Supporting Online Web-based Teamwork in Offline Mobile Mode too

Yun Yang
School of Information Technology
Swinburne University of Technology
PO Box 218, Hawthorn, Melbourne, Australia 3122
yun@it.swin.edu.au

Abstract

Research into Web-based teamwork support has assumed that the network connection exists permanently, i.e. online. However, there is an increasing demand that teamwork environments should also support the offline mobile scenario that is so far not well addressed. This paper describes the radical requirements, novel mechanisms and innovative prototype for team members to work in an offline mobile mode in addition to that of the normal online mode. The prototype supports teamwork between the online and offline mobile modes with smooth switching over.

1. Introduction

A study is pointed in [15] that developers of large systems spend 70% of the time working with others. 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 [12]. Given the exposure of the Web and Internet, it becomes more and more common to support teamwork online. However, there is also an increasing demand that teamwork environments should also support the offline mobile scenario. For example, given the popularity of laptop/notebook computers, with appropriate application software support, team members can work offline, such as at home or during travels, as well as online as usual when they have network access, such as in offices. According to the literature, the challenges of one of the three broad areas for mobile computing is building applications that deal with the arbitrary disconnected nature of mobility [11], i.e. offline.

In this computing era, a task is usually carried out by a cooperating team who may be physically dispersed by using various (software) tools. Systems for computer-mediated teamwork, groupware, workflow or computer-supported cooperative work (CSCW) offer various automatic support for team cooperation to increase the productivity. Generally speaking, a teamwork task or process is normally composed of sub-tasks which are partially ordered [6]. By partial ordering, it means that a sub-task should and can only start when its previous sub-tasks have been completed to a certain satisfactory level. How to manage sub-tasks is the key issue for completion of the entire task. Teamwork support systems such as process-centred environments 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 [3]. Similarly, workflow systems have also been investigated intensively and quite a few commercial products are available [14]. However, there are still many open issues to be solved in a long run [7, 2, 1, 14].

In a Web-based environment, there exist quite a few systems for teamwork support. The prototype developed by Scacchi and Noll [13] takes an approach of using HTML forms and associated CGI scripts for process support. The workflow system investigated by Groiss and Eder [8] 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 [4] is a Web-based system to support primarily a shared workplace in an asynchronous manner. The work implemented in Java by Ly [10] is mainly for project management. In Serendipity-II, Grundy et. al. use a decentralised architecture for process modelling and enactment [9]. However, so far, supporting offline teamwork in addi-
tion to the Web-based environment has not been well addressed. In reality, support for offline mobile computing is becoming more and more important to normal team members. According to [11], 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 fully explored. Our work is unique in terms of that we have an innovative semi-centralised multi-tiered client-server architecture for teamwork support, and more importantly, our prototype supports both online and offline mobile modes for teamwork.

There are many perspectives can be investigated in this area. However, this paper only focuses on fundamental architecture and the corresponding high-level prototype. In this paper, we describe our architecture for teamwork support first. We then discuss the radical requirements and novel mechanism design for supporting teamwork in an offline mobile mode. After that, we illustrate the teamwork coordination enabled by our prototype in both online and offline modes before we conclude our work.

2. Architecture for supporting teamwork processes

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 evolved from [16]. The online part of this teamwork support 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 offline part on the left will be addressed in the next section. The advantages of this kind of architecture is 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, e.g. REDUCE [20]. 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 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.

For data repository, we have experimented with two types of databases [18]: 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.

In addition to making 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. 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. Furthermore, using combination of Web/Java seems better than using Web/CGI (common gateway interface) [5] in terms of performance and control/data granularity. Therefore, we have treated the Web and Java as an excellent vehicle to prototype our teamwork support mechanisms in a global distributed environment.
3. Supporting offline mobile mode: requirement analysis and mechanism design

With the online teamwork support architecture depicted in Figure 1 and based on the initial prototype conforming to the architecture, to practically handle additional offline mobile teamwork support, teamwork should still be coordinated online with the Web support as before [19]. However, team members can be authenticated to check out sub-tasks in order to work offline, say with their laptop computers. Once a sub-task is checked out, proper locking needs to be in place from the consistency control point of view. After some work being done during the offline mobile mode, when they connect to the network next time, i.e. online, authenticated team members can check in sub-tasks with progress made so far to inform the online centralised server for continuous teamwork coordination.

In the offline mobile mode, since there is no such a centralised server available like that in the online mode, teamwork sub-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 central issue. This demands a proper view of teamwork in the online mode to be retained in the offline mobile mode. This offline view needs to provide a similar "look and feel" as in the online mode to allow team members to work alone. It must contain sufficient information to support the offline mobile mode that should be saved when the check out function is requested in the online mode. In particular, the saved view would be able to play the role of the virtual server to provide a to-do list for the team member and to work on sub-tasks. This view derived during check out should contain all information that can be retrieved from the centralised data repository. Similarly, all input information such as documents should also be saved in the view. More importantly, the view should contain the workspace which is a Java applet that can be invoked locally as of online. Certainly, in this case, only local offline tools can be used for carrying out the sub-tasks.

For offline data repository, there are two general mechanisms available: the file system and the database system. The file system may be more system independent with less software requirements. However, the database system is more flexible although it requires database systems installed locally. We planned to experiment with both the file system and the database system and for database system, we planned to focus on using (1) Java JDBC with the Access relational database from Microsoft and (2) the ObjectStore object-oriented database from Object Design. This would provide an ideal platform for us to compare various implementation strategies. It was also the plan to have dynamic configuration of the system so that the team member could select a suitable configuration if necessary.

To support the offline mobile teamwork, the online teamwork support architecture needs to be extended to some degree in a very natural fashion as reflected on the left-hand side of Figure 1. In principle, the front-ends need 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 authenticated check out facility would enable the online mode to switch to the offline mobile mode and the authenticated check in facility would enable the offline mobile mode to switch back to the online mode.

4. Teamwork support in both online and offline mobile modes

After requirement analysis and mechanism design addressed in the previous section, 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 [2]. However, process modelling is not the topic for this paper and readers are referred to our other publications [17, 20] 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, the task "take2" is on the to-do list for team member "choo". 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/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 good opportunity for teamwork support in an
offline mobile mode in addition to the normal online mode.

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 2. 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.

In the online mode, 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 co-operative editing tool (e.g. REDUCE mentioned earlier) 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 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.

In the 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 because of the platform independence. However, the team member may select relational or object-oriented databases as data repository. When a certain task is finished, for example, no matter via an online or offline mode, a notification can and should be sent to the centralized server to notify the process support environment. The process engine (a daemon) of the environment will utilise the decision-making policy to
generate updated to-do lists for all affected team members.

5. Conclusions and future work

In this paper, we have addressed the requirements and mechanisms for supporting process-centred teamwork in an offline mobile mode in addition to the normal online mode. The corresponding architecture and prototype have also been described for supporting both online and offline mobile modes of teamwork. The effective check in and check out features enable the similar "look and feel" user interface in the offline mobile mode as of the online mode. This smooth transfer between working online and offline opens up much more flexibility for teamwork in contrast to the current prevalent online-only fashion.

In the future, we need to work further on general issues like visual programming support for Web-based teamwork modeling and enactment. Many other research issues are also under investigation such as better process evolution, mobility with agent technology, interoperability and tool integration.

Acknowledgement

I am grateful for some implementation support from P. Wojcieszak, D. Zhang, P. Jeffers, L. E. Tan, E. N. Chia, L. C. Choo, and D. Brain.

References