Faculdade de Engenharia da Universidade do Porto

Generic Server implementation for Computing Resources Administration

Tiago Rodrigues Vieira Batista

Project developed in the scope of the
Integrated Master in Informatics and Computing Engineering

Supervisor at FEUP: Prof. Jaime Villate

July 2008

Generic Server implementation for Computer Resources Administration

Tiago Rodrigues Vieira Batista

Relatório de Projecto realizado no Âmbito do
Mestrado Integrado em Engenharia Informática

Aprovado em provas públicas pelo Júri:

Presidente: Pedro Alexandre Ferreira do Souto

_____

Arguente: António Amorim

Vogal: Jaime Enrique Villate Matiz

17 de Julho de 2008

Abstract

The organization of CERN is designed for the very specific needs of a scientific community. There are thousands of users within the organization worldwide and the requirement of accessibility worldwide of CERN applications is greater than ever. So, in respecting these constant demands the Computer Centre Database that handled the computer account information for the IT-managed systems and which served for more than 15 years has been replaced by the Computing Resources Administration (CRA) Web application. This Web application does the administration of many resources for users and services providers. These resources include e-mail, e-groups, space quota, Linux and Windows accounts, etc. But, CRA does not work alone and it requires that whenever a change resource request is made in CRA, the actual change has to be forwarded to the computer service responsible for the resource. So, this implies that the changes of the definitions and permissions of the different resources and the actual changes in the computer services have a delay. For instance, if the password of the NICE Windows account is reset, the new password only can be used after 20 minutes. This delay is caused by the polling mechanism inherited from the Computer Center Data Base that had a few drawbacks (services are obliged to support an Oracle database connection and delays between changes in CRA and actual changes in the service are introduced as a consequence).

This project has been created in the context of the CRA project and for two important reasons. Firstly the existing synchronization of definitions and permissions of the resources between the CRA application and the different computer services has a significant delay and secondly the need to replace the existing 'Simba List' Web Application which is available in the mail service by an easier and more powerful application called 'E-groups'. Both applications allow creation of email lists, but e-groups can be used for other purposes as well. Currently Simba Lists and E-groups run in parallel and are synchronized between the CRA and the mail service by a database process, once a day. The two reasons have a common problem the time taken by the process of synchronization is not considered reasonable.

Thus, it had been necessary to implement a mechanism capable of informing the services in a reliable, flexible and quick way. The solution was the implementation of a server capable of performing the polling of changes and the use of Web services to communicate with the different CERN computing services (e.g. mail service).

This document will explain the several technologies including libraries and APIs that have been used. It will also provide a general view and some background knowledge of the project.

Further chapters will discuss project requirements and server architecture. Also mentioned are some implementation details including server configuration and the deployment process.

And finally, conclusions that have arisen from this project as well as future perspectives will also be presented.

Resumo

A organização do CERN é projectada para as necessidades específicas de uma comunidade científica. Dentro da organização existem milhares de utilizadores e é necessário o acesso mundial à maior parte das aplicações do CERN. Então, depois de mais de 15 anos de serviço, a base de dados do centro de Computação, que tratou da gestão de sistemas de Tecnologias de Informação, foi substituída pela aplicação Web de administração de recursos de computadores (CRA). O CRA faz a gestão de recursos, para utilizadores e provedores de serviços. Estes recursos incluem o e-mail, *e-groups*, quota de espaço em disco, contas Linux e Windows, etc. Mas o CRA não funciona sozinho e é requerido que sempre que existe um pedido de alteração a um recurso no CRA, a alteração seja enviada para o serviço computacional responsável pelo mesmo. Portanto, isto implica que as alterações as definições e permissões dos diferentes recursos e as mudanças reais nos serviços computacionais têm um atraso. Por exemplo, se a palavra-chave da conta NICE Windows for alterada, a nova palavra-chave só pode ser utilizada após 20 minutos. Este atraso deve-se ao mecanismo de *polling* herdado da base de dados do Centro de Computação que tem alguns inconvenientes (os serviços são obrigados a suportar uma ligação a base de dados Oracle e como consequência são introduzidos atrasos entre mudanças reais no CRA e alterações nos serviços).

