Supporting Distributed Software Development Processes in a Web-based Environment

Yun Yang and *Paul Wojcieszak*
School of Computing and Mathematics
Deakin University
Geelong, Australia 3217
{yun,wojciesz}@deakin.edu.au

Abstract

Research into software development processes has been active in the software engineering community for more than a decade. However, given the exposure of the Internet and the Web, there is a significant impact on Web-based software development support which can be beneficial to many software project managers and developers. Our project focuses on investigation of an effective Web-based visual environment to support process modelling by software project managers and process enactment by software developers.

Our software process/project coordination is based on the innovative reactive system paradigm, determined by decision-making policies supported by sensors attached to process sub-tasks. However, with visual environment support, project managers need not master the underlying technology for process modelling. After modelling, the project can be enacted according to the schedule so that software developers can be visually coordinated in order to develop software either individually, i.e. asynchronously, or cooperatively, i.e. synchronously, by using various tools, in virtually shared workspaces.

1 Introduction

Process-centred software development environments have been viewed as the most recent generation of software development environments and process supported software engineering is by now a well-established research discipline [Avrilionis96]. However, there are still many open issues to be solved in a long run [Ambriola97, Fuggetta94, Sheth97]. In our project, as a whole, there are many issues to be investigated [Yang98a]. In this paper, we only focus on our Web-based environment for supporting software development processes with visual process modelling by project managers, and more importantly, process enactment by software developers. In particular, during enactment, software developers are effectively coordinated for various activities by the environment and synchronous cooperation is enabled by our complementary Web-based cooperative editor. Therefore, software developers involved in a project can work in asynchronously as well as synchronously shared workspaces.

Generally speaking, a process/project/task is normally composed of sub-tasks which are partially ordered [Feiler93]. How to manage sub-tasks, i.e. (sub)processes/steps, is the key issue for completion of the entire task. Hence, task-oriented technology is management-centred to facilitate project management focusing on coordination. With software support, software developers can be coordinated automatically by a system, which is normally more effective than merely managed manually by a human-being. This could allow for the cooperation among widely dispersed working groups, whose members may be in different organisations and different countries. For example, software developers may reside in Australia, Europe and North America. With around 8-hour time differences among locations, 24 hours a day working mode can be facilitated potentially [Gorton96]. Even if software developers are co-located in the same building, distributed software development is still desirable for various reasons such as automated coordination and information/tool sharing.

In addition to the heavy attention paid to software process modelling and enactment in the software process community, the most recent years show another trend that there is an emerging consensus that graphical process modelling would help to alleviate process modelling for non-language experts [Gruhn98], such as project managers. A practical environment should be designed to support visual programming to model the project without much knowledge of the underlying modelling language(s) and an easy-to-use graphical user interface for process enactment by software developers.

Nowadays, there is a growing interest to support cooperative work over the Internet (or Intranet) and the Web. The emergence and wide-spread adoption of the Web offers a great deal of potential for the development of collaborative technologies as an enabling infrastructure [Oreizy97]. In addition, the Java programming language, which has the capabilities of delivering applets over the Web as well as the slogan of "write once and run anywhere" - platform independence, has encouraged us to prototype our work in Java based on the Web environment [Yang99b]. In addition, no particular software needs to be installed for software developers regarding project management since Java applets can be downloaded on the fly and then run directly. Meanwhile, it has become popular recently with the component-based software solutions. Java matches the solutions quite smoothly. Furthermore, using combination of Web/Java seems better than using Web/CGI (common gateway interface) [Evans97] in terms of performance and control/data granularity. Therefore, we have treated the Web and Java as an excellent, if not ideal, vehicle to prototype our software process support mechanisms in a global distributed environment.

