

Enabling Cost-Effective Light-Weight Disconnected Workflow for Web-based Teamwork Support

Yun Yang^{1,2}

¹Key Lab of Intelligent Computing & Signal Processing, Ministry of Education

Anhui University, Hefei, Anhui, P. R. China 230039

²CICEC – Centre for Internet Computing and E-Commerce
School of Information Technology

Swinburne University of Technology, John Street, Hawthorn,
Melbourne, Australia 3122

E-mail: yun@it.swin.edu.au

Abstract

Research into Web-based teamwork support using workflow has assumed that the network connection exists permanently, i.e. online. However, there is an increasing demand that workflow environments should also support the disconnected offline mobile scenario which is so far not well addressed. This paper identifies radical requirements, describes cost-effective design and implementation strategies, and demonstrates the prototype for team members to facilitate workflow in a light-weight disconnected mode in addition to that of the normal connected mode. The work enables teamwork support with smooth switching over between the connected online and disconnected offline mobile modes. The disconnected workflow offers the similar view, i.e. “look and feel”, and is platform independent to *users* as with the connected workflow. The outcome is achieved with an attractive fractional effort resulted from significant reuse by *developers* based on the existing connected Web-based workflow architecture, design and implementation.

Keywords: Applied workflow systems, Teamwork processes, CSCW, Mobile computing

1. Introduction

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 [Oreizy (1997)]. Due to the exposure of the Web and Internet, it becomes more and more common to support teamwork in a network connected mode. However, a glance around any airport terminals shows that notebook computers are pervasive among business travelers. Common usage of these computers is for tasks that require no interaction with outside resources, also referred to as disconnected operation [Roman (2000)]. This means that there is also an increasing demand that teamwork environments should also support the offline mobile scenario. Given the current trends in computing technology including the manufacturing of increasingly smaller, more powerful, and more portable computing devices, with appropriate application software support, team members should be able to work in the disconnected offline mobile mode, such as at home or during travels, as well as connected online mode as usual when they have network access, such as in offices.

Term workflow is to describe an (organizational) process that is executed and managed automatically by a computer system. A workflow system is an application that uses a computer representation of the workflow logic to define, manage, and execute the process. Unfortunately, automated workflow (support) systems do not yet adequately address information mobility or tasks disconnected from the rest of a (business) process [Bolcer (2000)]. In reality, support for disconnected mobile computing is becoming more and more important. Based on [Nixon (1998)], it is predicted conservatively that by 2003, 20-40 million workers will require mobile computer access. Hence, it is essential to research into this significant area which has not been sufficiently explored. In general, according to [Nixon (1998)], the challenges for mobile computing lie in three broad areas: 1) providing reliable wireless communication services; 2) building applications that deal with arbitrary disconnected nature of mobility; and 3) building applications that are not tied to fixed locations. Our work presented in this paper is related to the latter two challenges. This paper focuses on issues involved in enhancing

our existing Web-based connected online workflow in order to support the disconnected offline mobile working mode in a cost-effective manner.

Our work is unique in terms of that we have a flexibly extendable semi-centralised multi-tiered client-server architecture for workflow support, and more importantly, our extended architecture and the corresponding prototype support both connected and disconnected mobile modes for workflow very effectively. The cost saving is reflected by smooth extension and minimal change of the system architecture of an existing connected workflow to accommodate the demands of disconnected workflow. In addition, significant reuse is facilitated in developing a light-weight prototype of the disconnected workflow support.

In this paper, we identify radical requirements for supporting disconnected workflow first. We then address the corresponding cost-effective design strategies. After that, we describe the cost-saving implementation and illustrate the teamwork coordination enabled by our prototype in both connected and disconnected modes. Finally, we summarise the related work before we conclude.

2. Requirement analysis for supporting disconnected workflow

In this computing era, tasks are usually carried out by a cooperating team who may be physically dispersed by using various (software) tools. Systems for workflow, computer-mediated teamwork, groupware or computer-supported cooperative work (CSCW) offer various automatic support for team cooperation to increase the productivity. Generally speaking, a teamwork process is normally composed of tasks which are partially ordered [Feiler (1993)]. By partial ordering, it means that a task should and can only start when its previous tasks have been completed to a certain satisfactory level. How to manage tasks with workflow support is the key issue for completion of the entire teamwork process. Most existing workflow systems assume the continuous network connectivity. However, we also need to intensively investigate disconnected mobile scenarios in order to support workflow in both connected online and disconnected offline modes. In this section, we tend to identify the radical requirements as follows.