Este projecto nasceu no contexto do projecto CRA e é baseado em duas razões principais. Primeira razão é que a sincronização das definições e permissões dos recursos entre o CRA e os diferentes serviços informáticos tem um atraso significativo. A segunda razão é a necessidade de substituir a existente aplicação *Simba Lists* disponibilizada pelo serviço informático de *mail* por uma aplicação muito mais fácil e poderosa chamada *e-groups*. Ambas as aplicações permitem criar listas de e-mails, podendo o *e-groups* ser usado também para outros fins. No momento, *Simba Lists* e *e-groups* correm em paralelo e são sincronizadas entre o CRA e o serviço informático de *mail* através de um processo de base de dados, uma vez por dia. As duas razões têm um problema em comum, ou seja o tempo despendido com o processo de sincronização não é considerado razoável. Portanto, foi necessário desenvolver um mecanismo capaz de informar os serviços de forma confiável, flexível e rápida. A solução foi a implementação de um servidor capaz de fazer *polling* das alterações e o uso de *Web Services* para comunicar com os diferentes serviços computacionais do CERN.

Durante este documento serão explicadas as várias tecnologias, bibliotecas, APIs utilizadas e dada uma visão geral do contexto em que o CRA se insere.

Após essa visão essencial, serão apresentados os requisitos do projecto, a arquitectura do servidor e alguns detalhes da implementação, como por exemplo a configuração e instalação do servidor.

No final, serão apresentadas algumas conclusões do trabalho desenvolvido, bem como as perspectivas de futuro.

Acknowledgements

First of all, I would like to thank to my supervisor Wim Van Leersum, who is a wonderful mentor, for his infinite patience and for all the knowledge he has passed onto me.

Appreciation words to our wonderful family that is the IT-AIS section at CERN.

I also have to appreciate the collaboration and orientation of Professor Jaime Villate.

Lastly I want to thank, my girlfriend Cassilda, Joana, Niece Sara, brother in law Yetin and my friends Luís and Pedro for all the support and love they have given to me whilst writing this project.

Tiago Rodrigues Vieira Batista

Conventions Used in This Report

Italic is used for:

- Pathnames, filenames, and application names
- Packages and Class names
- New terms or concepts
- Internet addresses, such as e-mail addresses, domain names and URLs
- Quotes

Bold is used for:

- Extra emphasis
- Parameters names

Italic with **Bold** is used for:

- New terms or concepts with extra emphasis

Contents

Figure Index

Table Index

Introduction

1.1 CERN Presentation

The European Organization for Nuclear Research [Abo07], commonly known as CERN is the world's largest particles physics laboratory in the world. It sits astride the Franco-Swiss border near Geneva.

The name CERN is derived from the French "**C**onseil **E**uropéen pour la **R**echerche **N**ucléaire" or European Council for Nuclear Research, a provisional body founded in 1952 with the mandate of establishing a world-class fundamental physics research organization in Europe.

When the organization officially came into being in 1954, the Council was dissolved, and the new organization was given the title European Organization for Nuclear Research, although the name CERN was retained.

CERN was one of Europe's first joint ventures and currently includes 20 member states and scientists from several nationalities and cultures from around the world. The unit focuses its efforts to study the building blocks of matter and the forces that hold them together. CERN was created to provide them with the necessary tools. These tools are called accelerators whose purpose is to accelerate particles to almost the speed of light and use detectors to make those particles visible.

CERN's accelerator complex is built around three principal inter-dependent accelerators. The oldest the Proton Synchrotron (CPS) which was built in the 1950s and was briefly the world's highest energy accelerator.

The Super Proton Synchrotron (SPS) which was built in the 1970s and was the scene of CERN's first Nobel Prize in the 1980s. The large Electron-Positron collider (LEP) came on stream in 1989. It was the Laboratory's flagship research machine until 2000.

Currently under construction, inside the same tunnel as LEP, is the Large Hadron Collider (LHC), which is scheduled to begin operation in August 2008. The LHC is expected to become the world's largest and highest energy particle accelerator, and probably the biggest machine ever built by man.

Our current understanding of the Universe is incomplete. Theories we currently use to describe it leave many unsolved questions. The reason why elementary particles have mass and why their masses are different are among the most perplexing questions. The answer may be the so-called Higgs mechanism. The Higgs field has at least one new particle associated with it, the Higgs boson. If such a particle exists then it will be detected by the LHC. The observation of this could confirm the predictions and 'missing links' in the standard model of physics.

It is also relevant to mention that the World Wide Web (WWW) born at CERN is a result of the project ENQUIRE initiated by Tim Berners-Lee and Robert Cailiau in 1990. It was

developed to improve and speed-up the information sharing between physicists working in different universities and institutes all over the world.

More recently, CERN has become a centre for the development of Grid computing, hosting among others the *Enabling Grids for E-science (EGEE)* and *LHC Computing Grid projects.*

The World Wide Web was CERN's response to a new wave of scientific collaboration at the end of the 1980s. The GRID is the answer to the need for data analysis to be performed by the world's particle physics community. With the LHC CERN experiments will have to handle petabytes (1 PB = 1015 GB). Each year the LHC experiments will produce enough data to fill a stack of CDs 20 km tall) of information that can not even be handled by the most advanced computers currently available. The proposed solution is the GRID, a very powerful tool tying computing resources distributed around the world into one computing service for all requesting applications. A rapid and natural consequence of the GRID has been the development of a Europe-wide database of mammograms for epidemiological, as well as teaching purposes. Today, 32 MB are needed to store a mammogram image, giving a total of 128 Mbytes per person per visit.