Our environment offers the following features which will be addressed in this paper. Briefly, we deploy an efficient semi-centralised multi-tiered client-server architecture to support Web-based software processes. The innovative underlying technology used for project coordination is the reactive system paradigm. For process modelling, the time frame is facilitated for better project management and the "divide and conquer" strategy is proposed to effectively model the process. For process enactment, software development sub-tasks are coordinated automatically where software developers can visually view the progress of the project and can use local tools as well as centralised tools such as our Web-based synchronous cooperative editor to develop software artifacts. This means that both asynchronously and synchronously shared workspaces are provided in our environment. The overall features offered by our environment are unique to other similar environments.

This paper is organised as follows. First of all, in Section 2, the background of our process support environment is described. Then the mechanisms for supporting visual process modelling are overviewed in Sections 3. Process coordination during enactment is addressed in Section 4 followed by the description of synchronous cooperation support in Section 5. Finally, conclusions and future work are drawn in Section 6.

2 Background of the environment

2.1 Reactive system paradigm

Over the last decade, process modelling, such as the rule based paradigm, has been investigated intensively which is assessed comprehensively in [Ambriola97]. We view software development process very much a reactive system. Reactive systems are characterised as owing much of their complexity to the intricate nature of reactions to discrete occurrences and the common notion imposed is the reactive behaviour [Harel85]. Extending [Harel85, Zhou98], we use the reactive system concepts to model three layers for the software process coordination control. At the bottom, the policy layer for making decisions relies on the middle mechanism layer for sensing and actuating application objects which are at the top application layer. With this architecture, if a top layer policy is changed, it may have no impact on related middle layer mechanisms and vice versa. We illustrate how the reactive system paradigm can effectively coordinate the project as follows.

As indicated in previous section, a process is composed of partially ordered sub-tasks. In the normal sense, the partial ordering implies that a sub-task should and can only start (including bypass etc.) when *all* its previous sub-tasks (i.e. the AND condition) have been *completely* finished (i.e. 100% completion rate). However, in reality, it may not be the case. For example, a sub-task may start when one of the two previous sub-tasks is finished (i.e. the OR condition). For another example, a sub-task may also start when the previous sub-tasks reach a certain threshold, say 80% completion rate. Certainly, there could be other (complex) conditions for invoking the execution of a sub-task. In other words, the coordination should be able to be supported by some fine-grained policies instead of very course-grained ones in most, if not all, existing process modelling paradigms. With our reactive system paradigm, for example, if the decision making condition based on two sensors was "AND" and is now "OR", and even the values for the sensor thresholds are changed, the sensors can still be used without any changes required, i.e. the mechanism layer can remain unchanged. These features offer the flexibility most other course-grained paradigms lack. Due to the space limit, details of the reactive system paradigm for process coordination cannot be discussed in depth in this paper.

2.2 Architecture

The variation of the semi-centralised multi-tiered client-server architecture of Web-based software development process support is depicted in Figure 1 [Yang98a]. It includes (1) clients as front-ends using local Web servers and tools, (2) centralised servers with tools, and (3) supporting tools such as databases and file systems as back-ends.

