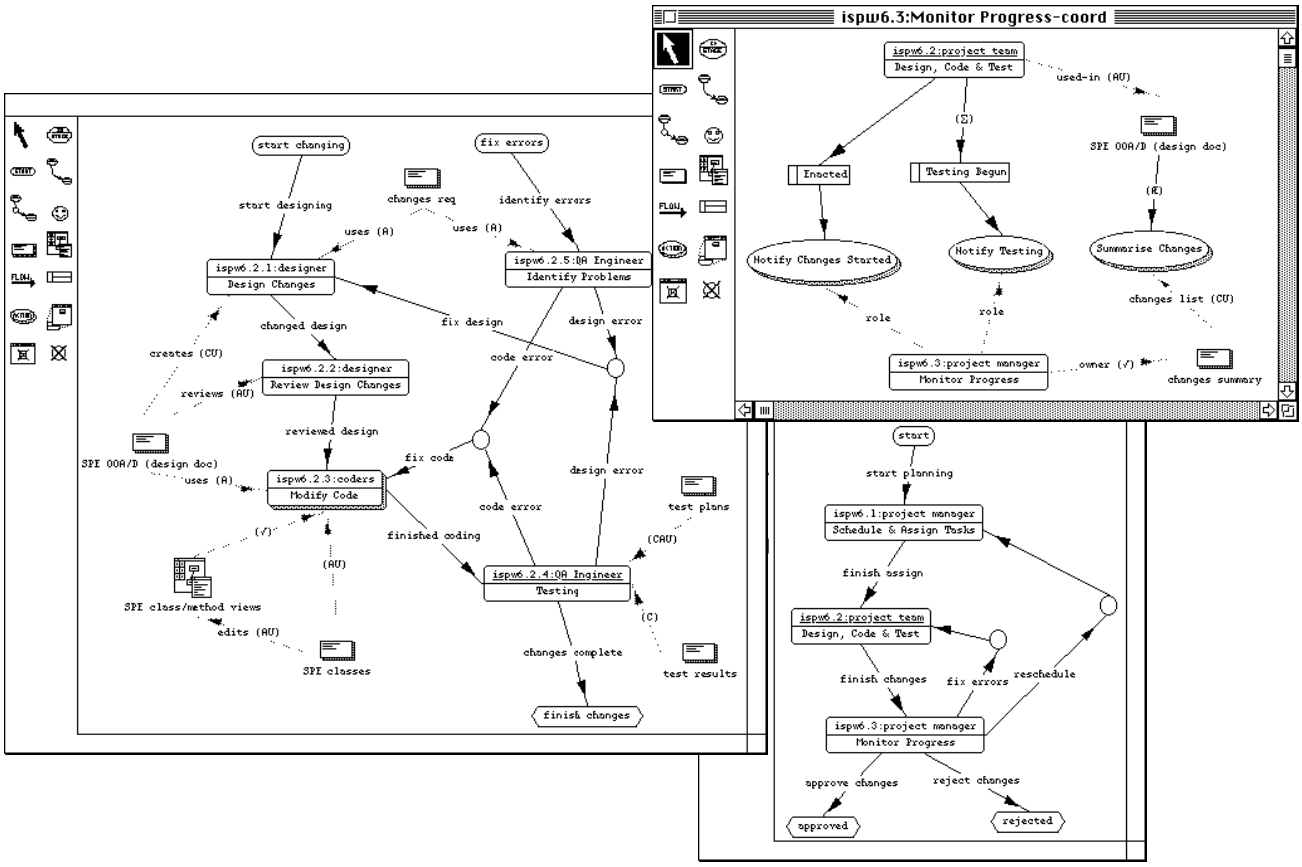

Figure 1. Part of the ISPW6 software process example modelled in Serendipity.

Modularity is provided in the form of hierarchical subprocess models. The window at the left of Fig. 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 Fig. 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. Low-level CSCW Support

In order to use the general process model defined in Fig. 1, users create copies of this "template" for the specific project they are working on. For example, in

Figure 2, a project team is updating the video rentals system by adding a fining facility. The ISPW6 process template has been instantiated as process model “aff” (add fines facility), with roles and artefacts also instantiated (Designer = “judy”, Coder = “john”, QA Engineer = “rick”, etc). The model is then enacted and each project team member may view the process model views, enact and complete stages, etc. as permitted by the filter/actions defined for these models. For example, in figure 2 the enacted process model is being viewed by user “judy” (the “Designer”). Stage “aff.2.2:Review Design Changes” is highlighted, indicating it is her current enacted stage.

