In Proceedings of the 2nd International Workshop on End-User Development, Barcelona, Spain, June 1997.

Support for End-User Specification of Work Coordination in Workflow Systems

John C. Grundy†, John G. Hosking†† and 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
{rick, john}@cs.auckland.ac.nz

Abstract

Workflow systems are a common example of end-user programming, in the form of work process modelling, enactment and evolution. Example applications of workflow systems include office automation, software process modelling and business process codification. We describe a workflow system which provides novel end-user programming support for the coordinated use of workflow models, communication and collaboration between multiple people, and the tools being used to perform work. We illustrate the utility of these work coordination mechanisms and outline our future research directions in this area.

1. Introduction

Workflow management systems and process modelling environments have become popular for use in various fields, including office automation, business process reengineering and software development [1, 28, 32]. Such systems typically allow users to model work processes using graphical and textual notations, and then run ("enact") these process models to guide (or enforce) work on particular projects. Workflow models are often reused on different projects and refined (improved) over time.

Workflow systems are typically used in conjunction with other tools for performing work, such as document editors, information systems and software development tools [1, 28]. Often multiple people collaborative on a project and thus use these tools and the workflows [32]. This requires coordinating the work done with both the tools for performing work and the workflow tools. Often the work coordination approaches, the workflow models and the work tools used on a project, evolve over time. Thus end-users of workflows require support for not only modelling, enacting and evolving the workflows themselves, but support for coordinating their work with others and integrating a variety of different tools into their work processes. Unfortunately most existing workflow and process modelling systems do not provide sufficient support for specification of work coordination mechanisms by end-users, or provide overly-complex facilities which many end-users can not use.

We describe a workflow/process modelling environment we have developed which supports the specification of simple and complex work coordination schemes by end-users. This uses a graphical notation to describe both workflow (work process) models and the handling of events to support work coordination and tool integration. We illustrate the utility of our approach for specifying various work coordination activities, compare its support to other workflow/process modelling systems, and briefly describe our future research directions.

2. Serendipity

Serendipity is a process modelling, enactment and work planning environment, which also supports flexible event handling, group communication, and group awareness facilities [12, 13]. Serendipity's notations are designed to be high-level and graphical in nature, and its coordination and rule mechanisms are easily extended by end-users. The notation is based on Swenson's Visual Planning Language [31] 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 [13]. 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, labelled with the finishing state of the stage the flow is from. There are a number of specialised types of stage including start, finish, AND, and OR stages. 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. Serendipity supports artefact, tool and role modelling for processes. Usage connections show how stages, artefacts, tools and roles are used. Optional annotations indicate: data is created (C), accessed (A), updated (U), or deleted (D); whether a stage must use only tools, artefacts or roles defined (√); and whether a stage cannot use specified tools, artefacts or roles (¬).

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). Complex filters and actions can be implemented by either reusing other, simple filter/action models, or by using a textual API interface from Serendipity to its underlying implementation language [12].

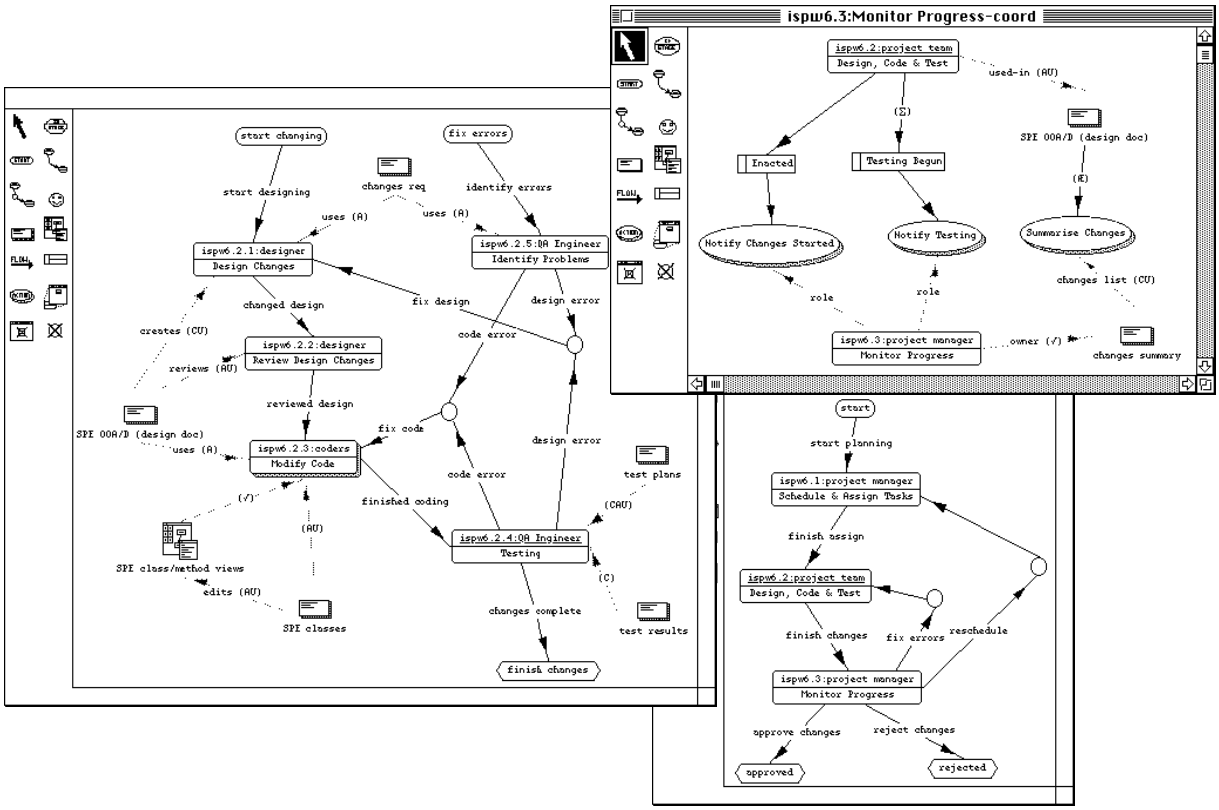

Figure 1. Part of the ISPW6 software process example modelled in Serendipity.

