

# CoDevFrame: An Event-drive Grid Oriented Cooperative Software Development System

LIU Bo<sup>1,2</sup>, QI De yu<sup>1</sup>

<sup>1</sup>College of Computer Science, South China University of Technology, Guangzhou 510641, China  
Email: liugubin@sohu.com

<sup>2</sup>Dept. of Computer Science, South China Normal University, Guangzhou 510631, China

***Abstract.** Though twenty years have passed since the birth of CSCW, the original goal of it is not reached as well as people expected. This situation is mostly due to the supporting technology especially the infrastructure. Because of Grid computing and Web services, the globalization of software development has become a reality. Many software projects are now distributed in diverse sites across the globe. The distance between these sites creates several problems that did not exist for previously co-located teams. Problems with the coordination of the activities, as well as with the communication between team members emerge. This paper describes how event-drive servers that are built on grid and web services are a useful technology for supporting global software development. They provide some general yet flexible cooperation related services and organizes them into different layers. In general, advantages of using event-drive servers are described as well as specific problems that they help solve.*

**Keywords:** Cooperative Software Development, Global Software Development, Event-drive Servers, Grid Computing, Web Services

## 1 Introduction

Nowadays, globalization means that the economical, cultural and social boundaries of countries are disappearing. Software is no exception to this rule. For example, due to different government regulations, telecommunication companies need development sites in different countries to get a position in the local market. Therefore, they need to adapt their processes, tools, and organizational culture to overcome the disparity between the sites. In order to do so, they need to solve a wide variety of problems. The most obvious is the physical distance between the sites [1]. In this case, the sense of working in a team decreases due to the lack of interaction between the members of different sites. As a consequence, there is a lack of trust because the members usually do not have knowledge about overseas' culture. Relatively simple activities like discussing requirements in meetings cannot be performed. There are also strategic, cultural, knowledge management, project (and process) management and technical issues to be solved [2]. All these problems must be understood and properly resolved for global software teams succeed.

On one hand, the requirements on cooperative software development technologies become more and more intensified especially in business domain. On the other hand, though many cooperative development systems/tools have been developed, they are not widely deployed. Today, Grid [3] and service-oriented computing [4] will significantly affect the design and implementation of cooperative systems. Roughly speaking, Grid provides a strong infrastructure for deploying cooperative systems while Web service reduces the cost of system/tool integration. Thus, if we can utilize these new technologies, more applicable systems can be obtained. As an endeavor in this direction, event-drive servers that are built on Grid and Web services can be used as an infrastructure to support global software development, they incorporate the emerging Grid and Web service technologies and can both ease the burden of system development and overcome the limitations laid on current systems. Several technical problems that arise can also be solved, for example, using this technology, the lack of informal communication can also be partially solved by building collaborative tools, such as instant messenger systems. In order to use event-drive servers, events (representing activities) must be captured. One way to capture these events is through instrumentation of applications. Another way is through event monitoring [5]. Others, such as MILOS [6], use a publisher-subscriber component in its architecture.

The rest of the paper is organized as follows. Architecture of this framework is presented in section 2, detail design of its core is discussed in section 3. Then, the next section presents the advantages that can be obtained using event servers. Finally, Example scenario of event service is given and some conclusions are described.

## 2 Architecture of CoDevFrame

Two main obstacles for global software development are resource limitation and tool isolation. By resource limitation we mean that the current underlying network doesn't provide enough support for resource management, by tool isolation we mean that different tools are running separately and no interaction between them is taken into account. Grid technology presents an attractive vision for the future usage of computer and other electronic devices. Investigation into running Grid-related projects shows that most of them are concerned with resource coordination covering the topics such as Grid architecture, resource/data management or domain-specific applications. However, little effort has been dedicated to providing a systematical framework to support the rapid development of cooperative applications. In fact, cooperation is the ultimate goal for people to use information technology, so we propose a comprehensive framework -- CoDevFrame that is based on Grid technology to utilize the power of Grid to support large-scale and/or resource-intensive cooperation. To facilitate cooperation, Web service is also introduced into this framework to integrate various tools or domain-specific applications.

As Figure 1 illustrates, the underlying fabric layer in Figure 1 provides high-speed communication channel for upper layers. It supports most current network standards such as IPv4/IPv6 and 802.11b. Grid Infrastructure layer is divided into two sub-layers: Resource sub-layer and Services/Interface sub-layer. Resource sub-layer manages various resources such as storage, software services, computing power, instruments, and so on. Access to these resources is accomplished via Grid/web services interface in Services/Interface sub-layer. For research on Grid has gained some useful results (e.g., the release of Globus Toolkit 4), here we just adopt existing technologies to construct Grid infrastructure. CoDevCore adopts service-oriented architecture. To bridge the gap between Grid infrastructure and the diverse applications, it ships various common services. With these services, developers can then quickly construct new cooperative applications simply by tailoring them and developing new domain-specific services. Details of CoDevCore and middleware layer will be explained in the next section.

