

# An Architecture and its Related Mechanisms for Web-based Global Cooperative Teamwork Support

Yun Yang

School of Information Technology, Swinburne University of Technology, Hawthorn, Australia 3122

Email: yun@it.swin.edu.au

**Keywords:** teamwork, CSCW, processes, Java, Web, visualisation, databases

*Given the exposure of the Internet and the Web, there is a significant impact on Web-based cooperative teamwork support which can be beneficial to many teamwork managers and normal team members who may be either computing or non-computing professionals. In this paper, we focus on our research into a more effective architecture for Web-based teamwork automation support and its corresponding innovative mechanisms for various perspectives including visual process modelling for teamwork managers and process enactment for team members in an asynchronous and synchronous manner. Our research prototype is implemented in Java and data repository used can be an object-oriented or relational database.*

## 1 Introduction

Teamwork is a key feature in any workplace organisation. In this computing era, a process/project is usually carried out by a cooperating team who may physically dispersed by using various (software) tools. Systems for computer-mediated teamwork, groupware, workflow or CSCW (computer-supported cooperative work) offer various automatic support for team cooperation to improve the productivity. Generally speaking, a process is normally composed of tasks which are partially ordered [Feiler93]. How to manage tasks is the key issue for completion of the entire process/project. With software support, team members are coordinated by a system which is clearly more effective than managed manually by a human being. In addition, team members may, for example, reside in Asia-Pacific, Europe and North America. With about 8-hour time differences among locations, 24 hours a day working mode can be facilitated potentially [Gorton96].

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", i.e. platform independence, has encouraged us to prototype our work in Java based on the Web environment. In this case, no particular software needs to be installed for team members regarding teamwork

coordination since Java applets can be downloaded on the fly and then run directly. 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 teamwork support mechanisms in a global distributed environment.

In this paper, we start with related work and then we focus on major issues involved in teamwork support. The topics we cover in this paper for supporting Web-based global teamwork include a dedicated architecture, visual teamwork modelling, enactment in an asynchronous fashion, enhanced by process evolution and dynamic resource management, as well as synchronous collaboration. Finally, the conclusions are drawn and the future directions are pointed out.

## 2 Related work

In a Web-based environment, there exist quite a few systems for teamwork support. 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. In Serendipity-II, Grundy et. al. use a decentralised architecture for process modelling and enactment [Grundy98]. In general, on one hand, most (process-centred) teamwork environments including the workflow systems focus on coordination support among partially ordered individual activities. On the other hand, most CSCW systems focus on collaboration support among the team members for some individual activities in the process.

Research into teamwork process support has been carried out for more than a decade and many fruitful outcomes have been achieved. However, there are still many open issues to be solved in a long run [Ambriola97, Fuggetta94, Sheth97]. Our work is unique as follows. Firstly, we have proposed an effective semi-centralised multi-tiered client-server architecture for teamwork support as explained in detail in this paper. Secondly, we have covered most inter-related areas to better reflect real world teamwork such as Web-based visualised global teamwork for asynchronous coordination with strong support of process evolution, resource management, as well as synchronous collaboration. Finally, we have offered many innovative mechanisms for teamwork support as addressed in individual sections in this paper when appropriate.

**3 Teamwork support architecture**

There exist various system architectures to support teamwork such as centralised and decentralised. We have chosen the semi-centralised multi-tiered client-server architecture for Web-based teamwork process support as depicted in Figure 1 [Yang98a]. This architecture 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. The advantages of this kind architecture is addressed below.

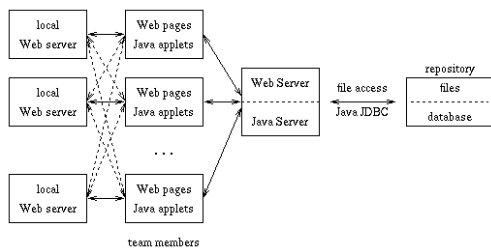


Figure 1. Architecture for supporting teamwork processes

The centralised server site plays the role for management of teamwork processes (or projects) 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 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 process enactment, information such as 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 architecture also offers flexibility to support teamwork in an offline mode in additions to the normal online mode. Basically, for process 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 tasks.

For data repository, we have experimented with two types of databases [Yang99b]: the Oracle relational database and the ObjectStore object-oriented database. The experiment 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.

**4 Supporting teamwork modelling**

Over the last decade, process modelling, such as the rule based paradigm, has been investigated intensively which is assessed comprehensively in [Ambriola97]. We view teamwork 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 [Harel90]. Extending [Harel90, Zhou98], we use the reactive system concepts to model three layers for the teamwork 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 paradigm, if a top layer policy is changed, it may have no impact on related middle layer mechanisms and vice versa. We illustrate next that how the reactive system paradigm can effectively coordinate the process.

