Tool Integration in a Process-Centred Web-based Teamwork Support Environment in Java

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

Abstract

This paper focuses on the tool integration perspective in software development to address the application of traditional tool integration mechanisms as well as new integration mechanisms based on the Web and Java. Our process-centred Web-based teamwork support environment is used as a case study to illustrate the potential power for tool integration.

Research into process-centred teamwork has been intensively carried out in communities of such as software and business engineering, computer-supported cooperative work and information systems for more than a decade. However, given the exposure of the Web and Java, a significant impact has formed to computer-mediated teamwork so that it can be beneficial to most people including both computing and non-computing professionals. This paper pays a special attention to tool integration mechanisms used in our Web-based teamwork environment development prototyped in Java, with respect to the access to Web data and integration with applications.

1. Introduction

The architecture of an integrated environment composed of multiple software tools must recognise two kinds of *interface* [10]:

- the interface presented to the user of the overall tool set
- the interface between component tools within the set

In other words, the first interface is related to *presentation* or *user interface* integration, and the second interface is related to *tool interface* integration including *control* and *data* integration. This classification scheme is based on such integration mechanisms as *user interface*, *control* and *data* integration, and is commonly referenced [7, 2]. In this paper, we focus on tool interface issues, and therefore the aspects of control and data integration are of the major concern.

To explain tool integration mechanisms in a more illustrative manner, we use our Web-based teamwork support environment as a case study which is described briefly in this section. Teamwork is a key feature in any workplace organisation. In this computing era, many computer-mediated tasks are carried out by team members, who may be physically dispersed, by using various (software) tools. Tasks such as successful development of complex (software) systems depend on support of an appropriate and useful set of tools.

Recently, there is a growing interest to support co-operative work over the Internet and the Web. The emergence and wide-spread adoption of the Web offers a great deal of potential for the development of collaborative technologies. The Web, as enabling technology for software development and distribution, changes the fundamental assumptions ingrained in the discipline as follows [6]: (1) accessible, cheap, direct customer channel; (2) remote, frequently updated resources; (3) new medium of software distribution; (4) large, globally accessible information space; (5) Internet-based collaboration tools; (6) large information space searches; and (7) simplicity, extensibility, and standardisation.

Generally speaking, a task is normally composed of sub-tasks which are partially ordered [4]. By partial ordering, we mean that a sub-task should and can only start when its previous sub-tasks have been completed or reached certain threshold. How to model and then carry out sub-tasks with various supporting tools involved is the key issue for completion of the entire task.

With software support, team members can be coordinated by a system, which is normally more effective than 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, team members may reside in Australia, Europe and North America. With around 8-hour time differences among locations, 24 hours a day working mode can potentially be facilitated [5]. Even if team members are co-located in the same building, distributed teamwork is still desirable for various reasons such as teamwork coordination support and effective information/tool sharing.

In order to integrate tools smoothly, we need to consider the tool interface which concerns control and data integration as indicated earlier. In general, control integration refers to the ability of tools to notify one another of relevant events, as well as the ability to activate other tools under program control, whilst data integration addresses sharing and exchange of data among tools. In this paper, we address the integration process, using our process-centred Web-based teamwork support as a particular example, to illustrate extra features enabled by the Web and Java. So far, this topic has not been explored thoroughly.

This paper is organised as follows. First of all, the traditional tool integration mechanisms are summarised. Then the background of our teamwork support environment is described. After that, the integration mechanisms involved in Web-based teamwork are illustrated. Finally, discussion on the tool integration process is addressed, followed by conclusions and future work.

2. Traditional mechanisms

From the control integration viewpoint, tools must be able to notify one another of relevant events. It is related to communication issues among tools [2]. Traditionally, control integration among tools can take place in following ways:

- C_1 – *indirect control integration*
  Tools can be activated via facilities accessible directly to the user. For example, an operating system command which is accessible to the user can also be issued (as a *system call*) by another tool.

- C_2 – *triggers*
  Tools can be activated through triggers which are events in a database or object base and cause certain actions to take place when particular item(s) are touched.