Event-drive services are the main parts of application layer, since the Internet is not reliable, a good way to think about how these applications work is as loosely coupled autonomous objects. Instead of using direct communication, these applications use an event-drive design style. Messages are not sent from an application directly to another, they are sent to an event dispatcher, which in turn distributes them to the applications that need them. In this case, there are two types of components: information sources and consumers. The information sources publish events, while the information consumers subscribe to particular categories of events. Events are sent by the information sources to the event servers, which ensure the delivery of these events to all interested information consumers. In general, in an event-based system, component interactions are modeled as events, which are transmitted in the form of notifications. An event server usually provides a *subscription service* that allows consumers to subscribe to different types of events using, for example, boolean (logical) connectors, regular expressions, sequence matching, and so on. There are many event notification servers in existence and they are surveyed elsewhere [7]. Three representative examples are CASSIUS [8], Elvin [9] and Siena [10]. CASSIUS was developed tailored to provide awareness information for groups. Elvin was originally developed to gather events for visualizing distributed systems, but it evolved later to a multi-purpose event notification system. Finally, Siena emphasizes scalability in distributed environments.

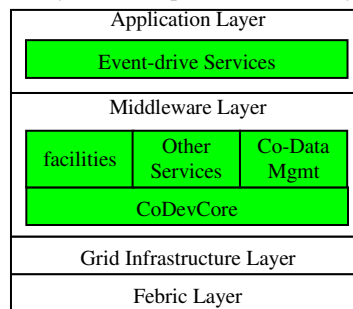
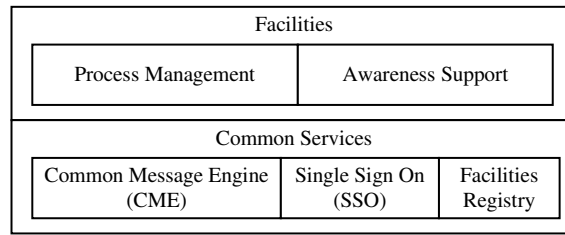


Fig.1. The Structure of CoDevFrame

## 3 Detail design of CoDevCore

Figure 2 gives the view for understanding the framework. As shown in the figure, the framework consists of some common services and a set of facilities. Details of them are as follows.



**Fig.2.** Functional View of CoDevCore

### 3.1 Common services [11]

Common services are built right on top of the underlying Grid infrastructure. In CoDevCore, Globus Toolkit 4 is adopted to construct Grid infrastructure. These services encapsulate some functions of the underlying Grid infrastructure and make them ease of use. The most important services are Common Message Engine (CME) and single sign on (SSO). CME provides an information exchange channel for facilities and other essential services of framework, including message specification and message transport. A message is defined as a quaternion  $\langle ID, D, S, M \rangle$ , where  $ID$  is the unique message identifier, and  $D = \{d_1, d_2, \dots, d_n\}$  is the destination set of the message with  $d = \langle DID, DURL, DPORT \rangle$  where  $DID$ ,  $DURL$  and  $DPORT$  are the identifier, URL and access point of the destination service respectively, and  $S = \langle SID, SURL, SPORT, EXCEPTION \rangle$  is the message source with  $EXCEPTION$  indicating what to do when a message fails to reach the destination, and  $M$  is the message content that is usually application-specific. CME provides support for message routing and multicasting, which is done by the message transport module. Once a message is received, message transport module resolves it and tries its best to forward the message to the destinations. A destination can be located via its identifier ( $DID$ ) or the pair ( $DURL, DPORT$ ). When  $DID$  is deployed, the transport module exploits facility registry to find out the real location of the destination service. Facility registry can be updated on the fly, thus dynamic service invocation is enabled. The implementation of CME incorporates the idea of Service-Oriented Architecture (SOA), that is, all the modules of CME are implemented as independent Web services. Compared with other message-oriented middleware [12], CME is more flexible and reusable because it inherits the advantages (e.g., platform-independent, loosely-coupled and so forth) inherent in SOA. Therefore, users can easily extend or recombine existing services through the interface provided according to the real world requirement without modifying the underlying system architecture. This is an outstanding feature of CME. In addition, CME provides alternate data transport schemes as stated above. This makes it even more applicable.

