XCS System: A New Architecture for Web-Based Applications*

Yijian Wu and Wenyun Zhao

Software Engineering Laboratory, Fudan University, Shanghai, 200433, China
yijian_wu@163.com

Abstract. This paper puts forward a model of Extended Client/Server (XCS) system, which is based on traditional Client/Server Architecture and borrows such merits from Browser/Server system as integrated code maintenance at Server-end and installation-free at Browser-end, etc. In an XCS system, Client is dynamically configured and automatically updated, and Server is able to commit self-health-checks and upgrade without shutting service down. XCS system is designed for secure application for authorized users, considering security extensions, and able to be extended to distributed application system.

1 Introduction and Assumptions

The Extended C/S System (XCS) inherit as many as possible the advantages in C/S technology and B/S technology. Our destination is to build a system that end-users don't have to care much about the installation and the server can be powerful and flexible to the max extent. We assume:

First of all, the logic of the application is so complex that it is not practical to immigrate all Client logic/function to the Server-end.

Second, all of our end-users should be authenticated before they are able to have access on the Server. In other words, the system is secured and only accessable to authorized users, instead of to all anonymous requests.

Third, all data transfer should be secured. Data should be classified. Sender of data should be responsible for what has been sent.

2 XCS Architecture Overview

The main architecture of XCS, depicted as Figure 1, can be divided into three layers, Client, Server and Resource. Basically there are two servers – Web Server and Application Server – in Server layer, and both are protected by a Firewall, which relays requests and responses between the Server layer and the Client layer.

* Supported by the National High Technology Research and Development Program of China (863 Program) under Grant Nos. 2002AA114010; 2001AA113070. Also supported by Shanghai Technology Development Foundation (No. 025115014).

M. Li et al. (Eds.): GCC 2003, LNCS 3033, pp. 1046–1050, 2004.

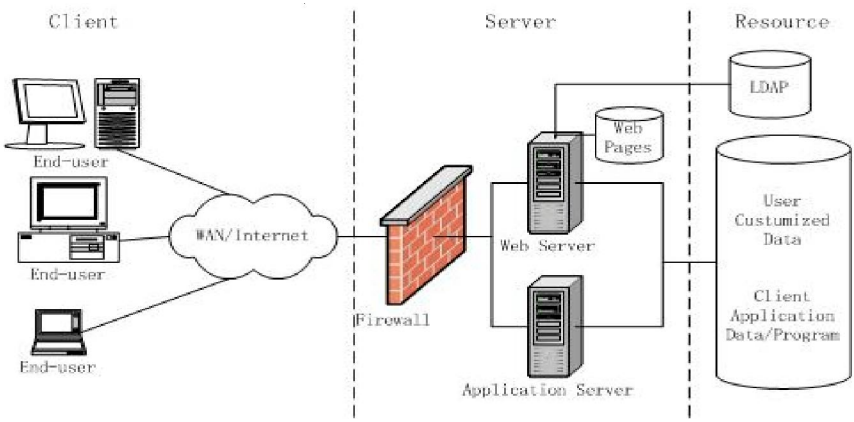

Fig. 1. Overview of XCS Architecture

Mention that there are Logic modules in the Web Server to provide information services to clients, such as User Authentication, Client Version Checking, Customize Configuration, etc. It consults the Resource tier for data needed.

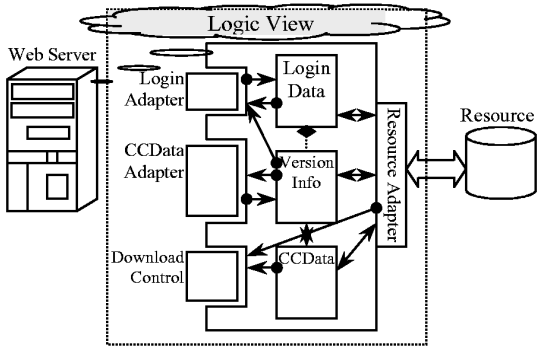

Fig. 2. Logic modules in Web Server

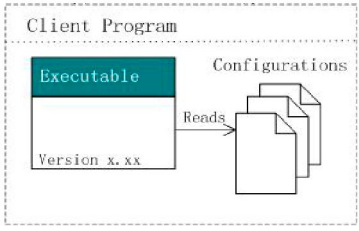

Fig. 3. Client Program

The Logic modules are encapsulated and predefined in various systems. To deal with the variations, a Resource Adapter is used to fetch data from Resource Tier, as shown in Figure 2. This can be implemented as an Adapter Pattern and a Strategy Pattern [GaH94].

The Application Server is an independent application that runs all business-related logical modules.

Resource layer is an organized collection of all kinds of data within the whole system, such as metadata and up-to-date client program, ready to be sent to end-user if necessary.

We devide Client program into Executable part and Configuration part, as shown in Figure 3. The client subsystem maintains information of executable version.

Configurations can be stored in Resource Layer in Server subsystem and/or transferred to end-user with the Executable part.