In general, according to [Dearle (1998)], there are three classes of mobile entities: users, views and platforms. A user is a person who uses a computer and may be mobile due to they move from home to the workplace, from city to city, and continent to continent. A view is what users see when they sit down at a display screen. A view is implemented by a platform which is defined as a collection of software and hardware. With these mobile entities, we believe that it is fundamental to have the following two requirements accordingly from the users' perspective:

- Every user needs to have a similar “look and feel” wherever the work needs to be done, i.e. a similar view is maintained across connected and disconnected modes. This will provide users a consistent work environment.
- Due to the mobility of users, they may work on different platforms. This requires that the system can easily run on all kinds of machines with various system configurations and yet offer a same or similar view.

Furthermore, we also have some other essential requirements with respect to workflow management support from system developers' perspective:

- Firstly, due to existence of many workflow systems which mainly support connected situations, it is unwise and costly to develop new workflow systems from scratch to accommodate the increasingly desired disconnected scenarios. Therefore, we need to investigate how to incorporate disconnected workflow support into existing workflow systems in a cost-effective manner.
- Secondly, there are many technical issues involved in workflow support like process modeling, resource management and process evolution for network connected workflow processes. Here we need to investigate the impact in the context of disconnected workflow. Hopefully, after the disconnected mode of workflow is introduced, no significant changes are required.
- Thirdly, smooth workflow enactment after introducing the disconnected mode needs to be maintained because tasks in a workflow process still need to be coordinated effectively. At the same time, certain consistency control is essential so that ideally no out-of-date input information (dirty data) is used and no output information is overwritten thereafter.

- Finally, some individual tasks may involve several team members to work together. For example, a synchronous cooperation task may require several users using a real-time tool. This kind of situation needs to be discussed.

3. Design strategies for supporting disconnected workflow

In accordance to the requirements identified in the previous section, we start with addressing the design strategies for workflow management support from the developers' perspective and then we come back to describe how to support the similar "look and feel" view and platform independence requirements from users' perspective.

There exist various system architectures to support teamwork such as centralised and decentralised. We use the semi-centralised multi-tiered client-server architecture for Web-based teamwork process support as depicted in Figure 1 which has the origin from [Yang (1998)]. In this section, we focus on our architecture to illustrate how to enable cost-effective disconnected workflow. It may be argued that selection of a particular architecture could result

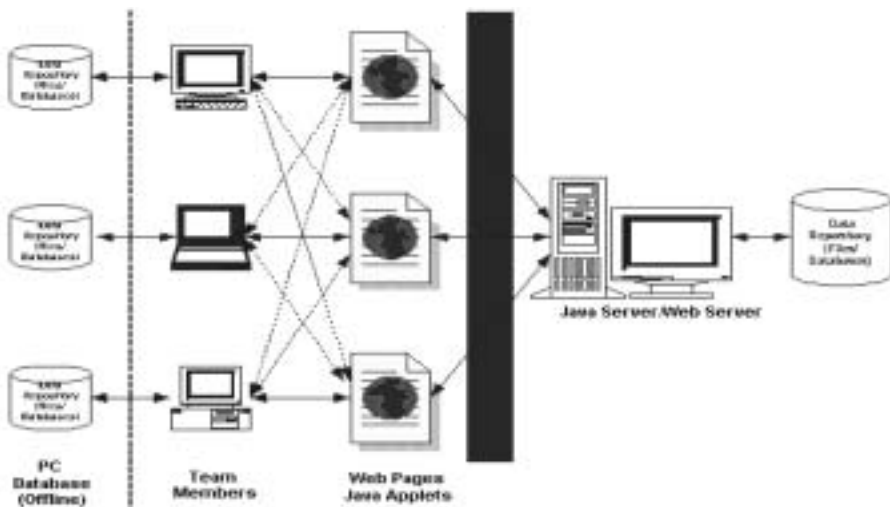


Figure 1. Architecture for supporting connected and disconnected workflow

in lack of generality which, in fact, is faced by any system architecture used. However, we believe our architecture does not jeopardise the generality. Our online teamwork support architecture includes (1) clients for team members as front-ends using Java applets, local Web servers and tools, (2) centralised servers with tools, and (3) supporting tools for data repository such as databases and file systems as back-ends. The left-hand side part of the vertical dotted line has been added to incorporate the disconnected workflow support which will be addressed in detail later in this section. The advantages of this kind of architecture are addressed below.