- C_3 – *a message server*
  Tools can be activated by the message broadcast mechanism in a message server when particular messages are received.

- C_4 – *procedure calls or method invocation*
  Tools can be activated through (remote) procedure calls or method invocation.

From the data integration viewpoint, tool integration requires both sharing of data among tools and managing the relationships among data objects produced by different tools. Accordingly, relevant data integration between tools can occur in a number of ways, which may be employed in a mixed manner:

- D_1 – *intermediate files*
  Tools can transfer data using intermediate files.

- D_2 – *a database or object base*
  Tools can exchange data via a shared repository such as a database or object base.

- D_3 – *message passing*
  Tools can exchange data using explicit messages.

- D_4 – *a "canonical" representation*
  Tools can share a canonical internal data representation.

Both control and data integration mechanisms can be used in a mixed manner for tool integration to form various integration paradigms [9]. The above classification scheme is in effect from the viewpoint of system developers which suits our needs although there are other tool integration classifications such as from the productivity gain point of view [10]. Therefore, the scheme described in this section is utilised in this paper.

3. Background for case study

To support process-centred Web-based teamwork, a variation of the semi-centralised multi-tiered client-server architecture is proposed as shown in Figure 1 [11]. It includes (1) clients as front-ends using local Web servers and tools, (2) centralised servers (for such as teamwork coordination) with tools, and (3) supporting tools such as databases and file systems as back-ends.

In Figure 1, the centralised server site plays the key role for managing a task. The coordination information resulted from teamwork modelling is stored in the database repository. We note that the database repository is a general concept which can include various databases such as relational and object-oriented

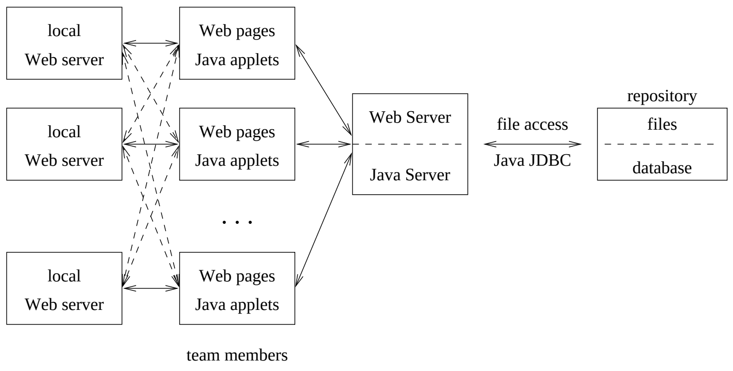


Figure 1. Architecture for supporting teamwork

databases. During 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 available in a distributed fashion rather than only centralised.

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 for heterogeneous environments, has encouraged us to prototype our work in Java, based on the Web environment. Moreover, no particular teamwork software needs to be installed for team members since Java applets can be downloaded on the fly and then run directly. However, local tools can still be used for carrying out sub-tasks. Furthermore, using combination of Web/Java seems better than using Web/CGI (common gateway interface) [3] in terms of performance and control/data granularity, though the Web/CGI technique can still be facilitated in some cases. Therefore, we have treated the Web and Java as an excellent, if not ideal, vehicle to prototype our teamwork support environment.

Clearly, our Web-based teamwork support environment is an integrated tool set. From the tool integration point of view, we need to consider both control and data integration issues. All these issues should be tackled for both teamwork modelling and enactment. In addition to built-in teamwork support tools, local tools and other Web-based tools can be facilitated to carry out a sub-task. Information, or data, can be stored in a centralised site as well as on various local Web sites. How to invoke tools and share/exchange data effectively in a Web-based environment is critical.

4. Integration for teamwork

In our process-centred Web-based teamwork support environment, firstly, the teamwork manager can and needs to model the teamwork for individual sub-tasks and the orders among them, via the teamwork modelling Java applet available from the server site according to Figure 1 in the previous section. In this section, we focus on handling sub-tasks regarding tool integration without detailing the ordering since it is not directly related.