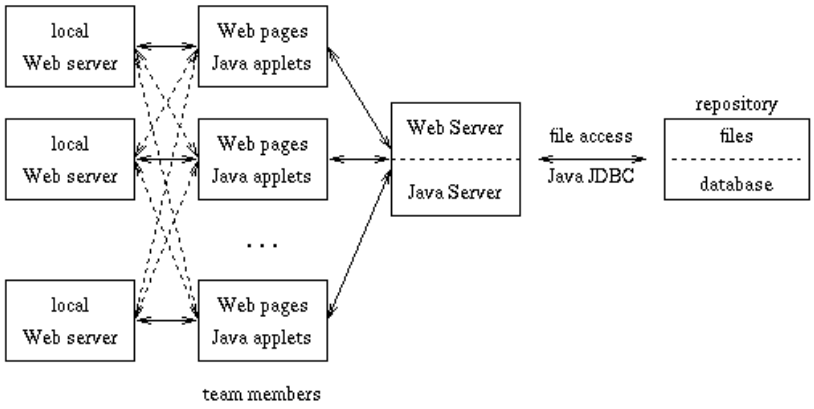

Figure 1. Architecture for supporting software development processes

The centralised server site plays the role for management of software development processes (or projects) and provision of some centralised tools. The process information resulted from modelling is stored in the database repository. Please note that the database repository is a general concept which can include various databases such as relational and object-oriented databases. During enactment, information such as documents can be stored locally at the client sites or at the server site and accessed by software developers (i.e. team members) based on the Web support which implies that information can be distributed rather than only centralised. Basically, for project coordination, at the client site, only an appropriate Web browser is required and no other particular software needs to be installed since Web pages and Java applets can be downloaded on-the-fly. Certainly, local tools can still be used for carrying out sub-tasks.

The centralised server site may also play a role for provision of some centralised tools. For example, to provide a shared workspace to allow dispersed software developers to collaborate synchronously, a real-time cooperative editor may need to be integrated with the software development process environment. In our environment, such a tool is available at the centralised server, again as a Java applet, to allow software developers to participate in a Web-based synchronous session. As described in this paper later on, such a session allow software developers to work on a same document together with electronic whiteboard support.

Based on this architecture, software development processes can be modeled first via a visual environment where process information is stored in the data repository. After that, software development can be carried out, as process execution or enactment. During enactment, a software development process is coordinated automatically. Meanwhile, centralised tools, such as the cooperative editor, and local tools can be utilised by software developers to conduct effective software development.

3 Overview of visual process modelling mechanisms

In this section, we illustrate the mechanisms of how a software process is modelled visually in our environment where some more details can be found in [Yang99a]. In general, given the exposure of graphical user interfaces, it is increasingly demanding and important to provide intuitive visual programming environments to support project managers to specify software development processes. In most cases, a project needs to be planned against some deadline which means that it is necessary to have a visualised time frame for project modelling. Figure 2 depicts our Java applet of the main layout for visual process modelling. This layout of the interface features the grids that form rows and columns. The numbers represented at the bottom of the grids are in fact the hours, days, months or years depending on the “mode” selected. The horizontal scrollbar is used if the manager wishes to scroll over the next consecutive hours, days, months or

years. The vertical scrollbar allows for parallel sub-tasks within the project. The idea behind the scrollbars is to ensure that the process modelling is not limited in any way which provides a full view of the project not restricted by the time or size constraints.