The centralised server site plays the role for management of teamwork processes based on a process engine, and for provision of some centralised tools such as synchronous cooperative editors. This kind of centralisation can reduce the teamwork coordination inconsistency in a Web-based environment dramatically. The process information resulted from modelling is stored in the back-end data repository. Please note that the data repository is a general concept which can include files and various databases such as relational and object-oriented databases. During process enactment, information such as files for documents can be stored locally at the client sites or at the server site and accessed by team members based on the Web support which implies that information can be distributed rather than only centralised. This makes the architecture semi-centralised.

For data repository, we have experimented with two types of databases [Yang (1999B)]: the Oracle relational database and the ObjectStore object-oriented database. The experimental results are in favour of deploying an object-oriented database to support process-centred teamwork. With a Java interface, such as that in ObjectStore, we only need to handle objects in Java directly without concerning mapping between objects in Java and tables in the (Oracle) relational database. In fact, it is more natural to carry out a process in the object-oriented manner which is another important reason that why we are in favour of using an object-oriented database as data repository. However, we still use Oracle as default data repository for the sake of wider availability.

With the online teamwork support architecture, network connection is mainly facilitated to exchange information among tasks, coordinate individual tasks and so on. Hence Web-based workflow systems can be treated as an excellent foundation to incorporate the needs of disconnected workflow. This is the start point to

motivate us to customise our Web-based workflow architecture to practically handle additional disconnected mobile teamwork support. Given this strategy, teamwork should still be coordinated by the online Web-based workflow support. Normally, in order to enable users to work in a disconnected environment from time to time, users should be able to check out tasks with all information needed, say onto their notebook computers. We assume that users can only check out the current enacting tasks, i.e. ongoing tasks on the to-do lists. After some work being done during the disconnected offline mobile mode, when they connect back to the network next time, i.e. online, users should be able to check in tasks with progress made so far. Once a (completed) task is checked in to inform the centralised server, normal coordination will occur as usual at the server side. If there are any new enacting tasks on the to-do lists, they can be checked out if necessary. We believe that it is a very reasonable assumption that only enacting tasks on the to-do lists can be checked out since users normally go online on a regular basis so as to carry out necessary check in and check out. This assumption can dramatically simplify the support required for enabling disconnected workflow since no coordination support is necessary in the disconnected mode – only the individual tasks need to be handled. In addition, it can also significantly reduce the complexity of consistency control since only the enacting tasks can be checked out so that simple locking is sufficient to prevent inconsistent check out.

Based on the above architecture design strategy, issues of workflow process modeling, resource management, process evolution and so forth can be dealt with as before, like that addressed in [Yang (2000A)], without changes. However, it is impossible to facilitate *online* multi-user cooperation, such as real-time cooperative editor REDUCE [Yang (2000B)], in the disconnected mode due to its nature. This can be considered as a limitation of disconnected workflow. Nevertheless, when it is necessary, users can still do it online with either a dedicated connection or a dial-up via the notebook, for instance.

To support the disconnected workflow, the online Web-based architecture only needs to be extended to some degree in a very natural fashion as reflected on the left-hand side of Figure 1. It is clearly a cost-effective design for enhancing Web-based architecture to incorporate disconnected workflow support, as

detailed above. In principle, the front-end needs to accommodate both online and offline teamwork support. In particular, the view saved during check out needs to be stored on a local computer using local data repository. The view saved is able to contain information needed to offer a similar “look and feel” for users since all information is available on the server. The authenticated check out facility would enable the connected mode to switch to the disconnected mobile mode and the authenticated check in facility would enable the disconnected mobile mode to switch back to the connected mode. Therefore, a light-weight workspace for disconnected workflow can be achieved.

In addition to make use of the benefits of the Web and Internet, 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”, i.e. platform independence, has encouraged us to prototype our work in Java based on the Web environment. By using Java, consequently, only an appropriate Web browser is required to work with the view saved to support disconnected workflow.