Collaborative notes, messages and talk-style dialogues, based on the MVNotes system [1], are integrated with Serendipity to facilitate collaborative dialogue between collaborators. These are context-sensitive, recording the artefact(s) and stage(s) (i.e. the work contexts of the collaborators) they are associated with, in contrast to traditional email and talk systems. Messages and dialogue contents are recorded as notes associated with the appropriate stage/artefact to retain a permanent, shared history of communication for a project.

Figure 2 illustrates these facilities. Two collaborators are discussing design decisions using a simple talk-style text chat (“Conversation with rick”), and user “rick” has left a shared note attached to process “aff.2.5:Identify Problems”, read here by user “judy”. Developers may further refine their work plans, define extra filters/actions to coordinate their work with other people, and so on. This may even be done while the model is enacted.

For example, in Figure 3 coder “john” has defined a new view and added filter/actions to keep him aware of modifications done by “judy” ("judy" here is used as a filter, thus selecting only her changes) on the artefact “video class” (by storing the artefact changes in a “change list” artefact), and to inform him when “rick” starts work on “Do Testing”. This ability to extend process model specifications with arbitrary artefact and enactment event handling mechanisms provides an extremely flexible way for users to specify interest in, and be made aware of, others’ work.

Serendipity and SPE support asynchronous, semi-synchronous and synchronous editing of process model and software development views [9, 10, 12]. We have found synchronous editing, where all developers share the same view version, is most useful for high-level process model or analysis/design view changes. Semi-synchronous editing, where each developer has an alternative view version, propagates changes to other developers, but the change is not immediately actioned. This is most appropriate for lower-level process models and design views, which are shared but may need to evolve separately for a time. Asynchronous editing, where developers have alternative view versions, edit these separately and then merge changes, is most appropriate for personal process models and SPE implementation views.