1.2 IT Department

CERN is organized in seven Departments, each one with a different scope of responsibilities. Several Groups constitute every Department and each Group has many Sections.

CERN's needs for information technology services can be grouped into six broad areas, of which the IT Department [CER04] has responsibility for the first four:

1. General-purpose computing
2. Administrative computing
3. Physics and engineering computing
4. Consolidation, coordination and standardization of computing activities
5. Physics applications (e.g., for data acquisition/offline analysis)
6. Accelerator design and operations.

The last two (5&6) areas are the responsibility of the Physics and Accelerator and Beans Departments.

It is in the IT department that this project/document has been developed. More specifically within the Administrative Information Services group and the Finance, Procurement & Foundation section (AIS-FPF).

1.2.1 Administrative Information Services Group (AIS)

The Administrative Information Services group [CER081] of the IT Department has the responsibility for all administrative applications and corporate data at CERN, covering the following services:

- The functional requirements of administrative applications;
- Purchase, develop, implement, and maintain administrative applications and all tools required for optimal exploitation of the corporate information;
- Support users of these applications, prepare user documentation and provide training;

Following a strategy to provide a unique, coherent and integrated environment for all applications, tools, and documentation, virtually all functionality is provided through a Web interface.

1.3 Computing Resources Administration (CRA)

Computer Resources Administration [Cer05] is an Administrative Information Services (AIS) Web application which handles the computing services resources for the IT-managed systems, storing the definitions of the resources and permissions.

CRA has functionalities for several kinds of user:

- End-users: user details, accounts configuration, space quota request, e-groups[1].
- Services providers: service details, accounts configuration and yellow pages[2];
- Helpdesk: user information, user accounts management and user space quota management;
- Administrators have full privileges in the system.

To improve computer security CRA has integration with personnel data (the HR database). Integration with the HR database allows blocking resources automatically once a person is no longer affiliated with CERN, notifying the supervisor, and tracking other responsibilities, besides the ownership of accounts that need to be transferred.

Support is made simpler by the automatic resource expiration that is mentioned above. It replaces the yearly manual account-review process. Also the application maintenance is simplified by supporting only one user interface - the Web.

1.3.1 Generic Server implementation for Computing Resources Administration

The aim of the project was the implementation of a replacement of the polling mechanism to inform the services of the changes made in CRA. Before, all changes that needed to be processed by the services were stored in an Oracle table with a status flag (P=Pending, A=Acknowledged). This polling mechanism was inherited from the Computer Center Data Base but this system had a few drawbacks (services are obliged to support an Oracle database connection and delays between changes in CRA and actual changes in the service are introduced as a consequence). The new mechanism allows the synchronization based on Web services. The polling server has 3 main parts: the server that is responsible for reading the configuration file, the Agent Manager that has the objective of creating and starting new Agents per computer Service and the agents themselves which call the actual web services. The modifications on the CRA objects (e.g. e-groups, external e-mail accounts) are registered in an Oracle table by triggers. The Agent looks for new or updated records on the modification and notification tables and compare them to know if the object needs to be synchronized or not. Whenever an object needs to be synchronized, the Agent should call the Web service passed by the Agent Manager and obtained from the configuration file. If the synchronization succeeds a new record will be added to the notification table. If not, the

[1] Electronic groups composed by CERN members and external CERN members.

[2] Search engine for services providers.

Agent should keep trying with a delay calculated using the configuration parameters. It is required that one is able to stop the server at anytime without ending in an inconsistent state.

1.4 Report Organization and Subjects

This report is organized in 6 chapters, with a structure that tries to provide a good understanding of the matters discussed within.

The first chapter introduces CERN and its essence, describing its history, mission and future goals. Also described are the departments/groups/sections and their respective importance, as well as a small view of their function.

The second chapter will review the technological context of the project. This involves a description of several technologies involved in the development of this project.

The third chapter will present the main requirements of the application and the architecture according to the functional, logic and physical perspective.

The fourth chapter will elaborate a detailed description of the implementation of the new CRA Server.

The fifth chapter will discuss the quality of the written code and code inspections. The tests performed on the server will also be described and evaluated.

And finally the sixth chapter is designated to promote some conclusions and future perspectives of this project.

Technological Context

In order to understand this project and the upcoming chapters some of the technological background of the CRA and other support APIs are necessary. This will attempt to improve the comprehension of what will be described in this report, when specific examples of the usage of these technologies will be exposed.