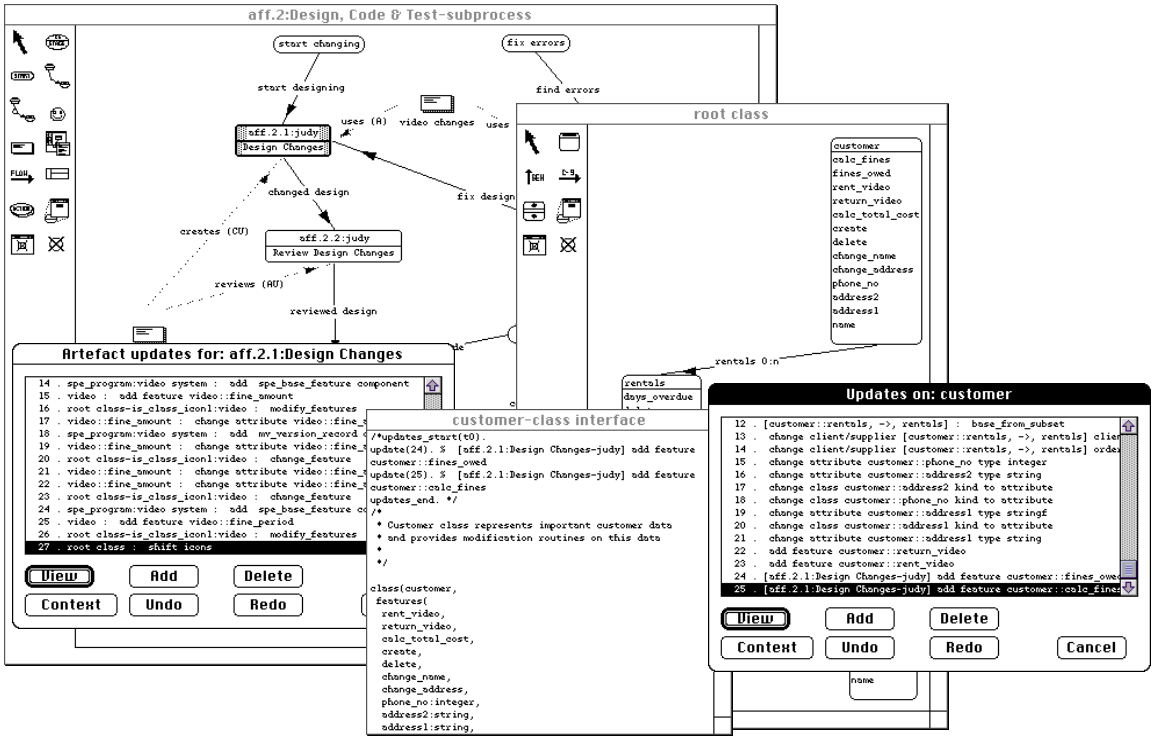

Figure 2. An example of work context capture and presentation in SPE–Serendipity.

Serendipity has been integrated with software and information system development tools [12, 13, 14], office automation tools [12], and a variety of CSCW tools [12, 13]. Figure 2 shows Serendipity being used with an integrated software development environment, called SPE [9, 13]. The Serendipity ISPW6 process model from Figure 1 has been tailored for use on this particular project and guides multiple developers' work on the software. Enacted parts of process models are shown by shaded icons and event flows. Changes made in the software development tools stored against process stages, and process stage information augmenting artefact update descriptions in the software tools.

Figure 3 shows Serendipity being used to support process modelling for a suite of office automation programs, including Macintosh versions of Microsoft Word™, Microsoft Excel™, the GlobalFax™ fax/OCR application, and the Eudora™ email utility [12]. Two simple Serendipity process models describing the processes of ordering and receiving stock for an organisation. Actions are used to launch the programs and to create, open, and save files via Apple Events sent from Serendipity to the running programs.