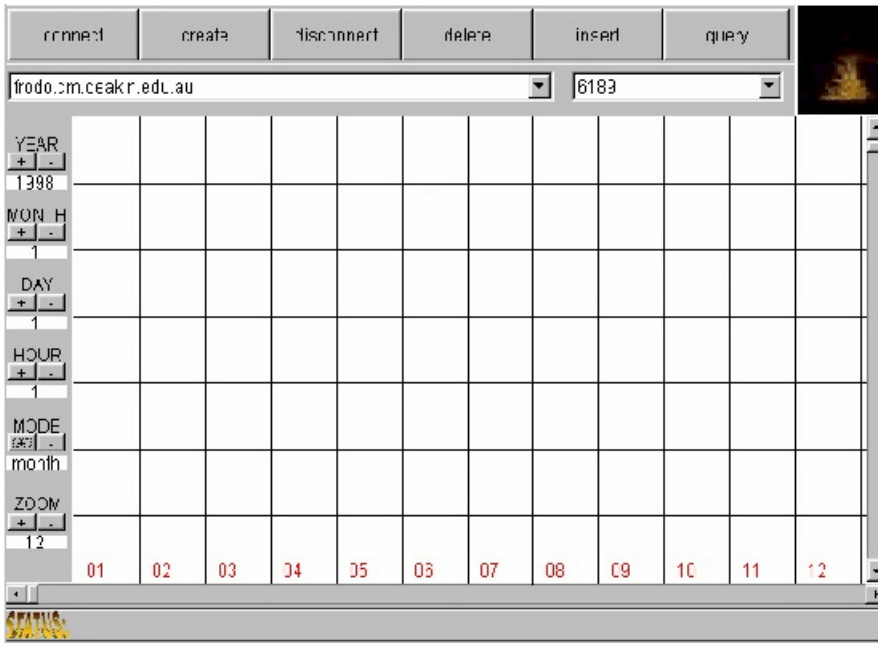

Figure 2: The main layout for process/project modelling

The project manager can model the process visually by drag-and-drop to create new sub-tasks and specify the ordering. At the moment, most existing environments only support "flat" bottom level modelling of a project. In reality, from the project manager's point of view, it is much more natural to model the project at a high level first and then to model further down step by step to the bottom level. This is the common sense of "divide and conquer" strategy for process modelling. The "zoom" button can enable the manager to zoom in or out. In addition, it is necessary to assign various artifacts/components to each sub-task which is very much form based. Those artifacts may include (constraint) resources like team members, documents, and hardware/software based on their availability with the certain dates/deadlines and instructions/messages for software developers. To support, tools can also be used for such as (semi-)automatic resource management [Yang97].

4 Process coordination during enactment

After a process is modeled by the project manager, it is ready to launch the project for coordinating software development. For process coordination in our environment, once the process is started, the most essential facility is that each software developer is provided a dynamic up-to-date *to-do* list. For example, as depicted in Figure 3, the "integration" task is on the to-do list for that particular person. There are practically two basic strategies for the to-do list notification: active and passive, which are both used. The active notification strategy is to send emails to appropriate software developers to notify the new to-do lists whilst the passive way is to get the to-do lists refreshed when a software developer is connected to the process environment, via the Java applet shown in Figure 3. There could be other information to be passed on such as instructions/messages for the work to be done and sensitive indicators for deadlines. In general, software developers normally do not rely much on the centralised server because they mainly work on the client side locally to carry out the tasks assigned.

Software developers can use local tools, or tools available in the software process environment, to carry out the work. Sometimes, tools can and need to be specified in the process. For instance, some tasks may involve several software developers to cooperate at the same time, hence a centralised Web-based cooperative editing tool described in the next section may be better specified to allow developers to use as a shared workspace. Even some local tools such as a single-user editor for individual developers can also be indicated to enable automatic tool invocation.

From the information/data exchange point of view, data can be either stored locally or at the server side which can then be easily accessed across the Internet with some simple and extensible standards, such as HTTP, based on the Web support. In addition, the richness of data/object types, such as multimedia, can be achieved. To manage data exchange in a software development project, most data types such as documents are specified during the process modelling. In addition, messages from software developers during process enactment can be recorded and forwarded to other software developers for fine-tuning effective data exchange.