The most relevant technologies for the context of the work described in this report are:

- Apache Struts
- Web Services
- Apache Log4j
- Oracle

1.1 Apache Struts

2.1.1 Introduction

Apache Struts [Apa08] is a free open-source framework for developing Java Web applications. The framework is based on standard technologies like Java Servlets, JavaBeans, and XML. Struts separate concerns in a software application by using a Model-View-Controller (MVC) architecture.

The framework provides three key components:

- A "request" handler provided by the application developer that is mapped to a standard URI;
- A "response" handler that transfers control to another resource which completes the response;
- A tag library that helps developers creates interactive form-based applications with server pages.

2.1.2 Model-view-controller

In software engineering the Model-view-controller (MVC) [Mod08] is a commonly used architectural pattern. This pattern isolates business logic from user interface, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other. The Model represents the information (the data) of the application and the business rules used to manipulate the data. The data access can be done by standard technologies, like JDBC and EJB, as well as most third-party packages, like Hibernate, iBATIS, or Object Relational Bridge. The View represents the page design code, corresponding to elements of the user interface such as text, checkbox items, and so forth. This page design code is normally done with JavaServer Pages, including JSTL and JSF, as well as Velocity Templates, XSLT, and other presentation systems. The Controller is a component provided by the framework and represents the navigational code that manages

details involving the communication to the model of user actions such as keystrokes and mouse movements.

The following picture shows the Model-view-controller architectural pattern:

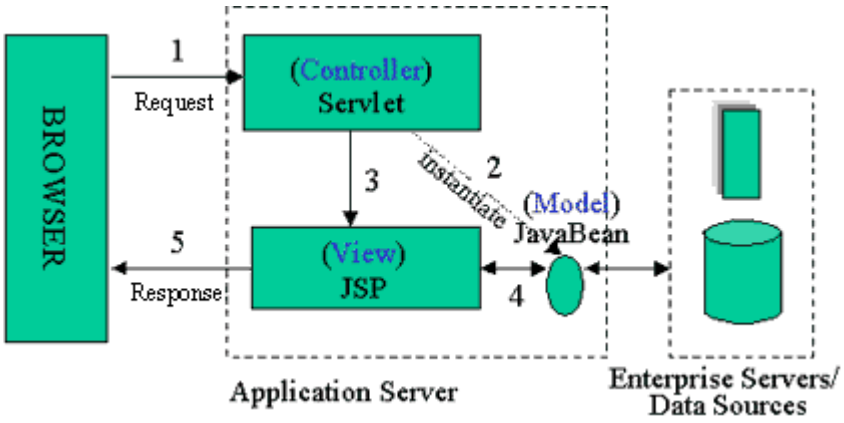

Figure 2.1: Model-view-controller architectural pattern [Mod]

1.2 Web service

A Web service [Wik08] is defined by the W3C as "*a software system designed to support interoperable Machine to Machine interaction over a network*". It can be also viewed as a "black box" that can be used without concerning any of the implementation details.

Using Web services, applications can publish their functions or messages to the rest of the world. Thus, applications can be converted into Web-applications. It also allows for communication between different applications installed in different systems to be possible. For example, an accounting application on the Windows Server can exchange data with IT supplier's UNIX server.

Web services use XML to code and decode data, and SOAP to transport the data using open protocols.

Web Services Description Language (WSDL) offers what is called a machine-readable description of the operations offered by the service. This is not a requirement of a SOAP endpoint but, it is a prerequisite for automated client-server-side code generation in many Java and .NET SOAP frameworks.

2.2.1 XML

The Extensible Markup Language (XML) [XML08] is a general-purpose specification for creating custom markup languages, normally used to encode documents and to serialize data. It is an extensible language because it allows its users to define their own elements and its main purpose is to facilitate the sharing of structured data across different information systems, via the Internet.

XML Schema

There are two well known XML Schema languages: the DTDs and XSDs. The first one is the oldest schema format for XML and is used in many applications because it is considered the easiest to read and write. But this schema is not considered ideal because it doesn’t support namespaces and uses custom non-XML syntax. The XSDs are the successor of DTDs and are far more powerful. They use a rich datatyping system, allowing for more detailed constraints on an XML document's logical structure. They also use an XML-based format, which makes it possible to use ordinary XML tools to help process them.

2.2.2 SOAP

Simple Object Access Protocol (SOAP) [SOA07] is a communication protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. There are many technical advantages of using SOAP over HTTP since it allows easier communication through proxies and firewalls. It is platform and language independent, simple and extensible.