As indicated earlier that a process is composed of partially ordered tasks. In the normal sense, the partial ordering implies that a task should and can only start (including bypass etc.) when *all* its previous 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 task may start when one of the two previous tasks is finished (i.e. the OR condition). For another example, a task may also start when the previous tasks reach a certain threshold, say a 80% completion rate. Certainly, there could be other (complex) conditions for invoking the execution of a 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 innovative 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, readers are referred to [Chen99, Zhang99] for the details of our reactive system paradigm for process coordination.

Modelling of a teamwork process using a computer modelling language is a time consuming and difficult task. Given the exposure of graphical user interfaces oriented environments, it is now expected that a modelling support provided should be a visual editing system [Gruhn98]. However, many teamwork support environments do not offer this critical feature. There is no doubt that visual modelling would provide a quicker, easier and less-time consuming process modelling support regardless whether it is used by computing or non-computing professionals.

Figure 2 depicts the main layout of the Java applet for visual process modelling [Yang99a].

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 to scroll over the next consecutive hours, days, months or years. The vertical scrollbar allows for parallel tasks within the process. The idea behind the scrollbars is to ensure that process modelling is not limited in any way in order to provide a full view of the process which is not restricted by the time or size constraints.

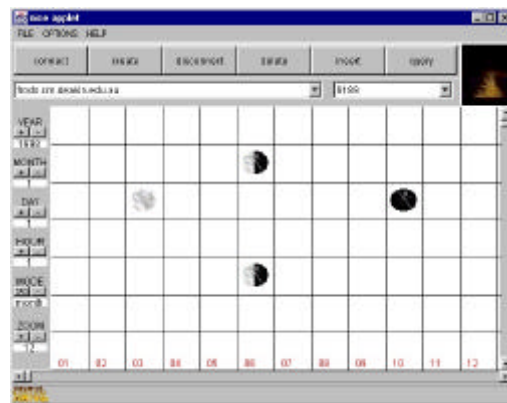


Figure 2: Main layout for teamwork process modelling

The teamwork manager can model the process visually by creating new tasks. The oval shaped objects located in between the grids are process tasks themselves which can be labelled, linked to specify the ordering and so forth. The “zoom” button permits the manager to zoom in or out of the interface. This allows for the manager to concentrate on the lower-level details of the process to “divide and conquer”. When facilitating the system, for example, if the manager clicks on the button to handle the starting/ending dates and other artefacts for a particular task, information of the selected dates and others will be mapped onto a database object and stored within the database. Similarly if the manager decides to link two tasks, the ordering information in the database will be update accordingly.

## 5 Supporting teamwork enactment

After a process is modelled by the teamwork manager, it is 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 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 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. We note that the layout of the process in Figure 3 is slightly different to that in Figure 2 because we are testing different visual presentations to see which one is more user friendly. With notification, 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, team members normally do not rely much on the centralised server because they mainly work on the client side locally to carry out the tasks assigned. This creates the opportunity for us to explore teamwork support in an offline mode, such as work at home from time to time, in addition to the normal online mode.



Figure 3. Process coordination user interface

Team members can use local tools, or tools available in the online teamwork 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 described in the next section may be better automatically invoked to allow team members to use as a shared workspace. Even some local tools such as a single-user editor for individual team members 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. The richness of data/object types, such as multimedia, can also be achieved. To manage data exchange in a teamwork 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 for fine-tuning effective data exchange.

When a certain task is finished or a value of the sensor based on team member's input is changed, a notification is sent to notify the process support environment over the server side. The process engine (a daemon) of the environment will utilise the decision making policy to generate new to-do lists for all affected team members. For a team member working in a team 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 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 task to indicate whether a task is enacted, enacting or 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. The global view of the process is adjusted automatically whenever the status of any task is changed.

## 6 Supporting synchronous teamwork

Process coordination described in the preceding section has involved various mechanisms for supporting shared workspaces in an asynchronous manner. It is common that most teamwork activities are undertaken by team members individually. That means that most tasks are carried out asynchronously, but interdependent, i.e. the outcome of a task of one team member is often the input to other tasks of other team members. However, some tasks are shared, i.e. they involve team members working together synchronously to complete the activity such as cooperative editing and brain storming. For example, the integration activity in Figure 3 of the previous section may involve two team members to edit synchronously in order to merge two separate parts of the document into one single piece.

Real-time distributed cooperative editing systems allow physically dispersed people to view and edit shared documents at the same time. They are very useful facilities in the rapidly expanding area of groupware and CSCW applications. 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 [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 (without noticeable delay) and the latency for remote user actions should be low; (2) distributed - cooperating users may reside on different machines connected by the Internet; and (3) unconstrained - multiple users may concurrently and freely edit any part of the document at any time, in order to facilitate free and natural information flow among cooperating users. Our novel 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 can be easily integrated with our teamwork process environment in order to provide a shared workspace for synchronous collaboration among team members.