4. Cost-saving implementation

In implementing the workflow support prototype to support disconnected workflow designed in the previous section, we made full use of the existing Web-based online workflow prototype. The Web-based prototype was implemented in Java with a well designed object model. Hence, on one hand, the existing Web-based prototype was easily extended to accommodate the extra needs for supporting disconnected workflow for such as check out and check in functionalities. On the other hand, many existing Java classes were reused for developing the disconnected workflow prototype.

In our environment, the common attributes for a task include such as *process ID*, *task ID*, *task name*, *user ID*, *start date*, *end date*, *tools*, *input documents*, *output documents*, and *status*. A task may also have attributes like *user name*, *user email address* to enhance communication and coordination among team members. In order to fulfill a teamwork workflow, a process should be able to flow forward and the *status* attribute of tasks is for this purpose. A task can have one of the three statuses: *enacted*, *enacting*, *unenacted*.

By enacted task, we mean that the task is completed. By enacting task, we mean that the task is currently ongoing but has not been completed. By unenacted task, we mean that the task has not been invoked yet. With using relational databases as data repository, for example, in our online prototype, there are eight tables used to store persistent process information. These tables are designed below which play an important role in coordination of the workflow. Some further details can be found in [Yang (1999B)] which also has the design for using an object-oriented database. Any change or update (e.g. task *status* changed from *enacting* to *enacted*) during the execution of the process is recorded into the corresponding elementary tables in the database. The server is then requested to check information of each subsequent task in tables to decide if any task should be launched, i.e. changing *status* from *unenacted* to *enacting*, and any input (documents) should be available to the related members. All the necessary information are derived from those elementary tables such as that status and dependants are from tables TASKS and DEPENDANTS respectively. Similarly, information such as user email address and user ID are retrieved from table USERS.

Persistent Elementary Tables:

1 TASKS

Attributes: process_id task_id task_name start_date end_date status
 Data type: char char char date date number

2 TOOLS

Attributes: task_id tool_name
 Data type: char char

3 DEPENDANTS

Attributes: task_id (dependant) task_id
 Data type: char char

4 INPUTS

Attributes: task_id input_doc_name
 Data type: char char

5 OUTPUTS

Attributes: task_id output_doc_name
 Data type: char char

6 PROCESSES

Attributes: process_id process_name
 Data type: char char

7 PEOPLE

Attributes: task_id user_name
 Data type: char char

8 USERS

Attributes: user_id user_name password user_email
 Data type: char char char char

For disconnected local data repository, there are two general mechanisms available: the database system and the file system. Using a local database system was straight forward as it can be implemented in a similar way as in the existing online prototype. We used Java JDBC with the Access relational database from Microsoft for doing so with the above table structures and Java classes available in the existing online prototype. In addition, we also implemented the prototype with using the local file system due to less software requirements and hence more platform independence. Certainly, using a local file system demands some extra transformation which is tedious in general. Furthermore, we also experimented with the ObjectStore object-oriented database from Object Design. Three different data repositories provided an ideal platform for us to compare various implementation strategies. Dynamic configuration of the system is offered so that the users could select a suitable configuration if necessary. At this stage, the file system is the default configuration for data repository.

Generally speaking, given our design strategy, team members normally do not rely much on the centralised server because they mainly work on the client side to carry out individual tasks assigned. This creates the good opportunity for teamwork support in a disconnected offline mobile mode in addition to the normal connected online mode. In the disconnected mode, since there is no such a centralised server available (or necessary) like that in the connected mode, workflow tasks have to be carried out on an isolated computer. Hence how to support check out and check in features appropriately in order to enable team members to work offline in a similar manner as they do online is the key issue. For check out, a proper view of workflow in the online mode needs to be retained in the offline mobile mode. Since check out is carried out when online, the view derived contains sufficient information that can be retrieved from the centralised data repository. Similarly, all input information such as documents can also be saved in the view. The saved view can play the role of the virtual server to provide a to-do list for the team member and to work on tasks. The view offers a similar “look and feel” as in the online mode depicted in Figure 2 in the next section with the capability of allowing each team member to work alone. Certainly, in the disconnected mode, only local offline tools can be used for carrying out the tasks. For check in, again, it is carried out when online, all changes can be derived from the updated

offline view to enable the workflow to continue. Therefore, the workspace of disconnected workflow is light-weighted because it normally does not require any specific software and hardware support except a Java-enabled Web browser and a conventional file system as a minimum configuration.