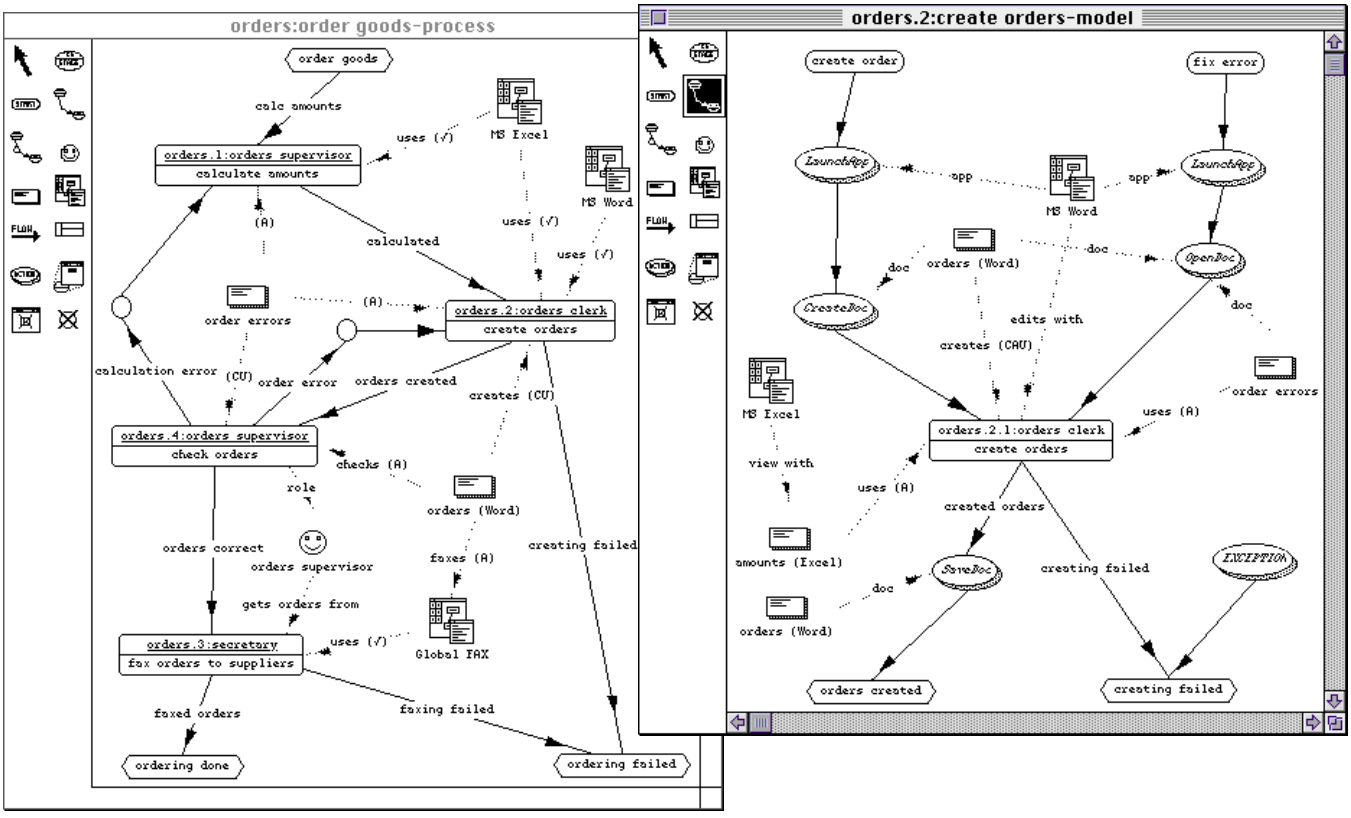

Figure 3. A simple office automation process described in Serendipity.

4. Workflow Development

End-users can readily model, enact, reuse and improve Serendipity process models, due to their primarily graphical nature [12, 13]. For example, the process models in Figure 4 were reused from the ISPW6 software process model template shown earlier. This reused model is to guide the work to be done for modification of a video library program in SPE. The idea is to add a fines facility to the software, and have Serendipity guide the work done in SPE and record all changes made to the software.

The project manager has copied the ISPW6 template and changed the process stage prefixes to "aff" ("Add Fines Facility"). The abstract names for developers have been changed to those who are filling these roles for this particular project using this workflow template. A project-specific "work plan" for coder "john" has also been developed by the project leader (bottom right window). Other end-users can also develop and modify their own plans, and the group can collaboratively modify any part of the workflow model as a whole. Changes to this project-specific workflow model can be abstracted back into the reusable template to improve it [13, 12].

Our visual language for workflow modelling allows both novice and experienced end-users to understand, reuse and modify both simple and complex workflow systems. Detailed information can be specified about a model using textual forms and views, but most information about the work processes being modelled are captured in the graphical diagrams. We have used Serendipity to model a range of small to medium work process models in a variety of application domains. These include software process models [12, 13], method engineering for information systems development [14], and business processes to coordinate use of office automation tools [12].