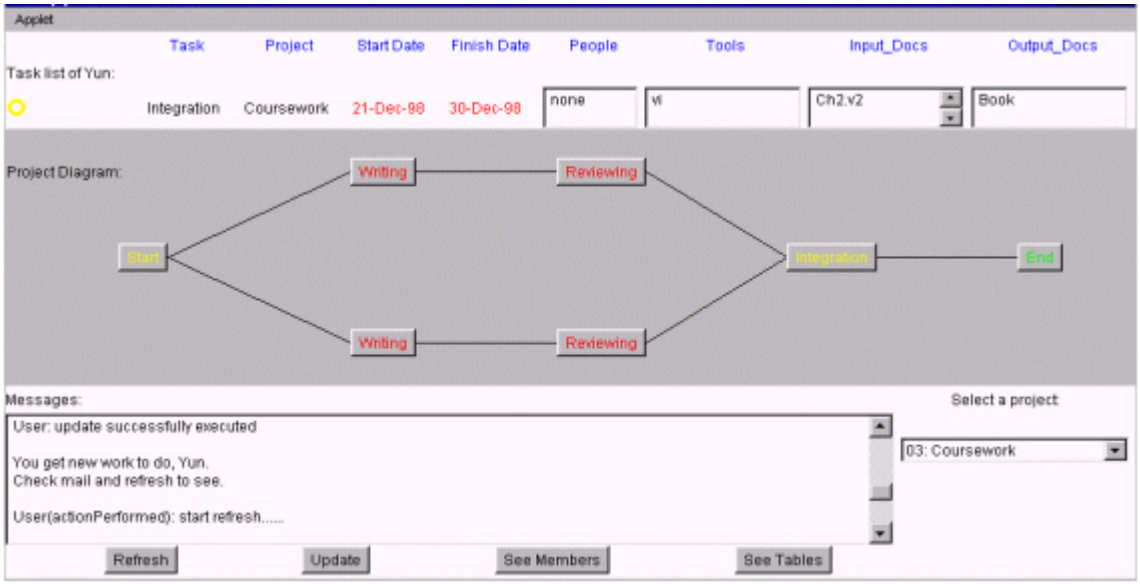

Figure 3. Process coordination user interface

When a certain sub-task is finished or a value of the sensor based on software developer's input is changed, a notification is sent to notify the process support environment over the server side. The coordination manager/daemon of the environment will utilise the decision making policy to generate new to-do lists for all affected software developers.

For a software developer working in a team environment, it is very useful to have a global view of the project in a visualised fashion in order to create a better teamwork atmosphere as shown in Figure 3. This is important from the psychological point of view when a person works in a computer-mediated teamwork environment. Different colours are used for status of each sub-task to indicate whether a sub-task is enacted, enacting or unenacted. The global project view is adjusted automatically whenever statuses of any sub-task is changed.

5 Synchronous team collaboration support

Process coordination described in the preceding section has involved various mechanisms for supporting shared workspaces in an asynchronous manner. In is common that most software development activities are undertaken by software developers individually. That means that most sub-tasks are carried out asynchronously, but interdependent, i.e. the outcome of a sub-task of one software developer is often the input to other sub-tasks of other software developers. However, some sub-tasks are shared, i.e. they involve software developers working together synchronously to complete the activity. Take a simplified typical scenario of the design stage, when there is more than one person involved in the design phase, the production of the design document including brain-storming becomes a cooperative activity to which some tool supporting co-authoring could be applied. The cooperative editor described in this section can serve well for the purpose for real-time co-authoring as well as brain-storming for software development.

Real-time distributed cooperative editing systems allow physically dispersed people to view and edit shared textual/graphical documents at the same time. They are very useful facilities in the rapidly expanding area of groupware and CSCW (Computer-Supported Cooperative Work) applications, such as electronic conference/meeting and collaborative documentation systems. Equally, they can be used in the software development context. Research into

cooperative editors has been a popular topic in the CSCW community since mid-80s and many papers have been published in various CSCW related conference proceedings and journals [Ellis91, Zhang94, Ressel96, Sun98].

The goal of our Web-based REDUCE (REal-time Distributed Unconstrained Cooperative Editing) research has been to investigate the principles and techniques underlying the construction of the REDUCE system with the following features [Yang98b]: (1) real-time - the response to local user actions should be quick (ideally as quick as a single-user editor without noticeable delay) and the latency for remote user actions should be low (ideally determined by external communication latency only); (2) distributed - cooperating users may reside on different machines connected by different communication networks; and (3) unconstrained - multiple users may concurrently and freely edit any part of the document at any time, in order to achieve free and natural information flow among cooperating users. The underlying technology for maintaining the consistency across different sites for unconstrained real-time cooperative editing is very complicated which has been comprehensively investigated by us in a text editing context [Sun98]. In this section, we only illustrate the functionality of the REDUCE prototype which is easily integrated with our software development process environment in order to provide a shared workspace for synchronous cooperation among software developers.

The screen snapshot in Figure 4 depicts a synchronous cooperation in action, again as a Java applet. The graphics canvas at the bottom plays a role of a whiteboard which enables software developer to free draw graphics, select and draw pre-defined shapes with optional fillings, or input text strings. The text editing panel on the top allows software developers to edit the document without any constrains, i.e. edit at any position of the text and at any time.