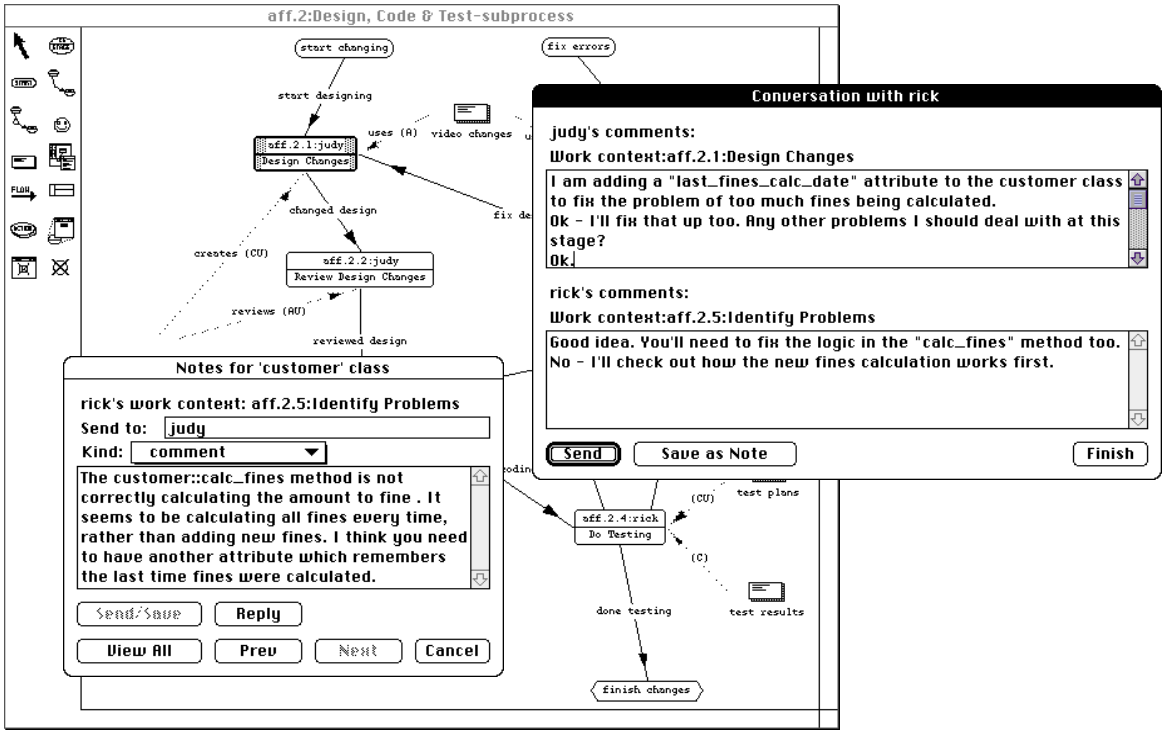

Figure 2. Collaborative notes and text chats in Serendipity.

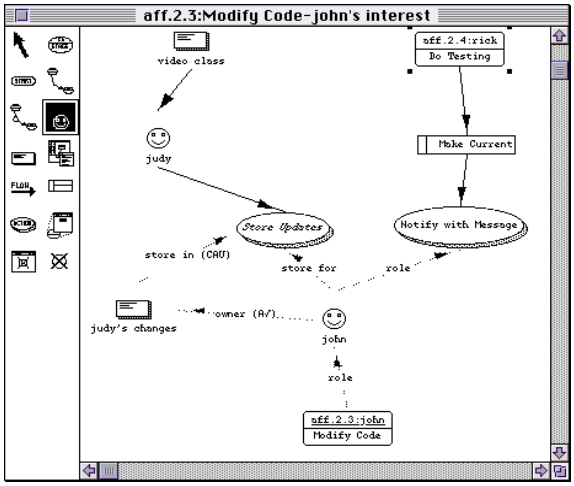

Figure 3. Actions used for low-level awareness.

4. High-level CSCW Support

In order to effectively collaborate on a large project, developers need higher-level awareness support in addition to the low-level messaging, communication and view editing mechanisms described in the previous section.

One aspect of awareness requirements is the need for collaborating users to be aware of other's work and be informed of tool events and changes to artefacts they are interested in. Figure 3 showed a low level approach to this using filters/actions to define how and when users are informed of changes that interest them. Figure 4 shows a higher-level approach using animation of the Serendipity process models.

Enacted stages (ie those that *may* be worked on) for all users are highlighted; the users who have enacted the stage can be viewed in a dialog. Current enacted stages (ie stages currently being worked on) are further highlighted with coloured borders, with different colours assigned to each collaborator. Built in template actions can also be used to highlight the tool(s) and artefact(s) currently (or recently) used by collaborators (for example, designer "judy" can see that the SPE class/method view editor is being used by coder "john" to edit the "video system classes" artefacts). Our integrated SPE–Serendipity environment allows developers to work in SPE (or Serendipity) and have views from the other environment open. This is illustrated in Figure 4, with an SPE class diagram and class interface view open, and a Serendipity model in the background.