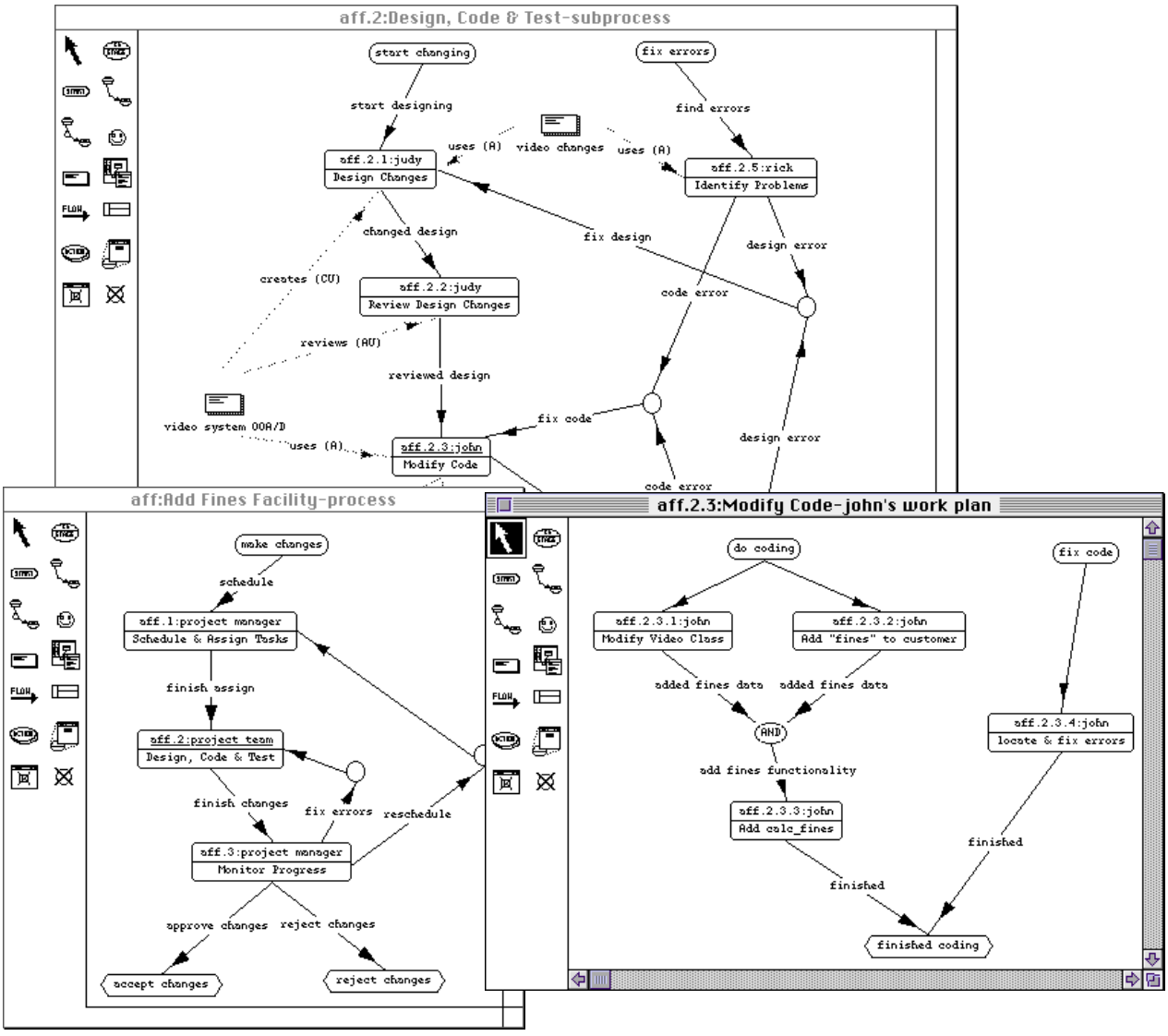

Figure 4. A reused template and expanded work plan for a particular project.

5. Work Coordination

Figure 1 showed an abstract work coordination view built using Serendipity filter/actions to keep the project manager informed of progress on a project using this workflow model. Various other work coordination schemes can be developed using our filter/action language to support simple or complex work coordination using workflow models and integrated tools.

For example, in Figure 5 coder “john” has defined a new event-handling view in Serendipity and added filter/actions to keep him aware of modifications done by “judy” on the artefact “video class” (the artefact changes are stored in a “change list” artefact), and to inform him when “rick” starts work on “test_unit”. In this example, changes judy makes are stored and thus john would browse this modification history at a later date, or have a dialogue showing these changes automatically updated for him. In contrast, when rick starts testing, john is informed immediately by a broadcast message.

Our visual event handling language has proved to be useful and accessible for novice users of Serendipity, allowing them to build simple work coordination schemes. It has also proved to be powerful enough for experienced users to build reusable, complex event handling models to implement a range of diverse work coordination facilities [12].

Figure 6 shows an example of a simple filter/action model to highlight icons in Serendipity views, to indicate access/update of artefacts and use of tools. A more complex form of this model has been used to highlight the background view. Other users' enacted stages are highlighted, in addition to the user who is being shown this view (judy). Artefacts and tools in use by judy and her collaborators are highlighted. This was achieved by storing recent artefact modification and tool events, and filter/actions use this history to highlight appropriate icons to support group awareness. End-users select the highlighting filter/actions they require and attach these to artefacts modelling Serendipity views and tools. End-users can modify these template filter/actions to suit their own awareness needs, due to the accessibility and power of expression of our visual event handling language. Highlighting and constraint filter/actions can also be applied to other tools integrated with Serendipity, as described in the following section.

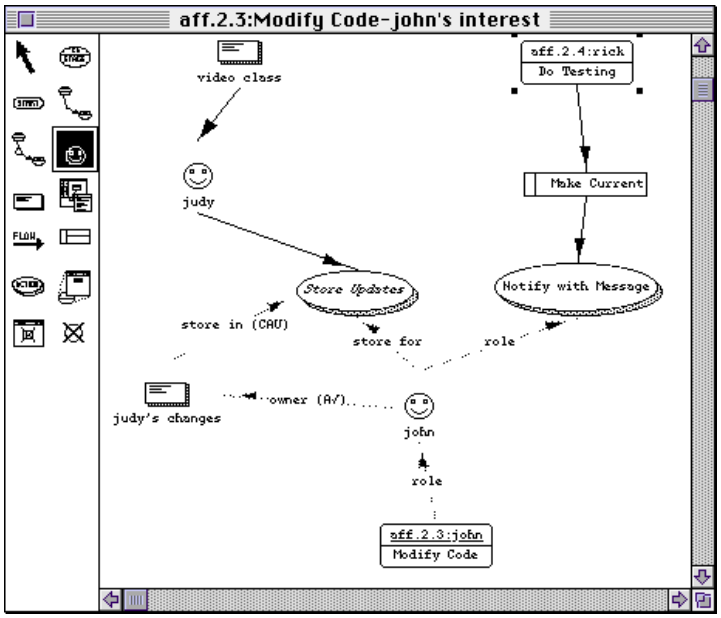

Figure 5. Extra actions specified to keep people aware of other's work.