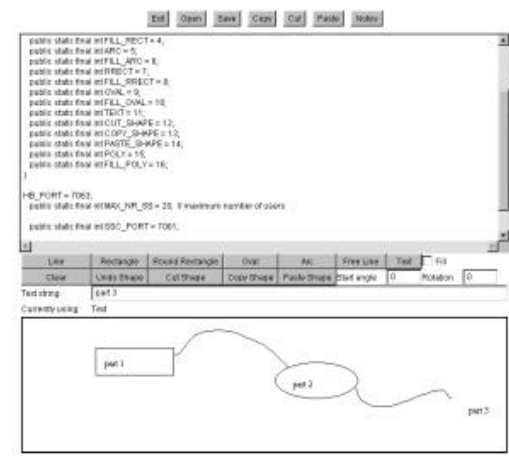


Figure 4. Cooperative editing supported by a whiteboard

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 team members to draw free style graphical objects, select and draw pre-defined shapes with optional fillings, or input text strings. The text editing panel on the top allows team

members to edit the document cooperatively without any constraints, i.e. edit at any position of the text at any time.

## 7 Supporting process evolution and resource management

Teamwork processes are dynamic entities that need to evolve to take into account changes in the technology, in the goals and requirements of the organisation, in the market place, and customers needs. Teamwork is often difficult to be planned completely in advance. In many occasions, process evolution or change on the fly is very critical to the applicability of the process-centred environments for teamwork because it is directly related to teamwork modelling and enactment we addressed earlier. Some interesting work has been done such as [Kaba96] and we have investigated this issue intensively [Yang97b]. In general, process evolution may be requested to any process tasks. When a change is requested, depending on its status (enacted, enacting, unenacted), corresponding actions need to be taken to accommodate the change appropriately.

Based on the process repository using a database system, we illustrate our effective process evolution mechanism by an example. For instance, at the current time, if some error is detected which is rooted to task A, it is necessary to rollback to the start point of task A to un-do and then re-execute/re-do it in the new context. To enable correct rollback, it is necessary to keep a process write-log to record the track of what have been done. In addition, task A may have some impact on the following-on tasks. For those following-on enacted and enacting tasks, depending on the circumstances, if they are affected by task A or in another word, dependent upon task A, they need to be rolled back and re-executed. However, we can re-use unaffected tasks in order to achieve incrementality because rollback and re-execution cost very much, especially when human resources are involved. To support such an incremental behaviour, it is essential to carry out a dependency analysis for each task. Clearly it is unnecessary to worry about unenacted tasks which will be executed based on the up-to-date process context eventually. Therefore, the change of unenacted tasks can be directly carried out individually.

Resource management is another important issue for teamwork modelling and enactment, given the nature of global teamwork in which

resources including team members, documents, and hardware/software are so dynamic. To better facilitate management of resources in an automatic or semi-automatic fashion, we have proposed to use a trader which can handle dynamic resource attributes effectively [Yang97a]. Our mechanism is based on the following reasonable assumptions: 1) there is a dynamic resource pool which includes all types of resources; 2) all resources have registered with a trader, i.e., exported to the trader as service offers; and 3) there are a number of system tools from which the dynamic attribute values can be obtained.

Given the space limit of the paper, we only address our mechanism briefly. In Sections 3 and 5, we mentioned the process engine which is the core part of a process-centred environment to coordinate and manage the process execution. For example, normally the process engine inputs a teamwork process description, allocates required resources to tasks according to the executing order defined in the task description. After completion of a task, the process engine releases the resources, resets any changed resource status, and then starts the next task(s). The process engine interacts with the trader to find available resources, i.e., importing. It is also responsible for updating the information held in the trader, for instance, modifying or withdrawing a service offer.

## 8 Conclusions and future work

In this paper, we have described an effective semi-centralised multi-tiered architecture for Web-based global teamwork support and indicated some of our prototypical work for carrying out a teamwork process in a computer-mediated automatic fashion. The issues involved include visualised teamwork modelling and enactment for asynchronous coordination with dynamic process evolution and resource management, as well as for synchronous collaboration. In this paper, we have also illustrated corresponding innovative mechanisms to implement the proposed effective architecture for teamwork support. With the fine-grained Web/Java approach, we can dramatically reduce the costs and delays associated with distributed information; provide up-to-date (Internet-based cooperation) software tools without local installation; and offer a simple, extensible and standard (platform-independent) environment.

In the future, we need to further improve Web-

based visualised teamwork support mechanisms and the reactive system paradigm as the underlying technology. We also need to further evaluate our current outcomes so as to improve the architecture adopted. For the teamwork support environment in general, many things can be investigated such as better mobility, interoperability, security and tool integration.