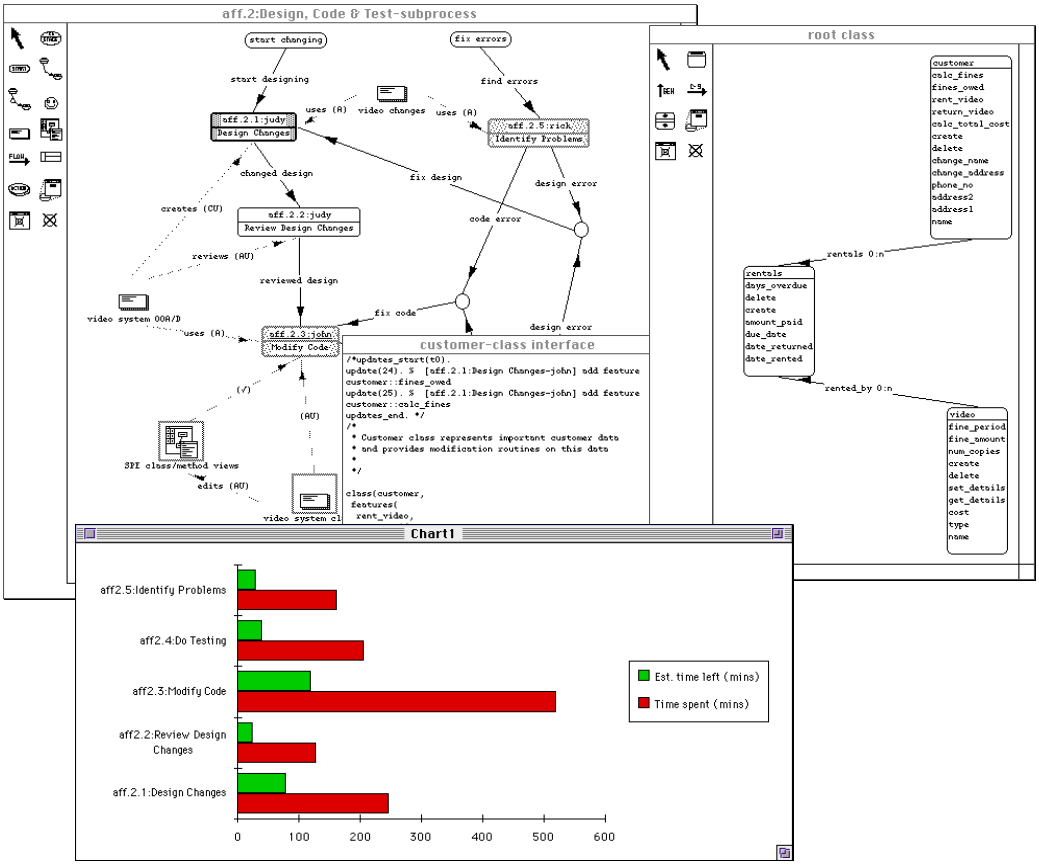

Figure 4. Examples of work context awareness and communication support.

Additonal high level support is provided by process model analysis and metrics facilities. These enable users to analyse process models and work histories, thereby assisting in the improvement of these processes. The complexity of subprocess models, ie the number of work artefacts stored by a stage, time spent by a stage as the current enacted stage, and so on may be analysed and graphed (via MS Excel [13]), as shown in Figure 5.

Another aspect of awareness support is the need to be aware of why changes have been made. For example, when views are modified in SPE, "change descriptions" are generated to document these edits. SPE (and Serendipity) use these change event representations to maintain view consistency, by presenting change descriptions to developers to indicate view inconsistencies [10, 8]. However, change descriptions only describe *what* has changed, not *why* it has changed nor *who* made the change. Explicating and capturing this "work context" information is a key component of awareness support.

Using Serendipity, work contexts correspond to the currently enacted process or work plan stage of each user. When modifications are made to a work artefact using SPE, the change descriptions generated are augmented with the current context by Serendipity. These change descriptions are also stored by Serendipity in artefact change histories for each process stage. An example is shown in Figure 5 for designer "judy"'s work modifying the video and customer tables. The left hand (Serendipity) dialog lists the artefact changes made for the aff2.1 stage. The right hand (SPE) dialog shows a history of changes made to the customer class. The centre SPE textual view of class customer includes a header indicating currently unenacted changes to that view sourced from another view. Each change is annotated with work context information.

5. Architecture and Implementation

Serendipity itself has no built-in tools for performing work, and so must be combined with software development tools, such as SPE, or office automation tools. SPE and Serendipity were developed by reusing the MViews framework for constructing ISDEs [6, 7, 11]. MViews provides a general model for defining software system data structures and tool views, with a flexible mechanism for propagating changes between software components, views and tools. ISDE data is described by *components* with *attributes*, linked by a variety of *relationships*. Multiple views are supported by representing each view as a graph linked to the base software system graph. Each view is rendered and edited in a graphical or textual form. Tools are interfaced as view editors, by external translators, or multiple base layers.