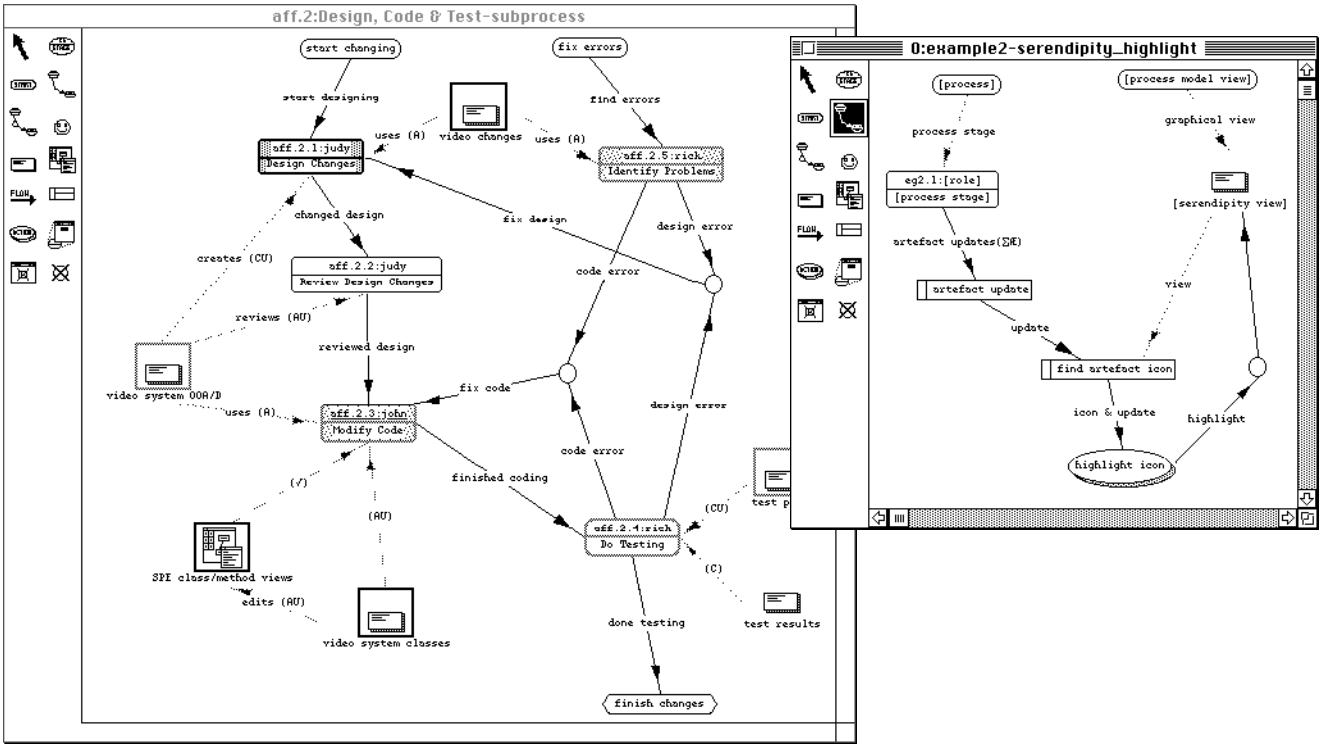

Figure 6. Highlighting icons to support awareness.

6. Supporting Tool Integration

Tools ranging from those built with the same underlying architecture as Serendipity to third-party tools can be integrated with the workflow environment, with differing levels of integration completeness. SPE, other software and information system development tools, and a variety of small Computer-Supported Cooperative Work (CSCW) tools have been tightly integrated with Serendipity [12, 13, 14]. Figure 7 shows how integrated messaging and annotation tools, as well as SPE, can be integrated with Serendipity and utilise its process model information. Descriptions of changes in SPE are augmented with enacted process stage information to help better describe why and when changes were made in SPE (examples of this are also shown in Figure 2). The messaging and annotation tools are sent Serendipity process model information which is incorporated into the sent messages and stored notes respectively.