## Acknowledgement

This project has been supported partially by several ARC (Australian Research Council) grants. We are grateful for some implementation support from P. Wojcieszak, D. Brain, D. Zhang and P. Jeffers.

## References

- V. Ambriola, R. Conradi and A. Fuggetta (1997). Assessing process-centred software engineering environments. *ACM Transactions on Software Engineering and Methodology*, 6(3):283-328.
- R. Bentley, T. Horstmann, K. Sikkil, and J. Trevor (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/>.
- C. Chen, W. Zhou and Y. Yang (1999). Coordination mechanisms for the teamwork support. In *Proc. of 1999 Asia Pacific Decision Sciences Institute Conference*, pages 389-391, Shanghai, China, June. 1999.
- E. Evans and D. Rogers (1997). Using Java applets and CORBA for multi-user distributed applications. *IEEE Internet Computing*, 1(3):43-55.
- P. H. Feiler and W. S. Humphrey (1993). Software development and enactment: concepts and definitions. In *Proc. of the 2nd Int. Conf. on Software Processes*, pages 28-40, Berlin, Germany.
- A. Fuggetta and C. Ghezzi (1994). State of the art and open issues in process-centred software engineering environments. *The Journal of Systems and Software*, 26:53-60.
- I. Gorton and S. S. Motwani (1996). Issues in cooperative software engineering using globally distributed teams. *Information and software technology Journal*, 38(10): 647-655.
- H. Groiss and J. Eder (1997). *Workflow*

- systems for inter-organizational business processes. *ACM SIGGROUP Bulletin*, 18(3):23-26.
- V. Gruhn and J. Urbainczyk (1998). Software modelling and enactment: an experience report related to problem tracking in an industrial project. In *Proc. of the 20th Int. Conf. on Software Engineering*, pages 13-21, Kyoto, Japan.
- J. C. Grundy, M. Apperley, J. G. Hosking and W. B. Mugridge (1998). A decentralised architecture for software process modelling and enactment. *IEEE Internet Computing*, 2(5):53-62.
- D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman and A. Shtut-Trauring (1990). STATEMATE: a working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403-414.
- A. B. Kaba and J-C Derniame (1996). Modelling processes for change: basic mechanisms for evolving process fragments. *Software Process Technology, Lecture Notes in Computer Science*, 1149:99-107.
- E. Ly (1997). Distributed Java applets for project management on the Web. *IEEE Internet Computing*, 1(3):21-26.
- P. Oreizy and G. Kaiser (1997), The Web as enabling technology for software development and distribution, *IEEE Internet Computing*, 1(6):84—87.
- M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhauser (1996). An integrating, transformation-oriented approach to concurrency control and undo in group editors. In *Proc. of ACM Conference on CSCW*, pages 288-297, Boston, USA.
- W. Scacchi and J. Noll (1997). Process-driven Intranets: life-cycle support for process reengineering. *IEEE Internet Computing*, 1(5):42-49.
- A. Sheth (1997). Workflow and process automation in information systems: state-of-the-art and future directions. *ACM SIGGROUP Bulletin*, 18(1):23-24.
- C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen (1998). Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems. *ACM Transactions on Computer-Human Interaction*, 5(1):63-108.
- Y. Yang and Y. Ni (1997a), Resource management with trader for distributed software processes. In *Proc. of Int. Symp. on Future Software Technology*, pages 63-68, Xiamen, China.
- Y. Yang and Y. Zhang (1997b). A process evolution mechanism using prevalent databases as process repository. In *Proc. of IASTED Software Engineering Conf.*, pages 40-44, San Francisco, USA.
- Y. Yang (1998a). Issues on supporting distributed software processes. *Software Process Technology, Lecture Notes in Computer Science*, 1487:243-247.
- Y. Yang, C. Sun, Y. Zhang and X. Jia (1998b), A Web-based real-time cooperative editor in Java. In *Proc. of WebNet98 (World Conf. of the Web, Internet and Intranet)*, pages 975-980, Orlando, USA.
- Y. Yang and P. Wojcieszak (1999a). Visual programming support for coordination of Web-based process modelling. In *Proc. of the 11th Int. Conf. on Software Engineering and Knowledge Engineering*, pages 257-261, Kaiserslautern, Germany.
- Y. Yang, D. Zhang and P. Wojcieszak (1999b), Coordination management with two types of databases in a Web-based cooperative system for teamwork. In *Proc. of the 1st International Symposium on Database, Web and Cooperative Systems*, pages 47-52, Baden-Baden, Germany.
- D. Zhang and Y. Yang (1999), Coordination of teamwork processes with a fine-grained decision-making policy. In *Proc. of the 2nd International Conference on Information, Communications and Signal Processing*, Singapore, Dec. 1999. To appear.
- W. Zhou and E. Eide (1998). Java sensors and their applications. *Australian Computer Science Communications*, 20(1):345-356.