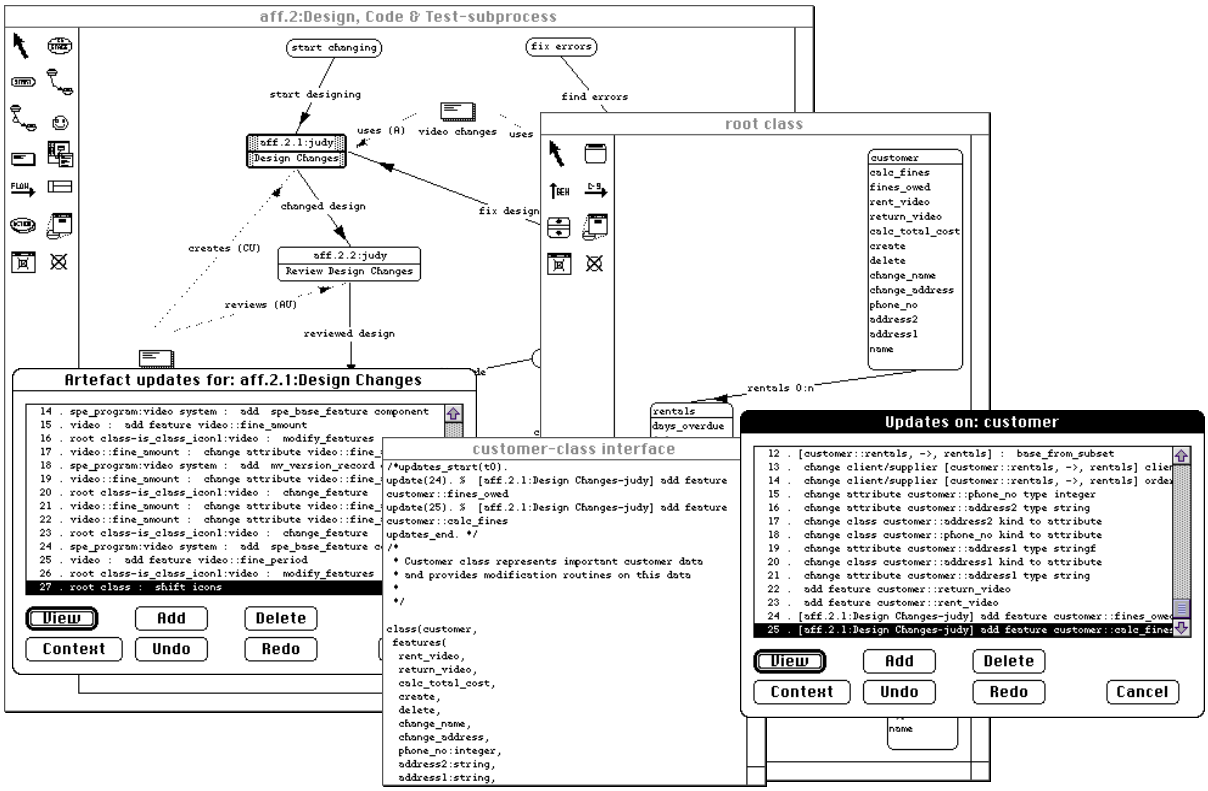

Figure 5. An example of work context capture and presentation in SPE–Serendipity.

