DEVELOPING DISTRIBUTED SOFTWARE DEVELOPMENT TOOLS IN JAVA ON THE INTERNET*

Yang Yun

[School of Computing and Mathematics, Deakin University, Geelong, Australia 3217]
E-mail: yun@deakin.edu.au

Manuscript received March 20, 1998; revised July 30, 1998

Abstract This paper focuses on employing the leading-edge Java programming language to develop Internet-based software development tools. In the real world, most (non-trivial) software development tasks are carried out by development teams using various software tools. In this Internet era, it becomes extremely desirable that software development can be conducted in a distributed fashion via a Web-based environment. This demands sophisticated development of distributed software development tools. In this paper, Java networking support is described in order to develop distributed software development tools. Requirements and the corresponding framework of a Web-based process-centred software development tool to support global teamwork are also introduced to illustrate the role of Java.

Key words software development, software processes, Java, Internet, World Wide Web, distributed computing, networking, teamwork

1 Introduction

The life-cycle of a software development task is essentially composed of sub-tasks such as analysis, design, coding, testing, and maintenance, which are normally carried out by team members in a cooperative fashion. This kind of cooperation, at the high level, is normally facilitated by a process-centred tool, often in a constrained manner. In addition, take a simplified typical scenario of the design stage, when there is more than one person involved in the design phase, the production of the design document becomes a cooperative activity to which some tool supporting co-authoring could be applied. Therefore, the nature of any non-trivial software development could be viewed as a task that is carried out by a team of people, using various tools as appropriate. From the above description, we can see clearly the three key components involved in software development: the task, the team, and the tool.

* Work reported in this paper has been partially supported by an Australian Research Council small grant in 1998 and a School of Computing and Mathematics seeding grant from Deakin University in 1997.

©1998 by Allerton Press, Inc. Authorization to photocopy individual items for internal or personal use, or the internal or personal use of specific clients, is granted by Allerton Press, Inc. for libraries and other users registered with the Copyright Clearance Center (CCC) Transactional Reporting Service, provided that the base fee of $50.00 per copy is paid directly to CCC, 222 Rosewood Drive Danvers, MA 01923.
In general, task component oriented technology, facilitating team coordination, is management-centred. Team component oriented technology, facilitating team communication, is interaction-centred. And tool component oriented technology is enactment-centred focusing on tool integration to support the above technologies.

Teamwork is a key feature in any workplace organisation. Developers of large systems spend 70% of the time working with other team members, who may physically dispersed. Tools for computer-mediated teamwork, groupware or computer-supported cooperative work (CSCW) offer various automatic supports for team cooperation to improve the productivity.

With software support, team members can be coordinated automatically by so called process centred tools [1], 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 Asia, Europe and North America. With around 8-hour time differences among locations, 24 hours a day working mode can be facilitated potentially [2]. Even if team members are co-located in the same building, distributed teamwork is still desirable for various reasons such as sharing information and tools effectively over the Internet or Intranet - as part of the Internet.

In this paper, we first summarise the motivations of using the Web and Java in Section 2. We then focus on Java support for developing distributed system tools in Section 3. After that, we describe our requirements and the corresponding framework for global teamwork support in Section 4 followed by a re-visit on the three key components of task, team and tool in Section 5. Finally, we conclude this paper and point out our future work.

2 Motivations

Recently, there is a growing interest to support cooperative 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 as an enabling infrastructure. According to [3], the Web, as enabling technology for software development and distribution, changes the fundamental assumptions ingrained in the discipline as follows: (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.

In addition to Web, Java is a simple, object-oriented, distributed, robust, secure, architecture neutral, portable, interpreted, high-performance, multithreaded, dynamic language. The motivation of using Java is as follows: (1) already the second desirable language for software development in early 1997 before it turned into age two – behind C++ which is more than a decade old; (2) a leading-edge (general-purpose) language – purely object-oriented (much neater than C++) and providing Web support; (3) write (the code) once and (then) run anywhere; (4) applets delivery over the Web; and (5) computation load
shift from the servers to clients which can normally increase the performance dramatically.

The above features have encouraged us to develop all our research prototypes in Java, especially in the Web environment. By doing so, no particular software needs to be installed for (client-side) team members since Java applets (i.e. up-to-date software) can be downloaded on the fly and then run directly. Meanwhile, it has become popular now with the component-based software solutions and Java matches the solutions quite smoothly. Moreover, using combination of Web/Java seems better than using Web/CGI (common gateway interface) [4] 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.