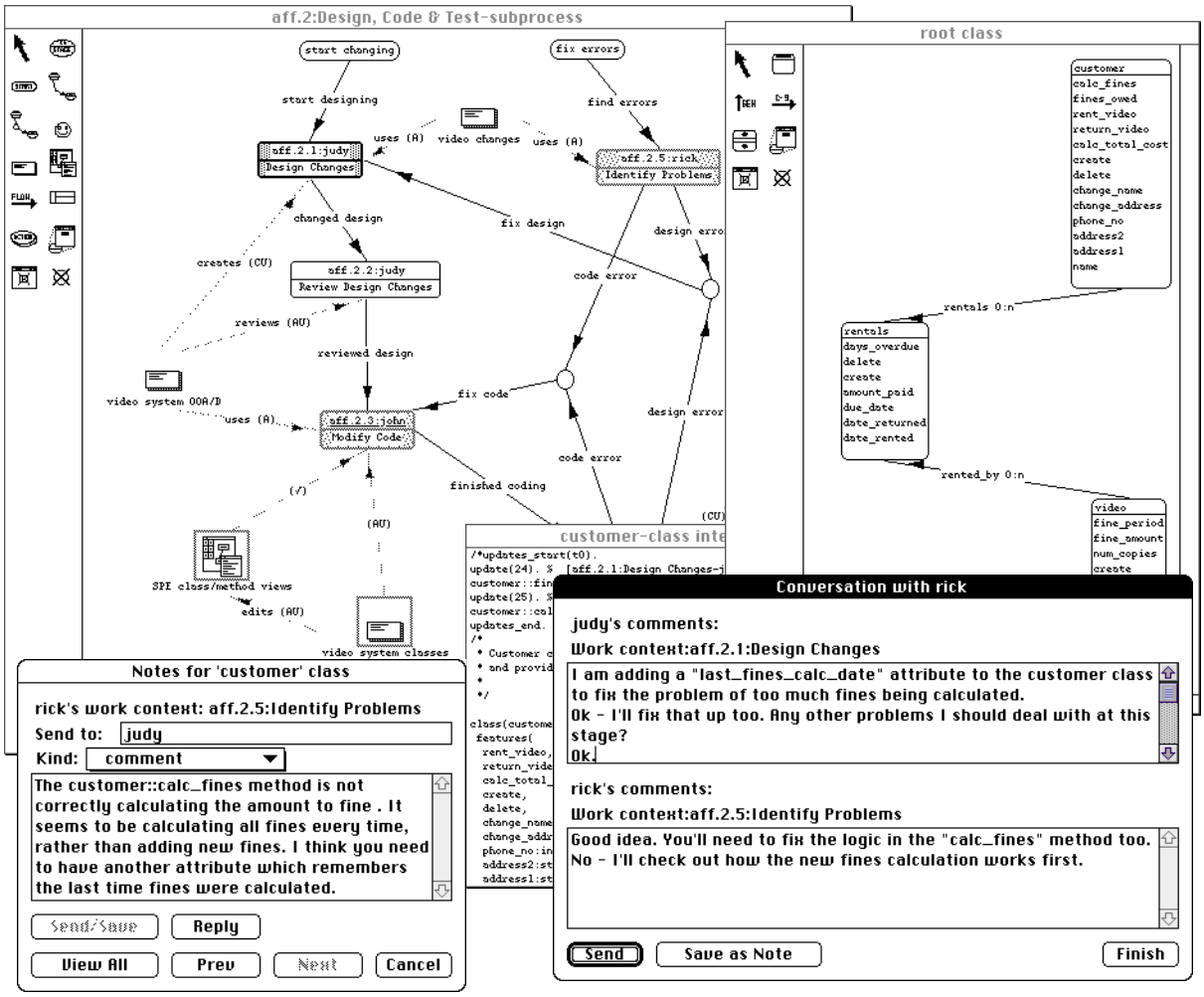

Figure 7. Examples of work context awareness and communication support.

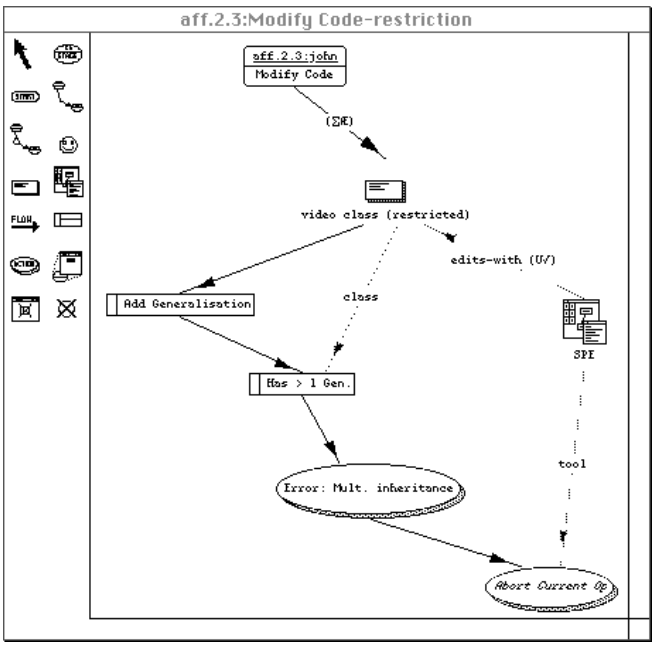

Figure 8. Constraining integrated tools with Serendipity filter/action models.

With tightly integrated tools like SPE, Serendipity filter/actions can even be used to constrain the behaviour of the tools. Figure 8 illustrates how SPE tool actions may be restricted by a software process model, illustrating a simple form of Method Engineering [19]. The video class is prevented from using multiple inheritance, thus extending the semantics of SPE itself by using Serendipity filters and actions to process SPE artefact modification events. We have also built considerably more

complex filter/actions and workflow models to support more comprehensive Method Engineering for a range of Information Systems Engineering tools [14].

Third-party tools, like those shown in Figure 3 can not be constrained or integrated to the same degree as our tools built with an open message-passing architecture [15, 16]. However, we have built up useful office automation system workflows and work coordination mechanisms, as illustrated in the Figure 3 example. Simple communication with these tools is supported by a range of template filter/actions to invoke tools, tell tools to open/close/save files, and to tell tools to exit. We have also developed filters which detect when tools exit or when they open, close and update files, generating events that end-user defined filter/action models can make use of.

7. Related Work

Much research into Computer Supported Cooperative Work (CSCW) systems has focused on low-level interaction mechanisms, such as synchronous and asynchronous editing. Examples include most Groupware systems [7], GroupKit [30], Mjølner [26], C-MViews [10], and Rendezvous [20]. These systems generally lack support for end-user tailoring of work coordination mechanisms.

Some work has been done on providing higher-level process modelling and coordination facilities, such as workflow configuration in Action Workflow [28], TeamWARE Flow [33], and VPL [32], software process protocols in ConversationBuilder [23] and Oz [4], and various kinds of shared workspace awareness in GroupKit [30]. Once again, these approaches generally do not provide the level of end-user configuration of work coordination and awareness mechanisms required in large workflow-based systems. Most workflow management systems, such as Action Workflow [28], Regatta [32] and TeamFLOW [33], provide a limited range of "interesting events", supporting interaction with other tools and some forms of notification and work coordination based on event occurrence. These are usually codified using a form-based approach where modellers specify simple actions to carry out based on a range of possible events. Our visual event handling language provides a much wider range of capabilities, but remains accessible to both novice and expert end-users.