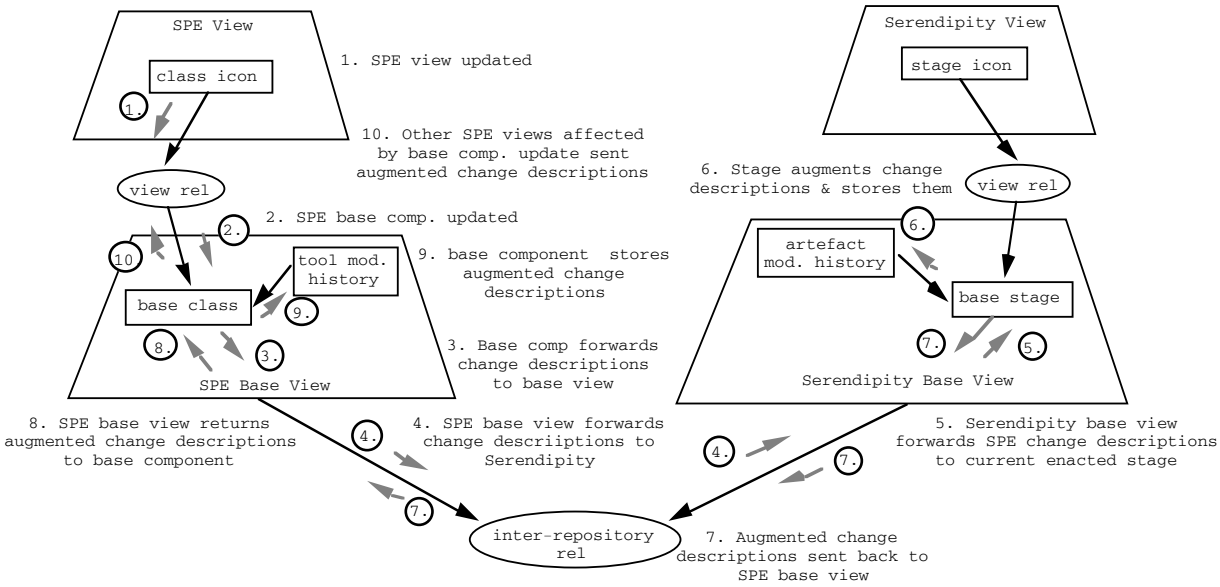

Figure 6. Integrating SPE and Serendipity environments: flow of change descriptions

When a software or view component is updated, a *change description* is generated. This is of the form UpdateKind(UpdatedComponent, ...UpdateKind-specific Values...). For example, an attribute update on Comp1 of attribute Name is represented as: update(`Comp1, Name, OldValue, NewValue`). All basic graph editing operations generate change descriptions and pass them to the propagation system. Change descriptions are propagated to all related components that are dependent upon the updated component's state. Dependents interpret these change descriptions and possibly modify their own state, producing further change descriptions. This change description propagation mechanism supports a diverse range of software development environment facilities, including attribute recalculation, multiple views with flexible, bi-directional textual and graphical view consistency, a generic undo/redo mechanism, component versioning, and collaborative view editing [10]. New software components and editing tools are constructed by reusing abstractions provided by an object-oriented framework. ISDE developers specialise MViews classes to define software components, views and editing tools to produce the new environment. A persistent object store is used to store component and view data.

SPE and Serendipity have been integrated by modifying MViews so that it sends change descriptions generated by tools to Serendipity, and by extending Serendipity to handle these "artefact update" events. Relationships between the Serendipity base view and SPE base view translate events (in the form of change descriptions) from one environment into appropriate events in the other. Figure 6 shows graphically how the integration mechanism works, with an update of an SPE view leading to a sequence of change description propagations between SPE and Serendipity.

This solution to SPE–Serendipity integration works for any other MViews ISDE. We have integrated OOEER (an EER/OOA modeller) [11], NIAMER (an ER/NIAM modeller), MViewsDP (a dialog painter) [10], and MVNotes [1] with Serendipity using a similar approach.

The C-MViews collaborative editing extensions to MViews [9] implement SPE and Serndipity's low-level synchronous, semi-synchronous and asynchronous view editing capabilities. The text chat facility was built as another, small MViews tool and integrated with Serendipity by extending a default MViews menu to allow access to this tool's facilities. Most of the process model visualisation and work context awareness facilities were built using Serendipity's own event process filters and actions.

We have also integrated Serendipity with non-MViews office automation programs, such as Microsoft Word™, Microsoft Excel™, Global/Fax™ and Eudora™ [12]. These office automation programs are launched and sent instructions via Apple Events, generated by Serendipity actions.

6. Other Work

Much research into Computer Supported Cooperative Work (CSCW) has focused on low-level interaction mechanisms, such as synchronous and asynchronous editing. Examples include most Groupware systems [5], GroupKit [27], Mjølner [22], C-MViews [9], and Rendezvous [16]. These systems lack information about the "work context" that changes have been carried out in, which Serendipity describes via work process models. Because of Serendipity's tight integration with its CSCW facilities and work tools, high-level information about work contexts is incorporated with other CSCW support.

Some work has been done on providing higher-level process modelling and coordination facilities, such as workflow configuration in Action Workflow [24], TeamWARE Flow [31], and VPL [30], software process protocols in ConversationBuilder [18] and Oz [3], and various kinds of shared workspace awareness in GroupKit [27]. Most of these tools do not currently use their high level descriptions of work contexts to augment descriptions of changes, highlight artefacts, tools and workflow/process model items, or augment traditional lower-level CSCW capabilities as does Serendipity. Serendipity's notion of a work context is similar to the idea of locales embodied in wOrlds [4], although we use an informal notion of work context which is user-defined. Obligations in ConversationBuilder and wOrlds may be thought of as similar to Serendipity filter/actions. However we claim our approach provides a higher-level of graphical expression for users, and is generally more flexible, able to be driven by all kinds of process model, tool or artefact update events.