5. A demonstration of teamwork enactment support

In this section, we describe the prototype which supports teamwork in both online and offline mobile modes. Over the last decade, process modeling has been investigated intensively as assessed comprehensively in [Ambriola (1997)]. However, in our case, process modelling and other related issues remain unchanged after introducing disconnected workflow support and hence they are not the topics for this paper. Interested readers are referred to our other publications [Yang (1999A), Yang (2000A)] for more information. In this paper, therefore, we only focus on descriptions of process enactment.

Suppose a process has been modelled by the teamwork manager, it is then ready to launch the process for teamwork coordination. For teamwork coordination in our environment, once the process is started, the most essential facility is that each team member is provided with a dynamic up-to-date *to-do* list. For example, as depicted in Figure 2, there is one task, indicated by a (yellow) circle

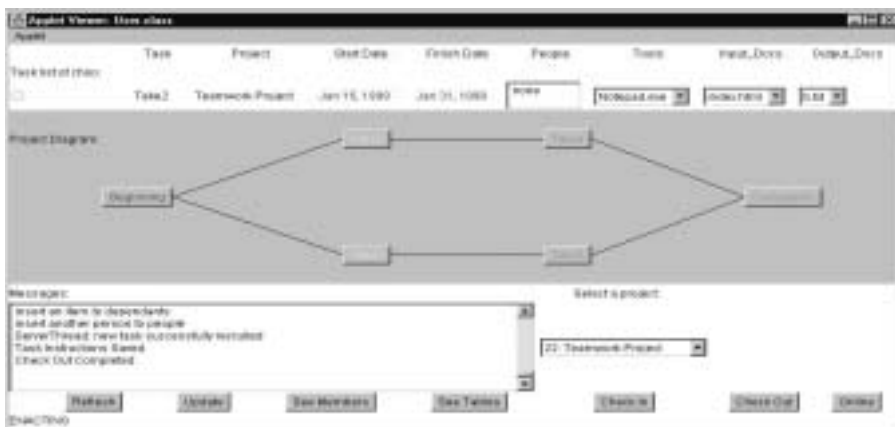


Figure 2. User interface for workflow enactment support

on the left above the workflow diagram, on the task list for that particular team member. There are practically two basic strategies for the to-do list notification: active and passive, which are all used. The active notification strategy is to send emails via JavaMail to appropriate team members to notify the new to-do lists. The passive way is to get the to-do lists refreshed on demand by team members. With notification, there could be other information to be passed on such as instructions for the work to be done and sensitive indicators for deadlines in the message box. At the same time, information is displayed on the top about such as the task, project (i.e. process), start/finish dates, input/output documents.

For a team member working in a workflow environment, it is very useful to have a global view of the process in a visualised fashion in order to create a better teamwork atmosphere as shown in the center part of Figure 2. This is particularly useful from the psychological point of view when a person works in a computer-mediated teamwork environment. Different colours are used for the status of each task to indicate whether a task is enacted, enacting or unenacted. The global view of the process is adjusted automatically whenever the status of any task is changed.

In the connected online mode, team members can use local tools, or tools available in the Web-based workflow environment, to carry out tasks. Sometimes, tools can and need to be specified in the process. For instance, some tasks may involve several team members to cooperate at the same time, hence a centralised Web-based cooperative editing tool may be better invoked automatically to allow team members to work on a shared workspace. Even some local tools such as a single-user editor for individual team members can also be specified 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. The richness of data/object types, such as multimedia, can also be achieved. To manage data exchange in a workflow process, most data types such as documents are specified during the process modelling. In addition, messages from team members during process enactment can be recorded and forwarded to other team members to fine tune effective data exchange.

In the disconnected offline mobile mode, the workspace is based on the view generated by the authenticated check out at the request

of team members by using the “check out” button as depicted in Figure 2. Once working offline, the user interface has the same “look and feel” as with Figure 2 except some buttons such as “check out” being disabled. With this view, team members can work normally as in the online mode described above, however, using local tools only. The work done can then be checked in when back to the online mode. The default data repository used for the view is the file system for the sake of the platform independence. However, the team member may select relational or object-oriented databases as data repository if the corresponding software is locally installed.

When a certain task is finished, for example, no matter via a connected or disconnected mode, a notification can and should be sent to the centralised server by clicking the “update” button when online to notify the workflow environment. The process engine (a daemon) of the environment will utilise the decision-making policy [Yang (2000C)] to generate updated to-do lists for all affected team members.