2.2.3 HTTP

The Hypertext Transfer Protocol (HTTP) [Wik081] is the most used communication protocol for the transfer of information on intranets and the World Wide Web. HTTP is a request/response standard between a client and a server, where the client is the end-user and the server is the web site.

2.2.4 WSDL

Web Services Description Language (WSDL) [WSD01] is an XML format document used to describe Web services. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema.

1.3 Apache Log4j

Logging is an important component in the development cycle. It offers several advantages, such as providing precise context about the execution of an application. Once inserted into the code, the generation of logging output is automatic. Moreover, log output can be made persistent so it can be studied later. In addition to its use in the development cycle, a sufficiently rich logging package can also be viewed as an auditing tool.

Apache log4j [Apa07] is an open-source package release under the Apache Software License and it consists in a Java-based logging utility, which is used primarily as a debugging tool.

The latest release version of Apache log4j has six logging levels from highest to lowest:

- FATAL
- ERROR
- WARN
- INFO
- DEBUG
- TRACE

There are two ways to configure Log4j. One uses a property file and the other an XML file. Within each you can define three main components: Loggers, Appenders and Layouts.

Configuring logging via a file has the advantage of turning logging on or off without modifying the application that uses log4j. For example, the application can be allowed to run with logging off until there's a problem, and then logging can be turned back on simply by modifying the configuration file and understanding the application.

Loggers are logical log file names. They are the names that are known to the Java application. Each logger is independently configurable as to what level of logging (FATAL, ERROR, etc) it currently logs. In early versions of log4j, these were called category and priority, but now they're called logger and level, respectively.

The actual outputs are done by *Appenders*. There are numerous *Appenders* available with descriptive names such as *FileAppender*, *ConsoleAppender*, *SocketAppender*, *SyslogAppender*, *NTEventLogAppender* and even *SMTPAppender*. Multiple *Appenders* can be attached to any Logger. For example, it's possible to log the same information to a file locally and to a socket listener on another computer.

1.4 Oracle

CERN and Oracle [Ora] have a partnership that dates back more than 20 years. It was in 1982 that version 2.3 of the Oracle Database was first installed at CERN. This partnership appears from the communal interest between the two organizations. CERN want to put emerging Oracle technologies to test against its extreme data processing needs, and Oracle continues to innovate as a result of pushing its technology to the limits.

2.4.1 Oracle JDeveloper

Oracle JDeveloper is an integrated development environment (IDE) for building applications using the Java, XML, Web services and SQL standards. It supports the complete development life cycle with integrated features for modeling, coding, debugging, testing, profiling, tuning, and deploying applications.

2.4.2 Oracle Designer

Oracle Designer [Ora1] is an integrated Computer-Aided Software Engineering (CASE) tool that covers the full development life cycle, from the design to the maintenance.

Its Key Features are:

- Centralised Repository - allows designers/developers to work simultaneously on the same application with all of the work being immediately available to others;
- Requirement and Process - Displays requirements and processes in easy-to-understand diagrams, matrices, and logical and physical data models.
- Graphical User Interface (GUI) - generates a GUI using Oracle Forms and Reports.
- Time Saving/Efficiency - if a part of the code (e.g. function or cursor) is being used in more than one package, then that code only has to be modified in one place;
- Maintenance - All source codes are easily located and modularized so it is easy to change a module and re-generate the changes back into the application;
- Documentation - allows a number of useful reports to be generated from the repository.

Most of the CRA functionalities have been developed using Oracle Designer.

2.4.3 Oracle Application Server

An Oracle Application Server [Ora2] is a collection of services that support deployment of applications and Web services over the Internet. The heart of the Oracle Application Server consists of the Oracle HTTP Server and J2EE containers which allow the deployment J2EE-based applications.

The main services are:

- Presentation Services - deliver dynamic content to client browsers through the support of servlets, Java Server Pages, Perl scripts, PL/SQL Pages, Oracle Forms and Reports, and Discoverer Viewer;
- Communication Services - handle all incoming requests by providing Web listening services;
- Business Logic and Intelligence Services - support the deployment and sharing of business information;
- System Services - provide tools to manage the Oracle environment and network security.

2.4.4 Oracle Database

The Oracle Database is a relational database management system (RDBMS) where data is stored in the form of tables together with the relationships among the data.

PL/SQL

Procedural Language/Structured Query Language [PLS] is an imperative third-generation language that has been designed specifically for the seamless processing of SQL commands. It provides specific syntax for this purpose and supports exactly the same datatypes as SQL. Server-side PL/SQL is stored and compiled as packages, procedures and functions in Oracle Database and runs within the Oracle executable.

Requirements and Architecture

This section describes the requirements identified for the implementation of the generic synchronization server and the Web services. Also described in this chapter is the functional, logical and physical architecture of the server.