The Java applet allows the manager to specify various attributes for each sub-task including the team members, tools, documents, deadlines, instruction messages and so forth. For tool integration in particular, if a specific tool needs to be used, the tool attribute can be specified as, for example, our Web-based cooperative editor [12] or a single-user editor such as *Notepad* on PC or *vi* on Unix which can be automatically invoked during enactment by the Java applet appropriately based on the running environment. Along the same line, the input documents need to be specified to allow team members to work on in order to generate output documents, also as specified. These documents can be stored at the centralised site or at local sites as long as their locations are specified and they are accessible to team members involved. Similarly, the instruction messages can be included to describe briefly the sub-task of how it should be carried out. It is also feasible to include messages to indicate what tools to be used and what and where documents are or should be put.

The teamwork coordination information is stored in the repository at the back-end according to Figure 1. At this stage, we use the Java JDBC interface to connect to Oracle (can be other relational databases without changing the Java code). In effect, JDBC is a low level middleware tool that only provides the basic features to interface a Java application with a relational database. With using JDBC, we have to design a relational schema to which they will map Java objects. Then, to write a Java object to the database, we have to write code to map the Java object to the corresponding rows of the corresponding relations. The same transformation has to be done in the other direction to read a Java object from the database. Hence, we have investigated the ObjectStore object-oriented database system, which has a Java interface, so that we only need to handle objects directly which not only should improve the productivity but also increase the efficiency [13]. In addition, use of an object-oriented database matches our needs better since Java itself is object-oriented.

In teamwork modelling, tool integration mechanisms for various tools, as an integrated tool set, are used. However, to keep the description simpler and to avoid repetition, integration mechanisms are only discussed in teamwork enactment next using symbols C_n and D_n (where $n = 1..4$) from Section 2.

After a particular task is modelled, teamwork can be enacted, or executed. Enactment of the task is coordinated in an automatic fashion based on the teamwork support environment.

The work space for a team member is a (signed) Java applet, i.e. a tool, which is downloaded and invoked automatically via a Web page, i.e. data, with a Web browser, i.e. a tool again. This kind of control and data integration is different from traditional mechanisms.

For a team member, it may be very useful to have a global view of the project in a visualised fashion, as shown in the middle panel of Figure 2, in order to create a better teamwork atmosphere, which is important from the psychological point of view when a person works in a computer-mediated teamwork environment.

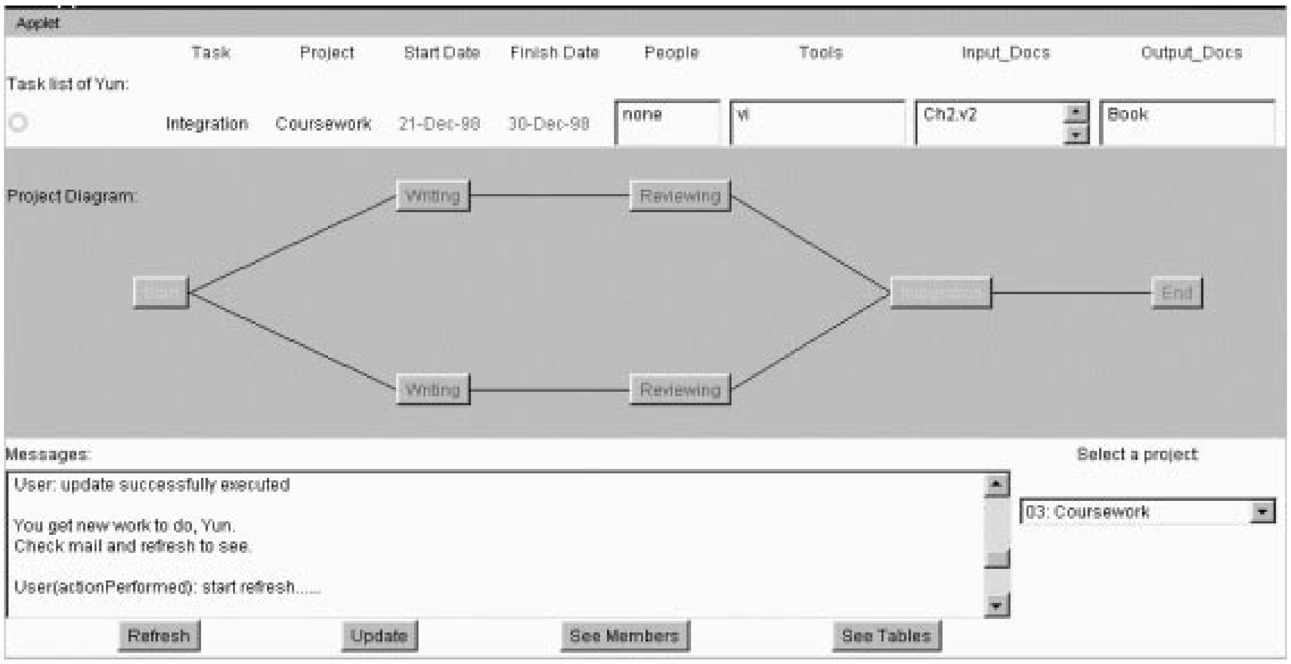
In addition, we divide sub-tasks into three different classes with respect to their enactment status: enacted, enacting and unenacted. By enacted sub-task, we mean that the sub-task has completed. By enacting sub-task, we mean that the sub-task is currently ongoing. By unenacted sub-task, we mean that the sub-task has not been launched yet. These three kinds of status are visualised by different colours on the screen.