SSO is deployed to facilitate system usage. It is implemented as an independent service. Since many literatures [13] have discussed the features of SSO thoroughly, we will not repeat it here.

### 3.2 Facilities

Facilities are some utility services based on the common services supplied. The interconnection of facilities with CME makes facilities as a whole to deliver more powerful and convenient functions for cooperative applications. That's the main difference with the standalone groupware. Two categories of facilities, that are, process management, awareness support, are critical and common to different domains and supplying these facilities will ease the burden of cooperative system development.

#### 3.2.1 Process management facilities

Currently process management facilities include only one service, that is, workflow management service. In our opinions, the control flow in process management should be balanced between human and system to bridge the socio-tech gap [14]. To achieve this purpose, two steps are taken. First, we decouple workflow logic from real business logic. That is to say, workflow just tells right person ("who") to do the right thing ("what") at the right time ("when"). However, how to handle the workitem ("how") is left to users or applications specified by the user to decide. After the given workitem is handled, the actor reports the result explicitly. Based on this paradigm, we devise a set of cooperation patterns as the elementary units for process construction. Each pattern contains a solution to a common business goal. Second, AI method is introduced as does in Pegasus [15]. One outstanding feature of our workflow planner is that efficient interaction with users is supported to keep the balance between full automation and human pro-activeness and to better reflect user's intension. The workflow execution engine in CoDevCore is built on top of CME and so it can easily communicate with the other facilities. For example, the message that a certain workitem is being handled can be sent via CME to awareness support facilities and then perceived by others. With message as a basis, process instance can be adjusted on the fly simply by resending or forwarding the corresponding

messages generated by some workitem. By this means, designers and developers avoid the intensive and trivial work on implementing various function-like business processes as does in traditional workflow management system.

### 3.2.2 Awareness support facilities

Awareness support is one of the most important features of cooperative systems [16]. The most intuitive perspective of awareness support is to provide suitable information about the environment during the cooperation, including people presence, cooperation history, network status, etc. It is widely accepted that providing support for awareness will greatly promote the efficiency of collaboration. CoDevCore aims to cover a wide range of cooperative applications, so awareness support must be supplied. In response to this requirement, cooperative awareness (CA) service is delivered. We'd like to point out the early work on awareness usually exploits very simple and inefficient strategy due to resource limitation and as a result, the power of awareness is not fully opened out. CoDevCore is constructed on top of Grid infrastructure and such a limitation disappears owing to the enormous resources shipped by the Grid. Taking this advantage, we set up a "Poly-Awareness" model [17], which integrates basic information services, situation awareness, process context, user customized information, profiles of person and devices as a whole. With this model, when designing other services, designers can clarify such problems as what information is needed by CA and how to produce the information needed. The interaction between CA and other services makes all of them more powerful.

## 4 Advantage of event-drive service [18]

In order to minimize the inadequate communication between developers of global software development, we argue that event-drive servers can be used because they help resolve the technical issues and they provide support for some organization issues such as role support. The advantage of event-drive service are as following.

### ◆ Flexibility

Event notification servers can provide several advantages when used to support distributed software development. The most important is the decoupling between the sources and consumers of the events. When coupling is decreased, the flexibility of the entire system is improved. For instance, an information source publishes their events without knowing which components are interested in them. Meanwhile, the information consumers subscribe to the events in order to be notified without knowing which component is the source of the event. In this manner, if a new information producer is added to the system, no modification is necessary to it. Furthermore, if a new information consumer wants to be notified about an event, it simply needs to subscribe to it. The information source does and the other components of the system do not need to be modified. In short, the event technology enables a "plug and play" approach to support introduction of new service components [7].

### ◆ Support for informal communication

The physical distance between sites makes it harder for distributed team members to locate remote colleagues and to be aware of their actions. This lack of awareness implies in fewer interactions since they cannot tell what people are up to and if it is appropriate to interrupt. In fact, one of the main problems in distributed software development is the lack of informal communication [19]. In other words, software developers rely also on informal, ad-hoc communication to fill in details, handle exceptions, correct mistakes and bad predictions, and manage the effects of all these changes [19]. As this type of communication is absent in global software development the coordination of these tasks is much more difficult. One possible approach to this problem is to use Instant Messaging (IM) to stimulate informal communication (opportunistic interactions) among workers at different sites. Usually, IM systems provide awareness information about the presence of others. Notification servers have been used as infrastructure to build IM systems [20], in so doing they can indirectly support informal communication.

### ◆ Role support

Roles are used to help the coordination of a large number of developers. Examples of roles are: project managers, team leaders, senior analysts, designers, programmers and so on. These formal roles need different types of information to be properly played. In fact, according to [21] a collaborative system should be able to direct particular information to particular people based on their roles.