3.1 Requirements

The requirements are a difficult and a decisive part of the project and need collaboration with other department teams. Therefore, in order to collect all the requirements a preliminary meeting between the CRA Team took place. Further meetings where scheduled and also took place with various IT sections (owners of the different CERN computing services). The implementation of a replacement of the polling mechanism to inform the CERN computing services of the changes made in CRA was and is an important goal of the project. This goal required three main implementations, a system to keep track of the modification on the CRA Objects, a generic server to do the polling of the changes and the implementation of Web services to communicate with the different services. Besides these main implementations it was also necessary implement a failure and alert mechanism.

The system to keep track of the modification on the CRA objects requires:

- Creation of a table to keep track of modifications. This table only keeps the essential information (object ID and last modification date) necessary for synchronizations, so it can't be used for historical or auditing reasons.
- Creation of triggers on the Oracle tables where the objects are stored. The triggers fire when an operation of insert, update or delete is executed in one of the tables of the object. When it fires a record is added or updated in the respective modification table.

The implementation of the generic polling server had the following requirements:

- The server would be composed by three main parts:
    - The abstract server - responsible for reading the configuration file;
    - The agent manager – that has the objective of creating and monitoring the agents (one per service that needs to be notified);
    - The abstract agent - which does the polling mechanism and call the web service;
- For each CRA object, a server would be instantiated. For example, for e-group objects, the server will be composed of the e-groups server, the agent manager and the e-groups agent;
- The notification of the successful synchronization should add or update a record (with the object ID, service ID and the notification date) on the notification table;
- The polling mechanism compares the modification table and the notification table of the object:
    - Only objects with modification dates greater than notification dates should be considered;

    - The synchronization of the modifications should be processed from the oldest to the newest modification date;

- The server should allow the configuration of:
    - which CERN computing services would be used in the synchronization process;
    - the URLs of those services;
    - how often the service agents should wait between synchronizations;
    - the maximum interval of failure;
    - the list of e-mails of administrators.
- It is also required to stop the server when necessary without ending it in an inconsistent state. An inconsistent state happens if for example, an e-group was successfully synchronized but a record on the notification table wasn't committed.

The different CERN computing services are responsible for the implementation of the Web services, of which CRA is the client:

- Creation of the XML Schema Definition for the CRA objects (e.g. e-groups);
- Creation of the WSDLs importing the XSDs of the CRA objects for automated client-server-side code generation;
- Creation of Web services for synchronization of CRA objects;

For uploading e-groups, a Web service has to be deployed on the CRA machine of which the services are the clients.

Beyond these requirements, the server should support a failure and alert mechanism defined in the configuration file:

- If the synchronization fails, the Agent should keep trying with a delay calculated using the configuration parameters and keep the number of failed attempts in the notification table;
- To alert the administrators of problems with the Web services, an e-mail is sent to them when the maximum failure interval is reached.

3.2 Architecture

The following sections describe the functional, logical and physical architecture of the polling server.

3.2.1 Functional Architecture

The server can have as many instances as CRA objects. The “behavior” of the server can differ based on CRA objects and configuration files.

If the server is started, the following sequence of steps will occur. Firstly, an instance of the CRA Monitor Server class will be created. This class will read the configuration file with parameters like: services to which the server should synchronize, the e-mails of the people in charge in case of maximum failed interval is reached, etc. Then a socket will be opened so the server can receive a command to be stopped at any time. After these last steps are completed, an instance of the agent manager is created, passing all the CERN computing services as a parameter. Then the agent manager creates as many agents as computer services the server has to synchronize with. The polling mechanism is done on the agent thread that compares the modification and notification tables to know which object is the next one to be synchronized. All the agents run the same code but call different Web services depending on the computer service. In order to control which CRA objects are successfully synchronized the notification table is updated with the object ID, the computer service ID and the system date. If there are no more requests the agent will sleep for a few seconds and then wake up again to see if there are new requests to process. If the synchronization fails the Agent should keep trying with a delay calculated using the configuration parameters. By default it will be (2^failedAttempts) in seconds until the maximum failed interval (30 minutes) is reached and then an error report will be sent by mail to the server’s administrators.

When a “stop server” command is sent, the CRA Monitor Server class informs the Agent Manager class, which warns all created agents to stop. If the agent is in the middle of a synchronization process then agent will only stop when the last request is completed. Then the Agent manager cleans all the agents and connections to the database, and finally informs the server that it’s safe to stop.

3.2.2 Logical Architecture

The polling server has 3 main parts:

- The server responsible for reading the configuration file;
- The Agent Manager that has the objective of monitoring the Agents per Service (e.g. AIS, AFS, etc.);
- The Agent which does the polling mechanism based on the modifications and notifications tables and calls the Web service.