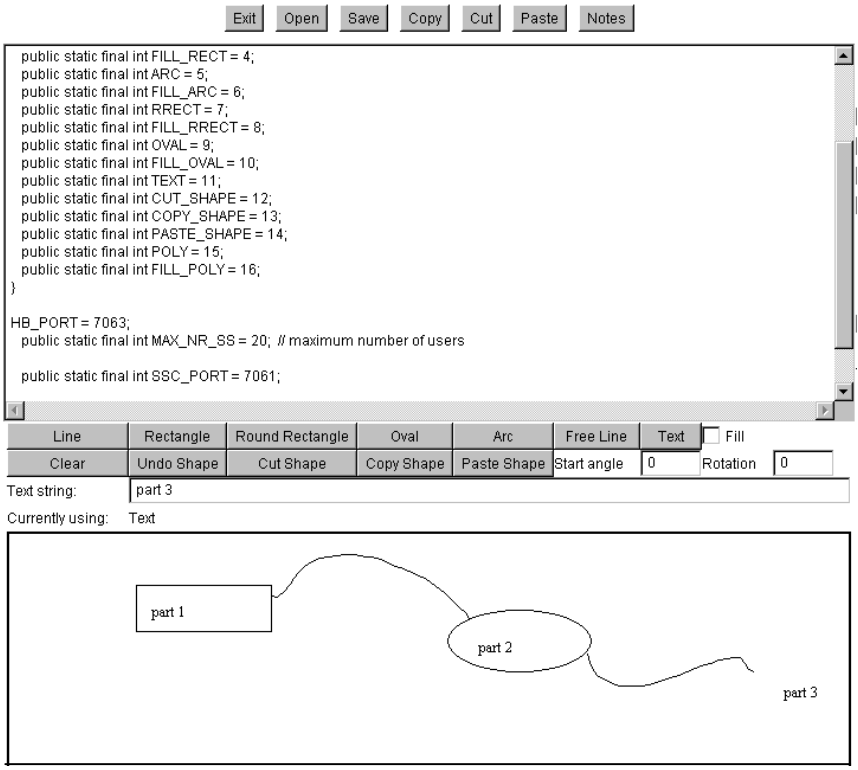

Figure 4. Cooperative editing supported by a whiteboard

6 Conclusions and future work

In this paper, we have addressed the importance of Web-based process support for software development. The paper has introduced the background of our software development process support environment which uses an effective semi-centralised client-server architecture and innovative modelling paradigm based on the reactive system concepts as the underlying technology for fine-grained process coordination control. The paper also has overviewed the mechanisms for realisling a visual environment to enable software project managers to model a software process. The other key features

of our work addressed in this paper are as follows. With the Web and Java support, our multi-tier client-server architecture is able to allow software development mainly done at the local client sides by software developers although coordination control is centralised at the server side. In addition, for software process coordination, no software installation is required and the environment is platform independent. During process enactment, software developers can be effectively coordinated by the dynamic to-do list with corresponding messages/instructions for carrying out the tasks assigned. Data exchange can be easily managed via storing at and accessing from either the local or the central sides. Moreover, software developers can enjoy the visualised workflow of the software development project and freely use both local and centralised tools, which can be specified in the process. In addition to share the workspaces in this asynchronous fashion enabled by the nature of coordination, synchronous cooperation is also enabled by such as our Web-based real-time unconstrained cooperative editor with a whiteboard for cooperative editing and/or brain-storming.

In the future, for the process support environment in general, many things can be further investigated for such as better process evolution, mobility, interoperability and tool integration. We also plan to deploy such as temporal logic to determine the hidden dependencies to effectively coordinate software processes. For visual programming support for Web-based process modelling, we need to keep improving the mechanisms for visualisation and the reactive system paradigm for the underlying technology. We are also keen to use an object-oriented database, instead of a relational database, for the data repository.