Fuchs [22] argues that the same event can be highly important for one user and completely uninteresting for another user in the same situation, or similarly, an event may be important for a user in some situation and irrelevant for the same user in another. Fuchs' work is about public administration, but the similarities to software projects is sufficient that we believe similar conclusions apply.

In general, in an event-based system there is communication: between the information source and the event server and between the event server and the information consumers. There are two types of architectures to implement this communication: pull and push. In a pull architecture, the component interested in receiving the events is responsible for contacting the other component to check for new events. Usually, it is necessary to specify how often one component wants to poll the other. On the other hand, in a push architecture the component that wants to send the events "pushes" these events onto the others. In other words, the second component is "called" by the one that is sending

events. In this case, these components need to implement some kind of standard interface to be invoked by the other components.

In order to support distributed software development both architectural models should be supported due to the different roles involved in a software development activity. For example, a push model is important in order to inform the manager of an invalid or problematic state. In real-time scenarios, it might be imperative to receive a notification of events as soon as they occur. On the other hand, a pull model is important in a mobile environment where the information sources can be disconnected. In this case, the events would be lost if they were not recorded for later delivery.

#### ◆ Network problems

Since wide-area networks are often slow and unreliable, i.e., network packages are lost due to problems in the transmission. For several applications, this is not a problem, however in global software development the events flowing in the network encapsulate critical data necessary to improve coordination of activities, communication, etc. Event meta-information could be used to avoid networks problems. By meta-information, we mean any additional information about an event that is recorded by the information source or event server and sent alongside with the event. Therefore, this meta-information is available to the information consumers, in addition to event information. Typical examples of meta-information are the type of the event being sent, the information source that sent the event, the timestamp when the event was sent by the information source and so on. A flexible mechanism to support event meta-information could be used to support additional services such as guaranteed delivery and security, therefore alleviating the problems of the wide-area networks.

## 5 Example scenario of event service

The following scenario illustrates how an event server might be used to support distributed software development. Its goal is to support developers keeping them continuously aware not only of which artifacts are changing, but also of the impact and severity of those changes. We currently want to implement the following scenario.

In this scenario, several distributed users are working on different, but dependent, parts of the system. For example, White is developing a component called "Reader.java" in Seattle, USA. This component depends on another component "System.java" being developed by another developer, Liu, in Guangdong, China. The dependency between these components implies that White needs to be notified when component "System.java" is modified. Otherwise, changes in System.java may "break" his code. In other words, White adds this component to his private workspace; therefore this system will send a subscription to an event server specifying that White be notified about changes (events) for that component. This subscription is sent to the event server located in Seattle, which forwards it to all other event servers of the company including one in China. Meanwhile, Liu, who has been working in China to complete his task, checks in "System.java" into the Configuration Management (CM) system. The CM system is instrumented in a manner such that it sends events to the event server, when components are checked in or checked out. In this case, a new event viz. "*component System.java checked in by Liu at 10:03 AM*" would be generated. The event server in China adds information about the current location to the event and then forwards it to the server in Seattle. The next morning, White's workspace would have an indication of changes to the component "System.java". John would be able to figure out the appropriate action to be taken: change his code, contact Liu, etc.

There are several important points in this. First, the event server must support Internet-scale notifications in order to deliver the information among the distributed sites. Second, the event server must support push and pull. Third, in this scenario, events are being treated as strings without any formal structure. However, in reality, events should be semi-formally structured in order to (1) facilitate their processing, and (2) provide support for error detection. It is also necessary to have some mechanism for defining event structure and type.

## 6 Conclusions

Recently, supporting global software development begins receiving more and more attention [11, 18], CoDevFrame borrows some ideas from them. In global software development, several problems arise due to the physical, social and cultural difference between the developers. For example, communication is much more difficult because of language barriers, informal communication among the team members cannot happen and trust is more difficult to achieve.

The goal of this paper was to present event-drive servers based on Grid and Web services as an important and useful technology to support global software development. The way deployed is to develop some generic common services on top of Grid and harmonize the interaction between these services, they provide a good framework to the development of collaborative tools. For examples, technical problems that collaborative tools face in widely distributed scenarios can be solved using these servers. Finally, it is important to note that event-drive servers are one

mechanism to distribute events across the sites. They provide the infrastructure for a variety of tools to be built upon, such as process visualization [23] or event reasoning [24].