6. Related work

Teamwork support systems such as process-centred environments including workflow systems have been investigated in various communities such as software engineering, business engineering, information systems and CSCW (computer-supported cooperative work) for more than a decade. For example, process-centred software development environments have been viewed as a recent generation of software development environments and process supported software engineering is by now a well-established research discipline [Avrilionis (1996)]. Similarly, workflow systems have also been investigated intensively and quite a few commercial products are available [Sheth (1997)]. However, there are still many open issues to be solved in a long run [Fuggetta (1994), Ambriola (1997), Abbot (1994), Sheth (1997)].

When we look at process centred environments, on one hand, in the software process community, some typical examples are SPADE [Bandinelli (1996)], OzWeb [Kaiser (1998)], Serendipity-II [Grundy (1998)], and MILOS [Maurer (2000)]. On the other hand, in business applications, there exist some typical commercial systems like Staffware 2000 (<http://www.staffware.com>), FlowMark

(<http://www.software.ibm.com/ad/flowmark>), and Teamware Flow (<http://teamware.fujitsu.com.au/teamware/Products/>). In a Web-based environment, there exist quite a few other systems for teamwork support in addition to such as OZWeb, Serendipity-II, MILOS and Staffware 2000. In general, there are several categories listed next. The prototype developed by Scacchi and Noll [Scacchi97] takes an approach of using HTML forms and associated CGI scripts for process support. The workflow system investigated by Groiss and Eder [Groiss97] is based on using the standard email and HTTP to process information sent as HTML pages. Both of them are coarse-grained compared to the Web/Java approach. BSCW [Bentley95] is a Web-based system to support primarily a shared workplace in an asynchronous manner. The work implemented in Java by Ly [Ly97] is mainly for project management. All of the above only support connected workflow. The Magi architecture [Bolcer (2000)] is a good example to address mobile and disconnected workflow. However, it is heavy-weighted in terms of a (micro-Apache) HTTP server needed for supporting disconnected workflow. Our work has some unique features. For example, our semi-centralised multi-tiered client-server architecture for workflow support can be extended flexibly and our prototype supports both online and offline mobile modes for workflow in a very cost-effective manner. The cost is saved by smooth extension and minimal change of system architecture and significant reuse of the existing connected workflow to accommodate the demands of disconnected workflow. The outcome is an extended Web-based workflow environment and a light-weight prototype for disconnected workflow support.

7. Conclusions and Future Work

Given teamwork support by conventional connected workflow as well as the rapidly increasing demand of disconnected workflow, in this paper, we have addressed radical requirements from both users' and developers' perspectives for supporting workflow for process-centred teamwork in a disconnected offline mobile mode in addition to the normal connected online mode support. The corresponding cost-saving light-weighted design, implementation and prototype have also been described for supporting both connected

and disconnected mobile modes of workflow. The effective check in and check out features enable the similar “look and feel” view, i.e. the user interface, in the disconnected offline mobile mode as of the connected online mode. This smooth transfer of working between connected and disconnected modes opens up much more flexibility for teamwork in contrast to the current prevalent connected-only fashion. In the future, we need to work further on general issues like the infrastructure for Web-based workflow modeling and enactment. Many other research issues are also under investigation such as better process evolution, transactions, mobility with agent technology, interoperability with XML and component-based tool integration.

Acknowledgement

Work reported in the paper has been supported partially by ARC (Australian Research Council) grants in 1998 and 1999 and a visiting scholarship from Ministry of Education of China. I am grateful for some implementation support from L. E. Tan, E. N. Chia, L. C. Choo, P. Wojcieszak, D. Zhang, P. Jeffers, and D. Brain. I also thank reviewers for their constructive comments and suggestions for improvement.

References

- Abbot, K. and Sunil. S. (1994). Experiences with workflow management: issues for the next generation. In *Proc. of ACM CSCW'94*.
- Ambriola, V., Conradi, R. and Fuggetta, A. (1997). Assessing process-centred software engineering environments. *ACM Transactions on Software Engineering and Methodology* vol. 6(3),283-328.
- Avrilionis, D., Cunin, P-Y. and Fernström, C. (1996). OPSIS: a view mechanism for software processes which supports their evolution and reuse. In *Proc. of 18th Int. Conf. on Software Engineering*, Berlin, Germany, 38-47.
- Bandinelli, S., Di Nitto, E. and Fuggetta, A. (1998). Supporting cooperation in the SPADE-1 environment. *IEEE Trans. On Software Engineering* vol. 22(12),841-865.
- Bentley, R. et al. (1995). Supporting collaborative information sharing with the World Wide Web: the BSCW shared workplace system.

