5J−1 Information Exchange and Security in a Collaborative Development Support System for Object-Oriented Software

Manuel J. PECE MONTENEGRO, Ken'ichi KAKIZAKI and Seiichi UCHINAMI
Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

1 Introduction

Although the object-oriented paradigm has well known advantages, the existence of inheritance, function overriding and polymorphism can be an obstacle for an efficient software development. Moreover, for large scale systems development, the concept of collaborative development support has emerged as an important issue. Nowadays it is common for the participants in a development group to be working in different host machines placed in physically distant locations, and hence the exchange of information needed for cooperation becomes complicated. A support system that eases the communication is needed [Oka91].

In this report we discuss information exchange and security issues in a collaborative development support system for object-oriented software.

2 Support System Overview

This section presents the design overview of our collaborative development support system for object-oriented software [Pec92]. We have chosen the C++ as the object-oriented language to be supported.

Object-Oriented Development Support We want to offer to the developer all the information that is relevant when developing object-oriented software. The information is distributed among various class definitions. The support system gathers this information and shows it in a friendly way. A class browser is used as the user interface. The whole class hierarchy graph is displayed in an area of the browser, as indicated in fig 1.

Collaborative Development Support The developers need to exchange information regarding to what is being done in the development group, without neither being aware of the network characteristics nor making any extra effort. The system also supports the construction of a compilation environment by providing the most recent versions of the files needed for a compilation, according to predefined compilation dependencies. The version management follows the checkin/checkout model. For security reasons, there is also an access control of source files to regulate intersubgroup file access.

The support that we claim for is not only for the developers that are realizing the programming work, but also for the project manager who is directing the process.

Documentation Support The need of producing an accurate technical documentation for testing, debugging, extending and maintaining large scale systems becomes even more important in object-oriented software development, due to the high reusability. The system supports two kinds of documentation: an interactive documentation in the form of on line help and a hardcopy one.

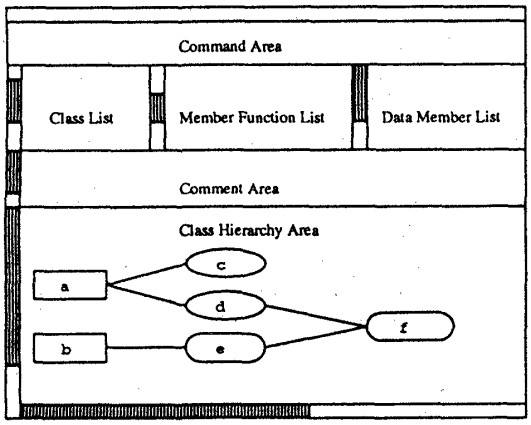

Figure 1: The class hierarchy displayed in the browser

3 Collaborative Development Support

3.1 Information exchange

With the widespread use of workstations and networks, developers participating in a project can be working in different host machines, placed in distant locations. Because of this, the exchange of information among the participants in a development group has become very difficult. Furthermore, this information exchange is even more important in object-oriented projects than in traditional programming, due to the inheritance capability [Kom91].

We want the developer to be able to get information about the classes that the other coparticipants are creating, in the same way he gets information about his own classes. The information must come from the most recent version of the class definitions.

In order to understand the structure of the software project as a whole, the user in a certain host machine automatically receives information from other host machines through the network, in a client/server interaction. The server analyzes the source files and extracts the relevant information, which can then be exchanged independently from the files. The client browser displays that information about the whole system being developed, including those classes defined by other developers and the common classes that existed since the beginning of the project. For the purpose of telling who is the developer of the class, the system uses different shapes to represent the classes in the hierarchy graph. This is shown in figs. 1. and 2.

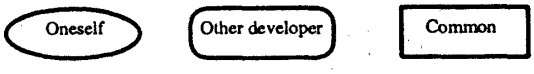

Figure 2: Class developer representation

The exchange of information we design consists not only in receiving information about other developers' classes, but also in a participant communicating to the others what he is going to do. There are few problems with the classes that have been clearly defined and specified during the analysis phase, but problems such as class name definition as likely to happen with support classes added later, since their definitions are not known by all the developers. When a developer wants to use a certain class name or global variable name, it is possible that before he completes the definition, other developer has already used the same name (i. e. duplication problem).