Process-Centred Environments (PCEs) define the processes used to construct software systems. Examples include Merlin [26], Marvel [2], Oz [3], CPCE [21], and EPOS [17]. Computer-Aided Method Engineering (CAME) tools support evolution of the methodologies and notations used for software development. Examples include Decamerone [15] and MethodBase [28]. Many of these environments use complex, textual descriptions of processes and tool configurations which are difficult to understand and modify, often can not be modified while in use, and have poor exception handling. Some provide high-level, graphical process modelling tools but most lack arbitrary event handling (from either the process model or work artefacts). A few provide triggering mechanisms which carry out actions based on some event. Rule-based PCEs provide complex, textual rules which specify constraints over process models, which are often difficult for users to understand and modify. This is particularly true in process modelling domains such as office automation, where end-users do not understand triggering or rule-based languages. A key design goal of Serendipity was to make process models and event handling mechanisms graphical and widely accessible, and to provide a range of both low-level and high-level CSCW capabilities to assist cooperative work in large systems.

Many workflow tools and PCEs are not well integrated with existing tools used to perform work [20, 23]. They thus can not achieve the same degree of utilisation of high-level workflow and process model information to augment work tool and low-level CSCW capabilities as supported by Serendipity. Some work, including ConversationBuilder [18], MultiviewMerlin [23], wOrlds [4], and Oz [32], attempts to bridge the gap between workflow/PCEs and CSCW. So far these have had limited success, due to continuing problems of integrating existing tools into the environments and the limitations of process modelling tools used [23, 4, 32].

7. Discussion and Conclusions

We have used Serendipity as a stand-alone work process modelling and enactment tool, integrated it with other MViews environments to facilitate collaborative software development, and integrated it with existing Office Automation applications to support coordinated use of these tools [12, 13, 14]. Serendipity has been used to model small-to-medium process models and work plans, with the largest so far developed having over 200 process stages, artefacts, tools and roles, with over 60 process model and filter/action views. Collaborative process modelling and enactment support allows groups of workers to coordinate their work using both low-level editing and communication facilities and higher-level work context awareness. Experience to date with using Serendipity has indicated it provides very abstract, and yet flexible, process modelling and enactment support, and its CSCW facilities aid collaborators in more effectively using both the process models and the tools they coordinate [12, 13]. Some of the high-level awareness capabilities use process model views which can consume much screen real estate. Users can choose to hide these views, using the other facilities to remain aware of collaborators' work contexts. We have found that leaving enacted process model views visible behind other windows is useful when using other multi-view tools.

We are continuing to develop Serendipity by incorporating further CSCW capabilities, such as telepointers, multiple scrollbars, and improved collaborative editing support. We have recently been implementing additional work context determination and visualisation facilities using filter/actions to produce "intelligent agents" to perform these tasks. We are also developing work context awareness widgets to incorporate in the views of environments integrated with Serendipity.

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] 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.

[3] 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.

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

[5] Ellis, C.A., Gibbs, S.J., and Rein, G.L., "Groupware: Some Issues and Experiences," *Communications of the ACM*, vol. 34, no. 1, 38-58, January 1991.

[6] Grundy, J.C. and Hosking, J.G., "A framework for building visusal programming environments," in *Proceedings of the 1993 IEEE Symposium on Visual Languages,* IEEE CS Press, 1993, pp. 220-224.

[7] Grundy, J.C. and Hosking, J.G., "Constructing Integrated Software Development Environments with Dependency Graphs," Working Paper, Department of Computer Science, University of Waikato, 1995.

[8] Grundy, J.C., Hosking, J.G., Fenwick, S., and Mugridge, W.B., Chapter 11 in *Visual Object-Oriented Programming*. Manning/Prentice-Hall, 1995.

[9] Grundy, J.C., Mugridge, W.B., Hosking, J.G., and Amor, R., "Support for Collaborative, Integrated Software Development," in *Proceeding of the 7th Conference on Software Engineering Environments,* IEEE CS Press, April 5-7 1995, pp. 84-94.