- In *Proc. of the 4th WWW Conference*, <http://www.w3.org/Conferences/WWW4/Papers/151/>.
- Bolcer, G. A. (2000). Magi: an architecture for mobile and disconnected workflow. *IEEE Internet Computing* (special issue on Internet-based workflow) vol. 4(3),46-54.
- Dearle, A. (1998). Toward ubiquitous environments for mobile users. *IEEE Internet Computing* (special issue on mobile computing) vol. 2(1),22-32.
- Feiler, P. H. and Humphrey, W. S. (1993). Software process development and enactment: concepts and definitions. In *Proc. of 2nd Int. Conf. on Software Process*, Berlin, Germany, 28-40.
- Fuggetta, A. and Ghezzi, C. (1994). State of the art and open issues in process-centred software engineering environments. *The Journal of Systems and Software* vol. 26,53-60.
- Groiss, H. and Eder, J. (1997). Workflow systems for inter-organizational business processes. *ACM SIGGROUP Bulletin* vol. 18(3),23-26.
- Grundy, J. C. et. al. (1998). A decentralised architecture for software process modelling and enactment. *IEEE Internet Computing* vol. 2(5),53-62.
- Kaiser, G. et al. (1998). WWW-based collaboration environments with distributed tool services. *World Wide Web Journal*, vol.1(1), 3-25.
- Ly, E. (1997). Distributed Java applets for project management on the Web. *IEEE Internet Computing* vol. 1(3),21-26.
- Maurer, F. et al. (2000). Merging project planning and Web-enabled dynamic workflow technologies. *IEEE Internet Computing* (special issue on Internet-based workflow) vol. 4(3),65-74.
- Nixon, P. and Cahill, V. (1998). Mobile computing: technologies for a disconnected society. *IEEE Internet Computing* (special issue on mobile computing) vol. 2(1),19-21.
- Oreizy, P. and Kaiser, G. (1997). The Web as enabling technology for software development and distribution, *IEEE Internet Computing* vol. 1(6),84-87.
- Roman, G-C., Picco, G. P. and Murphy, A. (2000). Software engineering for mobility: a roadmap, In *The Future of Software Engineering*, (A. Finkelstein, ed.). ACM Press, 241-258.
- Scacchi, W. and Noll, J. (1997). Process-driven Intranets: life-cycle support for process reengineering. *IEEE Internet Computing* vol. 1(5),42-49.

- Sheth, A. (1997). Workflow and process automation in information systems: state-of-the-art and future directions. *ACM SIGGROUP Bulletin* vol. 18(1),23-24.
- Yang, Y. (1998). Issues on supporting distributed software processes. In Proc. of 6th European Workshop on Software Process Technology. *Lecture Notes in Computer Science* vol. 1487:243-247.
- Yang, Y. and Wojcieszak, P. (1999A). Visual programming support for coordination of Web-based process modelling. In *Proc. of the 11th Int. Conf. on Software Engineering and Knowledge Engineering*, Kaiserslautern, Germany, 257-261.
- Yang, Y., Zhang, D. and Wojcieszak, P. (1999B). Coordination management with two types of databases in a Web-based cooperative system for teamwork. In *Proc. of 1st Int. Symp. on Database, Web and Cooperative Systems*, Baden-Baden, Germany, 47-52.
- Yang, Y. (2000A). An architecture and the related mechanisms for Web-based global cooperative teamwork support. *Int. Journal of Computing and Informatics* vol. 24(1),13-19.
- Yang, Y., Sun, C., Zhang, Y. and Jia, X. (2000B). Real-time cooperative editing on the Internet. *IEEE Internet Computing* vol. 4(3),18-25.
- Yang, Y. and Zhang, D. (2000C). Effective coordination for cooperation support in Web-based process-centred teamwork environments. In *Proc. of 3rd Asia Pacific Int. Web Conf.*, Xi'an, China, 323-327.