To solve this problem we propose a mechanism that allows to declare in advance the class name or global variable name, so that the use of already declared names is forbidden by the system. That function is necessary to maintain the project consistency, regarding not only to class name problem of C++, but also traditional C problem with global variables.

3.2 Security

Even in a collaborative environment, there can be occasions in which the project is being developed by various different organizations or subgroups (e.g. different sections within a company or different companies working in collaboration) where it is necessary to restrict the direct access to the source files of a subgroup by a member of another subgroup [Asa92]. We think that the support system must provide not only information exchange facilities, but also the possibility for each subgroup to clearly precise if its information is not to be offered, and then restrain the access to that information.

In the project definition, all the subgroups participating in the development are defined and then the access control to the files of each one is set. There are three types of access restriction : generally allowed, generally restrained and absolutely restrained. For the first two, the file creator can set a different restriction type for that file. The third one can not be accessed by any outsider. Fig 3 illustrates the different access restrictions among three subgroups. In the Subgroup A, which is generally allowed, one file has been set as not allowed. In the subgroup B, which is generally restrained, one file has been set as allowed.

As a consequence of using this access control, information that would be easy to understand by looking into the source file is no more available, and hence some other kind of information must be supplied. This task will be fulfilled by the interactive documentation that we cite in the support system overview.

3.3 Compilation environment construction

Developing software does not consist only in writing source programs. Tests of those programs and construction of the system by integrating them are also necessary. To do this, access to the information about other developers' classes is not enough. It is necessary to compile and link the programs, in which case various files, distributed in different host machines, will be needed. The construction of a compilation environment that brings together the most recent versions of the files needed for the compilation (header files, source files, libraries, object code, etc.) according to predefined compilation dependencies, and compile and link them is required.

In general, source files are provided for the compilation, but when the source files are not accessible, due to security reasons, object files are provided by the system. This enables compilation without breaking the security restrictions.

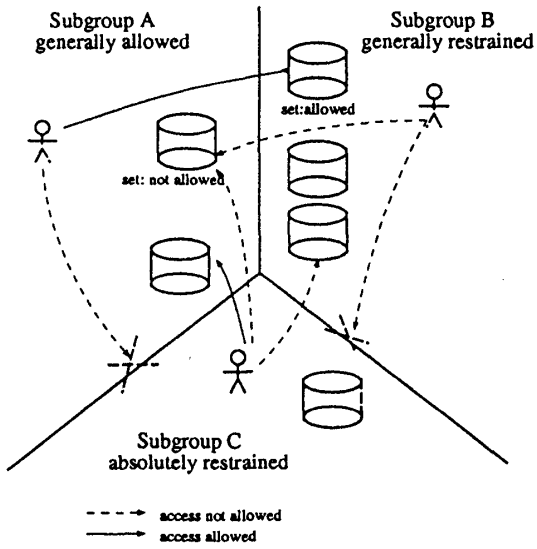

Figure 3: Access control to source files

4 Conclusions

We have designed a support system for collaborative developing of large scale object-oriented software. At the present stage, the object-oriented development support and documentation support functions have been implemented.

We present a support function that supplies the developer with all the interface information about other developers' classes. Since this information can be handled independently from the source files, the information exchange can take place, even if the developer can not directly access certain source files for security reasons. In this way, information exchange does not compromise the security.

Because the system provides all the relevant information distributed among various class definitions, it enables the utilization of all the object-oriented programming paradigm potential.

References

[Asa92] Asami, H., Miyawaki, M., Tanaka, K. and Fukuyama, S.: "A Security System for Distributed Software Enviroment", SIG Notes DSP-54-9, IPSJ (1992).

[Kom91] Komiya, S.: "Software Collaborated Design Process Modelling and Its Intelligent Support Method", SIG Notes SE-83-15, IPSJ (1991).

[Oka91] Okada, Y., Iida, H., Inoue, K., Torii, K., Nagaoka, W., Umumoto, H. and Sakai, M.: "Process and Communication Models for Distributed Software Development", SIG Notes SE-82-1, IPSJ (1991).

[Pec92] Pece-Montenegro, M. J. and Kakizaki, K.: "Requirement Analysis and Design of Object-Oriented Development Support System for Group Development", SIG Notes SE-87-8, IPSJ (1992).