Acknowledgment

Work reported in the paper has been supported partially by a seeding grant from School of Computing and Mathematics, Deakin University in 1998 and an ARC (Australian Research Council) grant in 1999. We are grateful for some implementation support from Dengsheng Zhang and Phil Jeffers.

References:

[Ambriola97] V. Ambriola, R. Conradi and A. Fuggetta. Assessing process-centred software engineering environments. ACM Trans. On Software Engineering and Methodology, 6(3):283-328, 1997.

[Avrilionis96] D. Avrilionis, P-Y. Cunin, and C. Fernström. OPSIS: a view mechanism for software processes which supports their evolution and reuse. In Proc. of the 18th Int. Conf. on Software Engineering, pages 38-47, Berlin, Germany, Mar. 1996.

[Ellis91] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: some issues and experiences. CACM, 34(1):39-58, 1991.

[Evans97] E. Evans and D. Rogers. Using Java applets and CORBA for multi-user distributed applications. IEEE Internet Computing, 1(3):43-55, 1997.

[Feiler93] P. H. Feiler and W. S. Humphrey. Software development and enactment: concepts and definitions. In Proc. of 2nd Int. Conf. on Software Process, pages 28-40, Berlin, Feb. 1993.

[Fuggetta94] A. Fuggetta and C. Ghezzi. State of the art and open issues in process-centred software engineering environments. The Journal of Systems and Software, 26:53-60, 1994.

[Gorton96] I. Gorton and S. S. Motwani. Issues in cooperative software engineering using globally distributed teams. Information and software technology Journal, 38(10): 647-655, 1996.

[Gruhn98] V. Gruhn and J. Urbainczyk. Software modelling and enactment: an experience report related to problem tracking in an industrial project. In Proc. of 20th Int. Conf. on Software Engineering, pages 13-21, Kyoto, Japan, Apr. 1998.

[Harel85] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and models of concurrent systems*, Pages 477-498, Springer-Verlag, 1985.

[Oreizy97] P. Oreizy and G. Kaiser, The Web as enabling technology for software development and distribution, IEEE Internet Computing, 1(6):84—87, 1997.

[Ressel96] M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhauser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. In Proc. of ACM Conference on CSCW'96, pages 288-297, Nov. 1996.

[Sheth97] A. Sheth. Workflow and process automation in information systems: state-of-the-art and future directions. ACM SIGGROUP Bulletin, 18(1):23-24, 1997.

[Sun98] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems. ACM Transactions on Computer-Human Interaction, 5(1):63-108, Mar. 1998.

[Yang97] Y. Yang and Y. Ni, Resource management with trader for distributed software processes. Proc. of International Symposium on Future Software Technology'97, pp63-68, Xiamen, China, Oct. 1997.

[Yang98a] Y. Yang. Issues on supporting distributed software processes. Software Process Technology, Lecture Notes in Computer Science, Vol. 1487, pages 243-247, 1998.

[Yang98b] Y. Yang, C. Sun, Y. Zhang and X. Jia, A Web-based real-time cooperative editor in Java. Proc. of WebNet98 (World Conf. Of the Web, Internet and Intranet), pages 975-980, Orlando, USA, Nov. 1998.

[Yang99a] Y. Yang. Visual programming support for coordination of Web-based process modelling. Proc. of 11th Int. Conf. on Software Engineering and Knowledge Engineering, Kaiserslautern, Germany, June, 1999, in press.

[Yang99b] Y. Yang, Developing distributed software development Tools in Java on the Internet. Chinese Journal of Advanced Software Research, invited paper, Allerton Press, USA, 1999, to appear.

[Zhang94} Y. Zhang and Y. Yang. On operation synchronisation in cooperative editing environments. In IFIP Trans. on Business Process Re-engineering, A-54:635-644, 1994.

[Zhou98] W. Zhou and E. Eide. Java sensors and their applications. Australian Computer Science Communications, 20(1):345-356, 1998.