3 Java support for distributed system tools

The current dominating architecture for distributed computing adopts the client-server paradigm. With this paradigm, the server plays a central role to serve multiple clients when the clients connect to the server to pass messages around via the input and output streams. Java has an extensive library for networking to support distributed computing and hence distributed software development tools. In general, there are following mechanisms available: (1) socket; (2) datagram; (3) URL – uniform resource locator; (4) RMI – remote method invocation; and (5) IDL – interface description language, incorporating with common object request broker architecture (CORBA). With the Java library support, it normally makes creation of network connections as simple as that for a local file system which is much easier than that in C/C++.

Socket communication is a reliable connection-based point-to-point communication channel based on TCP/IP. A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The java socket package provides two classes: one for implementing the clients side of connection and the other for implementing the server side of the connection.

In addition to the commonly used socket communication, in Java, it is possible to communicate using datagrams, normally for the following purposes: (1) low-overhead communication of non-critical data; and (2) when a stream model of communication is not necessary. UDP (unreliable datagram protocol) is used to send datagrams fast with no guarantee to reach the destination and no guarantee about the order.

The final low-level communication mechanism is to use a URL to communicate for a specific purpose, i.e. to access resources on the Internet via a URL address directly. With this mechanism, the object referred to by a URL can be downloaded with a single call.

In recent Java releases, high-level (fully) object-oriented networking mechanism of RMI has been provided. RMI, similar to remote procedure calls, supports homogeneous Java-to-Java communication, based on the specialised protocol. To be more specific, RMI allows a Java program to call methods and access variables inside another Java program, which may be running in different Java environments or systems, and pass objects back and forth over a network connection. In contrast, Java support for IDL, incorporating with
CORBA, enables heterogeneous Java-to-anything communication, based on the standard protocols, which is also at the high object-oriented networking level.

4 Supporting Web-based software development teamwork

In this section, we describe our requirements and corresponding framework for DWebTeam to support distributed software development based on Java and the Web. DWebTeam has the following characteristics: (1) effective teamwork coordination support; (2) seamless integration of team communication and collaboration; and (3) entirely Web based.

Generally speaking, a task/process/project is normally composed of sub-tasks which are partially ordered [5]. By partial ordering, it means that a sub-task should and can only start when its previous sub-tasks have been completed. How to manage sub-tasks is the key issue for completion of the entire task. Hence, task-oriented technology is management-centred to facilitate project management focusing on coordination. Teamwork support systems such as process-centred environments have been investigated in various communities such as software engineering, business engineering, information systems and CSCW 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 [6]. Similarly, workflow systems have also been investigated intensively and in effect, quite a few commercial products are available. However, there are still many open issues to be solved in a long run [7, 1]. In particular, existing process-centred environments are primarily aiming at providing support at a higher level for the basic coordination of the entire process life-cycle of such as sub-tasks of analysis, design, coding, testing, and perhaps maintenance in the software development context.

In reality, the high-level coordination support is insufficient to cover the whole spectrum of teamwork [8]. Hence, we need to integrate facilities to support more effective team communication and collaboration. Effective communication among team members may take place in some combination of ways as captured by the well-known time-space matrix [9]. This matrix has a time axis indicating whether team member communication is asynchronous or synchronous and the space axis indicating whether team members are co-located or dispersed. For example, most of the component sub-tasks undertaken by team members are individual — they are carried out asynchronously, but interdependent. i.e. the outcome of one sub-task of one team member is often the input to another sub-task of another team member. Some sub-tasks, however, are shared, i.e. they involve team members working together synchronously to complete the sub-task. Team members also need effective collaboration support on various aspects of information sharing. Collaboration should be supported for effective information sharing among the team members with two aspects. The first aspect is change flexibility, which allows team members to update the shared information without much restriction and ideally without any constrains for communication. Most tools impose certain control, which is often over-restricted from the users’ viewpoint. This kind of restriction limits the useability of tools and it is viewed as one of the major factors for the failures of many teamwork support tools [10]. The sec-
oud aspect is change awareness, which offers effective notification of each team member’s actions when appropriate. Without sufficient support for group awareness of sharing, it is very easy that team members get lost of what they are supposed to do, and lose close contact with each other and finally do not feel that the tool offers a proper way for cooperation. Group awareness has been emphasised recently, particularly in the latest international CSCW conferences. Hence, team-oriented technology is interaction-centred to facilitate team communication and collaboration.