For teamwork enactment, the most important and common facility provided for coordination in process support is a dynamic to-do list for each team member to inform the associated sub-tasks which need to be done. One effective method we envisage is that the server side, which realises coordination for process control, provides every team member a dynamic to-do list via email *actively* for notification and also within the Java applet, i.e. the work space, *passively* when a team member connects to the server with the identification or refresh the window. From the above description, we can see the usage of traditional D_3 (message passing) data integration mechanism of passing messages between the server and the Java applet for the updated to-do list whilst email notification can be seen as extension to D_3.

When a team member has a to-do list, the specified activities can be carried out. The team member may use server provided tools or locally available application tools. If a tool is pre-specified, it can be automatically invoked by the Java applet, which, from the control integration point of view, normally belongs to the usage of the traditional C_1 (indirect control integration) mechanism. In addition, input data (documents) should be available for the team member via the Web and output data (documents) may be sent to a centralised Web site, say via HTML forms supported by CGI scripts, or stored locally as specified. From the traditional tool integration viewpoint, the D_1 (intermediate files) data integration mechanism is normally used. However, the data integration mechanism supported by CGI scripts with HTML forms is different to traditional mechanisms. In addition, data integration has been extended in the Web context so that files can be distributed. Similarly, for each sub-task, instruction messages can be generated and/or made available to team members by either emails, via Java applet, or using CGI scripts which do not impose new features.

Once a team member has finished a sub-task on the to-do list, for instance, the notification should be made to the server via message passing to invoke the tool which reflects the usage of the C_3 (a message server) control integration mechanism. At the server side, appropriate coordination for process control can be adjusted to generate updated to-do lists for related team members. Since the coordination information is stored in the centralised database using the traditional D_2 (a database or object base) data integration mechanism, tools for sending notification are triggered using the traditional C_2 (triggers) control integration mechanism. At the same time, the database system is invoked via the corresponding JDBC methods, i.e. method invocation, which effectively uses the C_4 (procedure call or method invocation) control mechanism.

In this section, the traditional D_4 (a "canonical" representation) data integration mechanism is not explicitly demonstrated in our teamwork support environment. However, it has been utilised in some other tools as addressed by us earlier [10].