Many Process-Centred Environments (PCEs), such as SPADE [1], Merlin [29], Marvel [2], Oz [4], CPCE [25], and EPOS [22], use complex, textual descriptions of processes and tool configurations which are often very difficult for end-users to understand and modify, often can not be modified while in use, and have poor exception handling. The event-handling of most graphical process modelling languages is codified graphically for "enactment" events (state transitions), but textually for stage guards and actions, and for coding interaction with people or other tools. ProcessWEAVER [8] provides a textual co-shell language which allows users to specify actions for process model nodes when fired by input tokens. SLANG [1] uses textual specifications of guards and actions for nodes in a state transition network. Marvel and Oz [4] use guarded rules specified over data types. Adele [3] provides an activity manager, which uses a textual language to specify database-related event handling for process models. These approaches all use textual languages, and hence all but expert end-users find such specifications difficult to understand, modify and reuse.

Visual dataflow-based languages, such as Fabrik [21] and Prograph [6], provide graphical dataflow models which are similar in nature to our event-handling language, but use dataflow rather than event-flow, which is less appropriate in a workflow and tool integration domain. Some visual languages, such as ViTABaL [11], utilise an event-driven model but lack the equivalent of Serendipity's filters, actions, and interest specification capabilities. Because of their general-purpose nature, these visual programming languages lack specific workflow modelling capabilities, and thus can not express and represent process model event-handling and work coordination tasks as effectively as Serendipity's languages.

Many workflow tools and PCEs are not well integrated with existing tools used to perform work [1, 24, 27]. 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 SPADE/ImagineDesk [1], ConversationBuilder [23], MultiviewMerlin [27], wOrlds [5], and Oz [34], attempts to bridge the gap between workflow/PCEs and CSCW. So far these have had some success, but there are continuing problems of integrating existing tools into the environments, effectively utilising workflow/process model information in the integrated tools, and with limitations of the process modelling tools used [5, 27, 34].

8. Future Research Directions

We have had considerable success with using our workflow and event-handling notations in small and medium-sized software process, method engineering and office information system applications. Both novice and expert end-users can utilise the notations to configure their work coordination schemes and to make use of other tools from within their workflow models. We are continuing to improve our notations, adding improved modelling capabilities for expressing function calling and for expressing specialised forms of event filtering, tool repository querying and better supporting tool integration. We are also making compilation improvements to the efficiency of the filter/action models, which are currently interpreted, so they can be more extensively used in large workflow-based systems.

Serendipity is currently implemented using the MViews Prolog-based architecture for building multi-view, multi-user editing tools [17]. We are currently porting Serendipity to JViews, a Java-based successor to MViews [18]. The filter/action language from Serendipity has been reimplemented as a stand-alone tool for processing events from JViews components, and is part of

the JComposer componentware environment for building JViews-based systems [18]. Parts of these filter/action models can be compiled and thus can execute much more quickly than the Serendipity versions, making them more powerful for a wider range of end-user programming tasks.

Figure 9 shows a running Entity-Relationship modeller built using JViews, and our new event-handling language being used to provide work coordination support. The visualised JViews components (top left window) can be used in an event-handling view (top right window) to specify the presentation of events of interest to a user (in this case the user of this tool). The bottom-right window shows our visual query language being used to query the tool repository (in this case to find all entities with a specified name which have links to at least one relationship).

We are continuing to develop JComposer and its support tools, and will utilise the JComposer event handling language and visual query language in a port of Serendipity. We are also building components to provide better integration support for third-party tools, based on the Java Beans componentware API. This will allow end-users more control over a wide variety of tools used with JComposer and Serendipity, and thus support better end-user specification of work coordination mechanisms.