## References

- [1] Grudin, J. CSCW: History and Focus. *IEEE Computer* 27(5):19-27, May 1994.
- [2] Herbsleb, J. and Moitra, D. *Global Software Development*, IEEE Software, vol. 18, Issue 2, pp. 16-20, March/April 2001, Special Issue in Global Software Development.
- [3] Foster I., Kesselman C. and Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 2001, 15 (3): 200-222
- [4] Huhns M. N. and Singh M. P. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 2005, 9(1): 75-81
- [5] Hilbert, D., Redmiles, D. Extracting Usability Information from User Interface Events. *ACM Computing Surveys*, accepted and to appear, Vol. 32, No. 4, December 2000.
- [6] Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kotting, B., and Schaaf, M. Merging Project Planning and Web-Enabled Dynamic Workflow Technologies, *IEEE Internet Computing*, pp. 65-74, May/June 2000.
- [7] Cugola, G.; Di Nitto, E.; Fuggetta, A. The JEDI eventbased infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9), pp. 827 –850, Sept. 2001.
- [8] Kantor, M., Redmiles, D. Creating an Infrastructure for Ubiquitous Awareness, Eight IFIP TC 13 Conference on Human-Computer Interaction (INTERACT 2001), Tokyo, Japan, pp. 431-438, July 2001.
- [9] Fitzpatrick, G., Mansfield, T. et al. Augmenting the workaday world with Elvin, In *Proceedings of 6<sup>th</sup> European Conference on Computer Supported Cooperative Work*, pp. 431-450. Copenhagen, Denmark, Kluwer, September 12-16, 1999.
- [10] Carzaniga, A., Rosenblum, D., and Wolf, A.. Design and Evaluation of a Wide-Area Notification Service. *ACM Transactions on Computer Systems*, 19(3), 332-383, 2001.
- [11] Jinlei Jiang, Meilin Shi and Bo Jing, CoFrame: A Framework for CSCW Applications Based on Grid and Web Services. 2005 IEEE International Conference on Web Services, Jul. 2005, pp.570-577.
- [12] Sun MicroSystem. Java Message Service Specification. <http://java.sun.com/products/jms/>
- [13] Welch V., Siebenlist F., Foster I. et al. Security for Grid Services. In Proc of the 12th IEEE International Symposium on High Performance Distributed Computing, Seattle, 2003, 48-57
- [14] Ackerman M. The intellectual challenge of CSCW: The gap between social requirements and technical feasibility. *Human-Computer Interaction*, 2000, 15(2&3): 179-203
- [15] Gil Y., Deelman E., Blythe J., Kesselman C. and Tangmunarunkit H. Artificial Intelligence and Grids: Workflow Planning and Beyond. *IEEE Intelligent System*, 2004(1/2): 26-33
- [16] Dourish P. and Bellotti V. Awareness and Coordination in Shared Workspaces. In Proc of International Conference on Computer Supported Cooperative Work, Toronto, Canada, 1992, 107-114
- [17] Li Yushun, Gong Neng and Shi Meilin. A New Collaborative Awareness Model and its Application. In Proc Of 8th International Conference on Computer Supported Collaborative Work in Design, Xiamen, China, 2004, Vol. 1: 53-58
- [18] C.R.B. De Souza, S.D. Basaveswara, and D.F. Redmiles. Supporting Global Software Development with Event Notification Servers. *Proceedings of the ICSE 2002 International Workshop on Global Software Development*, 2002.
- [19] Herbsleb, J. and Grinter, R. Architectures, Coordination, and Distance: Conway's Law and Beyond, *IEEE Software*, September/October, 1999.
- [20] Cabrera, L. F., Jones, M. B. and Theimer, M. Herald: Achieving a Global Event Notification Service, In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Elmau, Germany. IEEE Computer Society, May 2001.
- [21] Steinfield, C., Jang, C.Y., and Pfaff, B. Supporting Virtual Team Collaboration: The TeamSCOPE System. In *Proceedings of ACM GROUP Conference*, pages 81–90, Stockholm, Sweden, Nov. 1999.
- [22] Fuchs, L.. AREA: A cross-application notification service for groupware. In *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work*, pp. 61–80, Copenhagen, Denmark, Kluwer, September 12-16, 1999.
- [23] Cook, J. Internet-based Software Engineering Enables and Requires Event-Based Management Tools, *Proceedings of the 3rd Workshop on Software Engineering over the Internet*, At International Conference on Software Engineering, Limerick, Ireland, June 2000.
- [24] Cohen, D., Feather, M. S. and K. Narayanaswamy. An Event-Based, Reactive, and Extensible Infrastructure for Distributed Collaboration, In *Proceedings of the Workshop on Coordinating Distributed Software Development Projects*, IEEE Intl. Workshops on Enabling Infrastructure for Collaborative Enterprises, June 1998.