Task list of Yun:

Task	Project	Start Date	Finish Date	People	Tools	Input_Docs	Output_Docs
Integration	Coursework	21-Dec-98	30-Dec-98	none	vi	Ch2.v2	Book

Messages:

```
User: update successfully executed

You get new work to do, Yun.
Check mail and refresh to see.

User(actionPerformed): start refresh......
```

Figure 2. A screen for team members

5. Discussion

Although the traditional tool integration mechanisms including control and data integration are well understood and commonly used in the integration process, the impact of the Web and Java on tool integration has not been thoroughly investigated. In this section, we summarise the unique features involved in tool integration based on Web and Java support.

From the control integration point of view, with the CGI support, tools can be invoked at the server site either directly or via an HTML form. In addition, with Java support, an applet, i.e. a tool, embedded in a Web page, i.e. data, at the server site can be invoked automatically and executed at the client site when the Web page is downloaded via a Web browser, i.e. a tool.

From the data integration point of view, with the Web support, data can be accessed across the Internet with some simple and extensible standards, such as HTTP, which make integration much easier. In addition, the richness of data/objects types, such as multimedia, can be achieved. However, this may make data integration more complicated.

In terms of integration process in a general sense, with Web and Java support, some extra features are available as follows. Firstly, a distributed heterogeneous computing environment is natural for tool integration. Secondly, tools can be distributed easily and updated timely, for example, Java applets are downloaded on the fly which can always be kept up to date. Thirdly, Internet and/or Web based tools can be accessed and used for the task straight away. Finally, the Web also enables flexible distribution of data and provides a large, globally accessible information space which, for example, can be used for search to solve problems and carry out tasks more effectively.

However, we should be aware that software design for the Web is very different from that of traditional applications - designing and developing for the Web opens up a broad range of new, often untried, possibilities [8]. Some issues listed in [1] include the constraint with the user interface design based on HTML, limitation on “quality of service” due to HTTP, and so on where the user interface can be significantly improved if Java applets are used.

6. Conclusions

In this paper, we have overviewed the traditional tool integration mechanisms with respect to control and data integration from the viewpoint of system developers. With the Web and Java as enabling technology, some new mechanisms have surfaced for tool integration since tools can now be invoked via the Web page with both CGI and Java support, and a much larger and easily accessible information space is now available. These tool integration mechanisms have been illustrated in our process-centred Web-based teamwork support environment prototyped in Java to a certain degree in this case study.

On one hand, we can see the good opportunities to support more flexible tool integration, given the existence of the Web and Java. On the other hand, we should also note the potential complexity and constraints imposed to software development, given the exposure of the Web and Java. In the future, with Web and Java support, we need to work on further experiments on tool integration for our teamwork support environment in particular, and the integration process for software development in general.

7. Acknowledgement

I am grateful for implementation support from many people, particularly D. Brain, D. Zhang and P. Jeffers.

References

[1] R. Bentley, T. Horstmann and J. Trevor. The World Wide Web as enabling technology for CSCW: The case of BSCW. http://bscw.gmd.de/Papers. *Computer-Supported Cooperative Work: Special issue on CSCW and the Web*, Vol. 6, 1997.

[2] D. Clément. A distributed architecture for programming environments. *ACM SIGPLAN Notices*, 26(1):11-21, 1991.

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

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

[5] 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.

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

[7] A. I. Wasserman. Tool integration in software engineering environments. it Proc. of Int. Workshop on environments. Lecture Notes in Computer Science, 467:137-149, Chinon, France, Sept. 1989.

[8] A. I. Wasserman. Design issues for WWW development. http://www.methods-tool.com/SEweb.html, 1997.

[9] J. Welsh and Y. Yang. Integration of semantic tools into document editors. *Software – Concepts and Tools*, 15(2):68–81, 1994.

[10] Y. Yang and J. Han. Classification of and experimentation on tool interfacing in software development environments. *Proc. of Asia-Pacific Conf. on Software Engineering*, pages 56-65, Seoul, Korea, Dec. 1996.

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

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

[13] Y. Yang, D. Zhang and P. Wojcieszak. Coordination management with two types of databases in a Web-based cooperative system for teamwork. *Proc. of 1st International Symposium on Database, Web and Cooperative Systems*, pages 47-52, Baden-Baden, Germany, Aug. 1999.