Based on the above statements, in summary, the framework of teamwork support should take all of them into account. To refine an appropriate Web-based framework for distributed teamwork, which is fundamental, we believe that coordination is suitable as a backbone for management of teamwork. At the same time, the inter-related communication and collaboration support should also be included to cover the spectrum of teamwork.

![Diagram](image)

**Figure 1: Information flow for supporting teamwork**

The multi-tiered client-server architecture of Web-based teamwork support from the end-users’ point of view is the most representative as depicted in Figure 1. It includes clients as front-ends using local Web servers and tools, centralised servers with associated tools, and supporting tools such as databases as back-ends. The centralised server site plays the key role for management/coordination of a task as well as provision of some tools to support communication and collaboration. A Java application for a particular teamwork-oriented task runs at the centralised server site as a daemon all the time serving for the entire life-span of the task. All task coordination related information is stored in a database repository which is accessed by the Java application for the task and the file system may also be used to store centralised information such as documents. Information such as documents can be stored locally at the client sites in a distributed manner and accessed by team members based on the existing Web support. Basically, at the client site, only an appropriate Web browser is required and no other particular software needs to be installed since each team member uses Web pages on the Internet and Java applets downloaded from the server site on-the-fly to carry out the sub-tasks assigned. At the moment, DWeb‘Team is implemented in Java using socket communication.

Once the server side has been set up for a teamwork task, the team members, who
are at the client side, can start to work on individual sub-tasks based on the Web via such as the Web pages and the downloaded Java applets. With coordination itself, team members can be informed of the sub-tasks to be carried out via email or by visiting the task Web page. For experimenting integration with synchronous teamwork, our REDUCE '11 (REal-time Distributed Unconstrained Cooperative Editing) prototype is employed, which is another Web-based tool implemented in Java based on socket communication.

5 Three key components: a re-visit

As mentioned earlier, normally a software development task is composed of sub-tasks, and the task component is primarily related to the software process context. How to manage sub-tasks, i.e. (sub)processes/steps, is the key issue for completion of the entire task and coordination of team members. The previous section has indicated how a distributed software development task can be managed by DWebTeam which is being implemented in Java in a Web environment. The management and/or coordination of carrying out sub-tasks can be manual, or systematic/automatic which implies some means of process support/integration, as implied by DWebTeam.

In general, this process support focuses on process integration which implies that sub-tasks are systematically managed for the purpose of coordination. Team members can cooperate (asynchronously) via sub-task control. For instance, when team member A finishes the coding sub-task, pre-defined team members B, C, D can start the review sub-task of inspecting the code produced by A, normally individually.

As soon as we consider work involving a team of more than one, we immediately have to consider interaction among the team members. With the well-known time-space matrix, team members involved in a software development task can interact via DWebTeam asynchronously based on coordination as well as interact synchronously via our REDUCE tool integrated with DWebTeam.

In general, this support is concerned with providing explicit support for interaction among team members, normally in relation to a single task, or sub-task in the context of the overall process. For instance, further to the example described earlier, in addition to individual asynchronous code review by team members B, C, D, a tool can be used for group review involving team members B, C, D synchronously.

The tool component focuses on the integration of a set of tools used by team members for the execution of the task. It plays an essential role for the enactment of task and team. Tool integration is recognised by Clément [12] as a central issue for software development environments because it addresses the cooperation of many tools, from many sources, in a uniform but open framework.

Tools are fundamental blocks for software development. Tool integration does not support teamwork directly, rather, it provides the basis for enactment. For example, there are many separate tools in the real world such as process and interaction support tools. Putting those tools together so that they behave like an integrated system is the major concern. DWebTeam and REDUCE can be viewed as integrated tools to support distributed software development.
6 Conclusions and future work

In this paper, we have described Java networking support for developing distributed software development tools on the Internet. In the real world, any non-trivial software development involves three key components: a task, a team and a set of software tools, as we addressed in this paper. In this Internet era, we indicate the reasons of using Java and the Web as the vehicle and the enabling technology to develop distributed software development tools, such as our process-centred global teamwork support tool, real-time cooperative editor and the integration of the two. In the future, we need to further investigate and prototype our teamwork support tools, with provision of a visual environment. In addition to the existing integrated tool set, we need to consider how to accommodate other important issues such as process evolution, dynamic resource management and process interoperability.

References