After all, a Metadata Editor (not presented in Fig 1) is defined data such as names of properties needed to describe configuration data for a specific application.

3 Logic and Detailed Structure

Primary business logic is dealt between Client Application and Application Server, after Web Server and User Browser have initiated communication. Client is updated automatically and Application Server can be upgraded without shutting service down.

3.1 Logic and Workflow

Before business-specific communication protocol is activated, common communications and authentication procedures are working. A first-time user has to visit the website and pass the authentication. User Authentication Module consults Resource layer to verify username and password.

Version check procedure checks user's local information to decide whether the latest Client Application is correctly installed. If so, system will be ready to fetch user configuration data; otherwise, an error page will be returned.

User configuration data is kind of Customized Configuration Data (CCData) in Resource. CCData contain optional settings of specific users, including GUI style and business-specific features, which can be of great variation. A metadata configuring subsystem is used to maintain the variation efficiently. User commits configurations to Resource through a web page before using the Client Application. A list of CCData is stored in Resource and will be sent to end-user as needed.

3.2 Detailed Structure: Subsystems

The whole system can be roughly divided into the following functioning subsystems: **BS Login Subsystem**, which consists of Browser (End-user), Web Server and login related web pages, User Authentication Logical Module and User Authentication Data; **Client Application Management Subsyste**m, which consists of Version Checking Module, Release Control Module, Client Application stored in Resource layer and CCData Modification Module; **Metadata Management Subsystem**, which consists of a User Customized Data Definition Tool and Client Application Release Tools; and **Business Logic Subsystem**, consisting of Client Application, Application Server and Application Data.

The subsystems are designed to meet the following requirements:

- Secured login and communication
- Client Application version control
- Integrated configuration control
- Easy way of upgrading client program

The machenism that keeps client program up-to-date and application server 24x7 working is as the following. Application Server with Release Server structure is depicted in Figure 4. Server is currently deployed on computer *SVR*. Release Server *REL* will do the following to deploy a new version of Server and Client:

S1. Fetches a)version b)listening port of the server c)server IP of current server *SVR* from Resource. If the version is newer than that of the candidate program, terminates with an exception. Otherwise marks it out-of-date in Resource layer.
S2. Selects a Server *DEST* that the new server shall be deployed (*SVR* is default).
S3. Sends an Agent to *DEST* to negotiate an available port *P* on *DEST*, and waits.
S4. Agent returns to *REL*, carrying the Result (success with *P* or failure with a short reason). If not succeed, Release process terminates with an exception.
S5. Checks system status on *DEST*. If no service is on, put the new version on *DEST* and start it up (with port *P*). Else create new application working space on *DEST* and put the new version in it and start it up (with port *P*).
S6. If *DEST* starts up normally, updates current system information in Resource with new information, including version, new IP and new port, and kills *SVR*.

An out-of-date service, marked in S1, cannot accept any new connections and will automatically shut itself down as soon as no active client is connected. *SVR*, *DEST* and *REL* may be the same computer, but it is recommended that *REL* be independent from *SVR* or *DEST* and especially reliable. Other redundant measures can be taken to increase the reliability of the whole system, such as backup *REL* in case it fails.

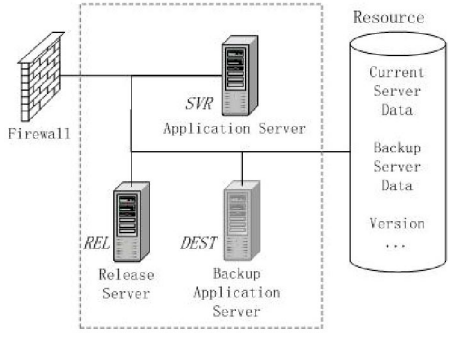

Fig. 4. Release Server Structure

4 Discussion and Conclusion

Resource can be either centralized or distributed. [MiK01] describes a Mobile Agent based solution for Network Measurement, which illumes a Mobile Agent solution to improve performance of distributed system.

Although we defined a metadata definition tool to achieve extensibility, we cannot guarantee all kinds of extensions. One solution is that the architecture turns to be domain-specific with only some specific logical modules, which proves to be more efficient and practical in engineering.

We have to take it for granted that behind the Firewall, where all Server works, are trustworthy. To overcome the trust crisis, Application itself should be strength enough against security attack[How02].

XCS system integrates some merits in both BS and CS applications. It automatically and intelligently delivers new client application to out-of-date clients, thus keeps *almost* all clients are up-to-date. The Metadata Management Subsystem offers a fail-safe mechanism to upgrade Server program without shutting service down. Secure XCS system puts emphasis on not only secure network transport, but application security and robustness as well.

References

[GaH94] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Chinese Edition, CMP, 2000, p92,p208

[How02] M. Howard, *Writing Secure Codes*, S. Chinese Edition Translator Y.J. CHENG, et al, CMP, 2002,7, p31

[MiK01] A. Michalas, T. Kotsilieris, et al, *Enhancing Performance of Mobile Agent based Network Management Applications*, Sixth IEEE Symposium on Computers and Communications, 2001