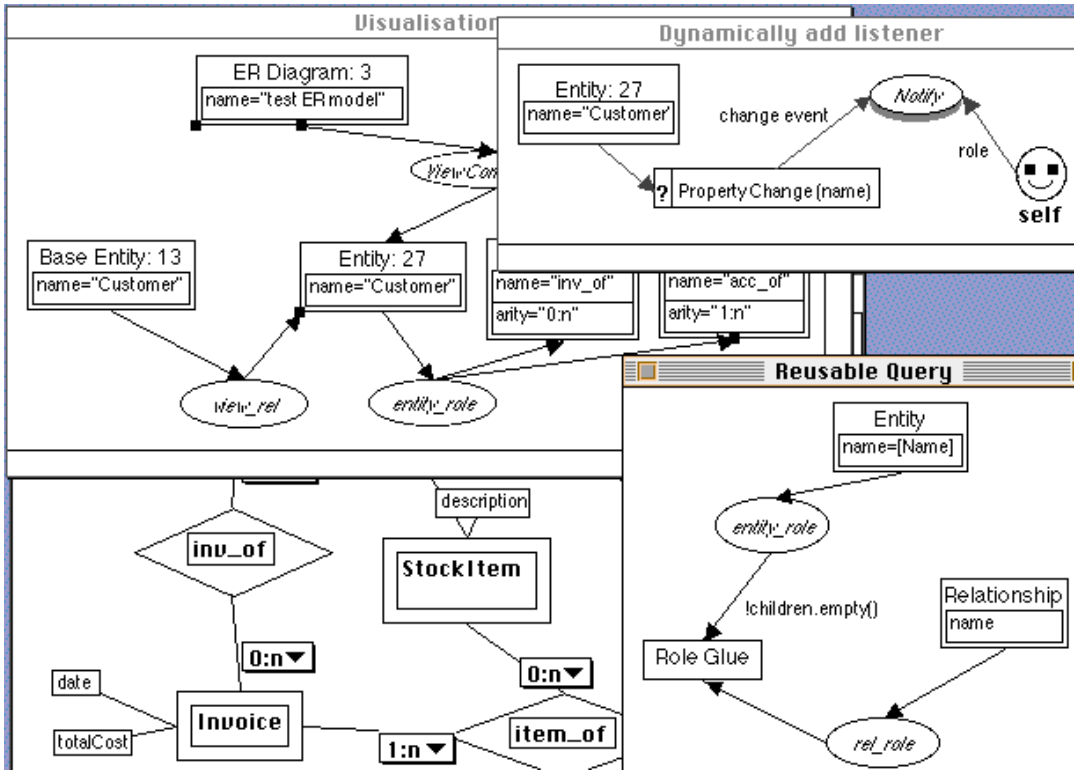

Figure 9. Visualising running tool data and extending tool event handling with JComposer.

9. Summary

We have described visual languages and a support environment for the end-user specification of work coordination mechanisms is workflow systems. Our notations allow users to readily model their work processes, to specify both simple and complex event-handling for these models, and to integrate a variety of tools for both performing work and communicating with other users into their workflow models. We have had success using these end-user programming techniques in a variety of application domains, including software process modelling, software development, method engineering, CSCW applications and office automation systems. We have ported a modified form of our event-handling language to Java, and are continuing to redevelop our workflow, CSCW and software development tools. This Java Bens-based implementation will allow our techniques for work coordination specification and tool integration to be more effectively used with wider range of third-party tools in complex application domains.

References

[1] Bandinelli, S., DiNitto, E., and Fuggetta, A., "Supporting cooperation in the SPADE-1 environment," *IEEE Transactions on Software Engineering*, vol. 22, no. 12, December 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] Belkhatir, N., Estublier, J., and Melo, W.L., *The Adele/Tempo Experience*, Software Process Modelling & Technology. Finlelstein, A., Kramer, J. and Nuseibeh, B. Eds, Research Studies Press, 1994.

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

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

[6] Cox, P.T., Giles, F.R., and Pietrzykowski, T., "Prograph: a step towards liberating programming from textual conditioning, , IEEE Computer Society Press," in *Proceedings of the 1989 IEEE Workshop on Visual Languages,* 1989, pp. 150-156.

[7] 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.

[8] Fernström, C., "ProcessWEAVER: Adding process support to UNIX," in *2nd International Conference on the Software Process: Continuous Software Process Improvement,* IEEE CS Press, Berlin, Germany, February 1993, pp. 12-26.

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

[10] 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.

[11] Grundy, J.C. and Hosking, J.G., "ViTABaL: A Visual Language Supporting Design By Tool Abstraction," in *Proceedings of the 1995 IEEE Symposium on Visual Languages,* IEEE CS Press, Darmsdart, Germany, September 1995, pp. 53-60.

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

[13] Grundy, J.C., Hosking, J.G., and Mugridge, W.B., "Low-level and high-level CSCW in the Serendipity process modelling environment," in *Proceedings of OZCHI'96,* IEEE CS Press, Hamilton, New Zealand, Nov 24-27 1996.

[14] Grundy, J.C. and Venable, J.R., "Towards an environment supporting integrated Method Engineering," Proceedings of the IFIP 8.1/8.2 Working Conference on Method Engineerig, Atlanta, August 26-28, Chapman-Hall, 1996.

[15] Grundy, J.C., Hosking, J.G., and Mugridge, W.B., "Supporting flexible consistency management via discrete change description propagation," *Software - Practice & Experience*, vol. 26, no. 9, 1053-1083, September 1996.

[16] Grundy, J.C., Hosking, J.G., Mugridge, W.B., and Amor, R.W., "Support for Constructing Environments with Multiple Views," in *Joint Proceedings of the SIGSOFT'96 Workshops,* ACM Press, San Francisco, October 14-15 1996, pp. 212-216.

[17] Grundy, J.C. and Hosking, J.G., "Constructing Integrated Software Development Environments with MViews," *International Journal of Applied Software Technology*, Vol 2, No. 3/4, 1996.

[18] Grundy, J.C., Mugridge, W.B., and Hosking, J.G., "A Java-based toolkit for the construction of multi-view editing systems," in *Proceedings of the Second Component Users Conference,* Munich, July 14-18, 1997.

[19] 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 (APSEC'95),* IEEE CS Press, Brisbane, December 1995, pp. 430-438.

[20] 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 Transactions on Computer-Human Interaction*, vol. 1, no. 2, 81-125, June 1994.

[21] Ingalls, D., Wallace, S., Chow, Y.Y., Ludolph, F., and Doyle, K., "Fabrik: A Visual Programming Environment," in *Proceedings of OOPSLA '88,* ACM Press, 1988, pp. 176-189.

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

[23] 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.

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

[25] Lonchamp, J., "CPCE: A Kernel for Building Flexible Collaborative Process-Centred Environments," in *Proceedings of the 7th Conference on Software Engineering Environments,* IEEE CS Press, 1995, pp. 95-105.

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

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

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

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

[30] Roseman, M. and Greenberg, S., "Building Real Time Groupware with GroupKit, A Groupware Toolkit," *ACM Transactions on Computer-Human Interaction*, vol. 3, no. 1, 1-37, March 1996.

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

[32] 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, 1994.

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

[34] 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.