[10] Grundy, J.C., Hosking, J.G., and Mugridge, W.B., "Supporting flexible consistency management via discrete change description propagation," to appear in *Software - Practice and Experience*

[11] Grundy, J.C. and Venable, J.R., "Providing Integrated Support for Multiple Development Notations," in *Proceedings of CAiSE'95,* LNCS 932, Springer-Verlag, Finland, June 1995, pp. 255-268.

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

[13] Grundy, J.C., "Supporting flexible collaborative software development with SPE–Serendipity," Working Paper, Department of Computer Science, University of Waikato, 1996.

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

[15] Harmsen, F., and Brinkkemper, S., "Design and Implementation of a Method Base Management System for a Situational CASE Environment," in *Proceedings of the 2nd Asia-Pacific Software Engineering Conference* IEEE CS Press, Brisbane, December 1995, pp. 430-438.

[16] Hill, R.D., Brinck, T., Rohall, S.L., Patterson, J.F., and Wilner, W., "The Rendezvous Architecture and Language for Constructing Multi-User Applications," *ACM Trans. on Computer-Human Interaction*, 1994.

[17] Jaccheri, L., Larsen, J., and Conradi, R., "Software Process Modeling and Evolution in EPOS," in *Proc. Fourth International Conference on Software Engineering and Knowledge Engineering (SEKE),* Capri, Italy, June 1992, pp. 17-29, .

[18] Kaplan, S.M., Tolone, W.J., Bogia, D.P., and Bignoli, C., "Flexible, Active Support for Collaborative Work with ConversationBuilder," in *1992 ACM Conference on Computer-Supported Cooperative Work,* ACM Press, 1992, pp. 378-385.

[19] Kellner, M.I., Feiler, P.H., Finkelstein, A., Katayama, T., Osterweil, L.J., Penedo, M.H., and Rombach, H.D., "Software Process Modelling Example Problem," in *Procs. of the 6th International Software Process Workshop,* IEEE CS Press, Hokkaido, Japan, 1990.

[20] Krishnamurthy, B. and Hill, M., "CSCW'94 Workshop to Explore Relationships between Research in Computer Supported Cooperative Work & Software Process," in *Procs. of CSCW'94,* ACM Press, April 1995, pp. 34-35.

[21] 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.

[22] Magnusson, B., Asklund, U., and Minör, S., "Fine-grained Revision Control for Collaborative Software Development ," in *Procs. of the1993 ACM SIGSOFT Conference on Foundations of Software Engineering,* Los Angeles CA, December 1993, pp. 7-10.

[23] Marlin, C., Peuschel, B., McCarthy, M., and Harvey, J., "MultiView-Merlin: An Experiment in Tool Integration," in *Procs. of 6th Conference on Software Engineering Environments,* IEEE CS Press, 1993.

[24] 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.

[25] Nitto, E.D. and Fuggetta, A., "Integrating process technology and CSCW," in *Proceedings of IV European Workshop on Software Process Technology,* LNCS, Springer-Verlag, Leiden, The Nederlands, April 1995.

[26] Peuschel, B., Schäfer, W., and Wolf, S., "A knowledge-based software development environment supporting cooperative work," *Int. J. of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, 76-106, 1992.

[27] Roseman, M. and Greenberg, S., "Building Real Time Groupware with GroupKit, A Groupware Toolkit ," *ACM Transactions on Computer-Human Interaction*, .

[28] Saeki, M., Iguchi, K., and Wen-yin, K., "A Meta-model for representing software specification and design methods," in *Proceedings of the IFIP WG8.1 Conference on Information Systems Development,* Prakash, N., Rolland, C., and Pernici, B., Como, 1993.

[29] Swenson, K.D., "A Visual Language to Describe Collaborative Work," in *Procs. of 1993 Symposium on Visual Languages,* IEEE CS Press, 1993, pp. 298-303.

[30] 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, .

[31] TeamWARE Inc., *TeamWARE Flow*, 1996.

[32] Valetto, G. and Kaiser, G.E., "Enveloping Sophisticated Tools into Computer-Aided Software Engineering Environments," in *IEEE Seventh International Workshop on Computer-Aided Software Engineering,* July 1995, pp. 40-48.