The following picture describes the logical scheme of the Server:

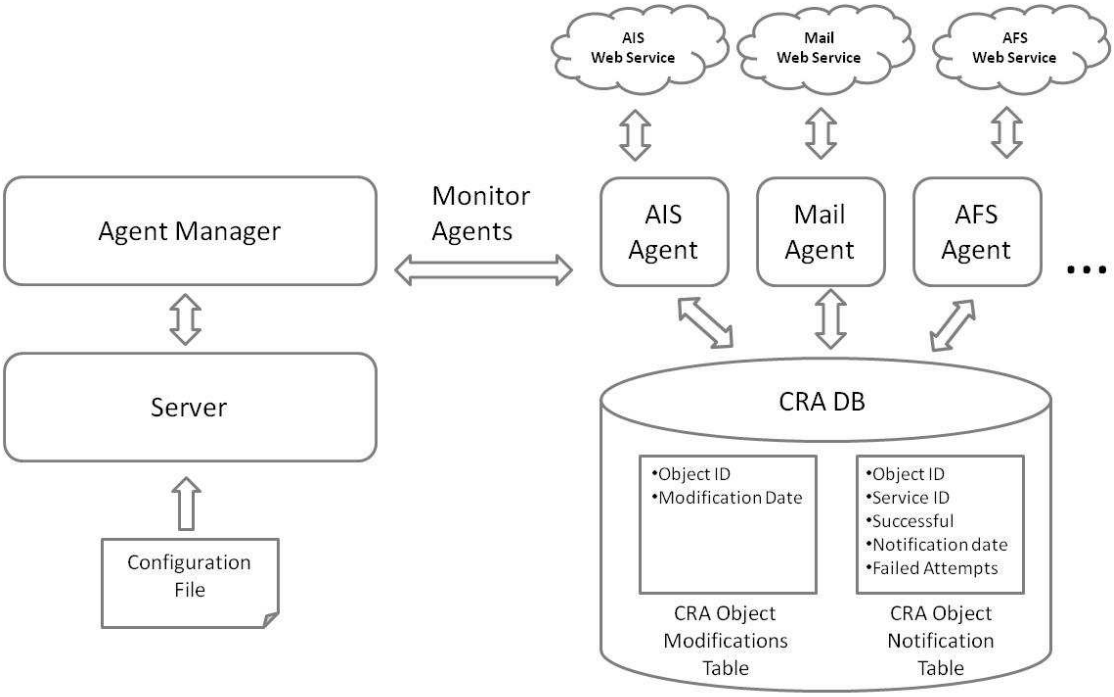

Figure 3.1: Logical scheme of the Server

3.2.3 Physical Architecture

The physical architecture is described in the following picture:

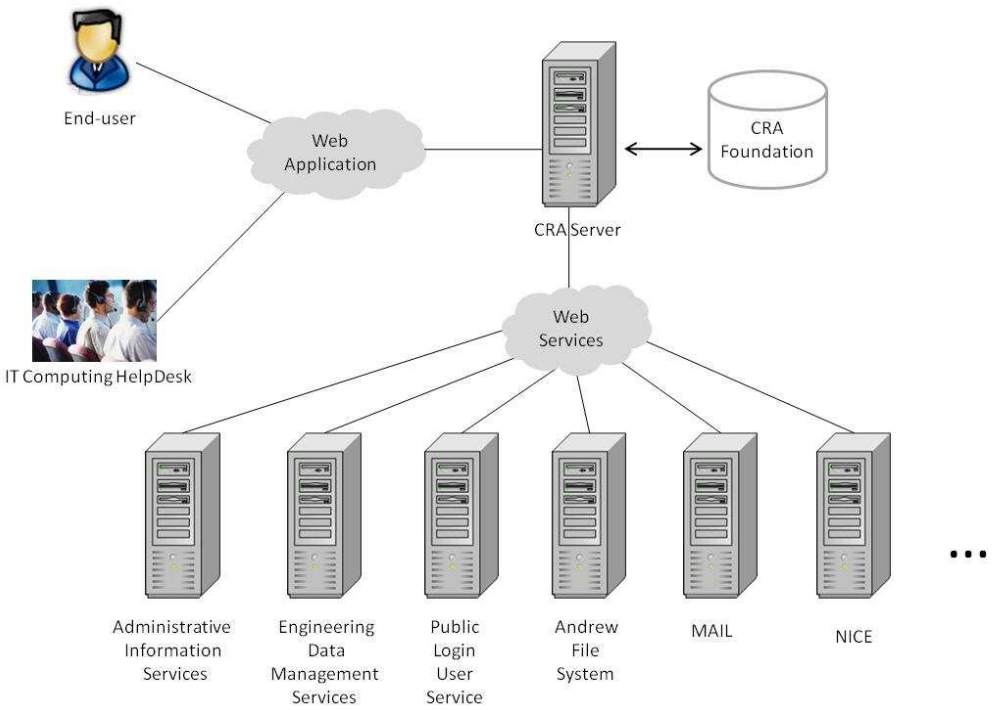

Figure 3.2: Physical scheme of the Server

- The end-user represents the CERN users. Users can have different privileges levels depending on their needs.
- The "IT Computing HelpDesk" is a first level "hot-line" for problems with computing services which is provided by the IT Department.
- The Helpdesk is the central entry point for user queries. It aims to resolve most queries (e.g. "password reset" or "quota increase") immediately (or within a very short time). If that is not possible, it will pass the queries on to specialized support staff (people in IT) whom have the expertise and additional privileges. For all such cases, users will receive an e-mail with a ticket number for further reference and to track the problem resolution.
- The CRA Foundation database represents the CRA and Foundation schemas used by the CRA server.
- The CRA Server provides a Web application to the end-user and helpdesk. The communication between the CRA server and all the services (e.g. Administrative Information Service or Engineering Data Management Services) is made using Web services.

Server Development

This chapter will explain the implementation of the server in detail. Firstly it will describe the CRA objects and then the server class diagrams. Furthermore, details of the server configuration and CERN computing services and the Web services will provide an insight into the development of the server. Also mentioned are the implementation details, the setup of the server environment and the deployment process.

4.1 Common decisions and implementation details

It had been decided that the first objective is to keep running Simba Lists and E-groups in parallel, but replace the database process that synchronizes once a day by the synchronization server. Thus, e-mail list created, updated or deleted in e-groups are immediately reflected on the mail service.

Then in a near future, the Simba List Web application available in the mail service will be replaced by the e-groups available in the CRA Web application. The reason why e-groups can replace Simba list is because e-groups cover all features of Simba list and provide more functionalities to the end-user. These functionalities will be explained in detail later in this chapter.

During the implementation phase, weekly CRA team meetings and monthly CRA Collaboration meetings (with other IT sections) were conducted to better monitor the course of events.

4.2 CRA Objects

There are many different CRA objects, but in this project only the synchronization of e-groups and CERN External Mail Accounts were considered. The reason to choose these two objects is the need of replacing the Simba list application.

4.2.1 E-groups

The e-groups application allows the definition of static lists of members (managed by the owner or administrator through the Web interface), and dynamic lists of members (for example the list of all staff employees of a particular department).

Members can be a person (any CERN User), a service provider (e.g. CRA Support), a static or dynamic e-group (e.g. staff members of the IT-AIS group), an external e-mail (e.g. tiago.batista@gmail.com), an account (e.g. user NICE account) or even a CERN Unit (e.g. IT Department).

A static e-group defines the list of members by searching them in the HR database, by other e-groups, or even importing the list of members by a file. The use of the HR database to find members is very powerful, because there isn’t a need to know exactly the e-mail address. The member can be searched by name, phone, fax, Unit (department/group), building, login,

CERN ID, etc For example, if you want to build a list of all persons belonging to the IT department then you just type IT department in the CERN Unit field. Finally if you want the e-mail list to be synchronized with the members of the CERN Unit you can use the automatic synchronization.

A dynamic e-group is defined by selection criteria. Criteria can be defined by a huge list of predefined fields like age, "at CERN" (Y/N), building, department, experiment (e.g. "Atlas"), nationality, sex (M/F), status ("FELL", "STAF"), etc. For example, in order to define a dynamic list of all staff member of the IT-AIS group with age between 30 and 40 the following criteria is needed:

Field	Operator	Value 1	Value 2
Department	Equals	IT	
Group	Equals	AIS	
Status	Equals	STAF	
Age	Between	30	40

Table 4.1: Dynamic e-group criterions

After explaining the e-groups application, it is clear to see that it can be used for many purposes. The first idea to replace the Simba lists was because e-groups provide all the functionalities that Simba lists already have and much more. Simba lists are provided by the mail services that allow static email lists to be created by typing mail by mail or importing a file. However, with e-groups one can create static lists using HR database to search for members and, more importantly it allows the user to create dynamic lists using selection criteria. So if somebody leaves the section, retires, leaves CERN, joins the section, etc, the list does not have to be updated as it's automatically synchronized with the HR database.

But that is not the only purpose of e-groups. This application can be useful for numerous other cases. It can be very useful for other IT groups and departments that are responsible for many computing services. For example, the Indico[3] service could use e-groups to set the list of participants in a conference or meeting. The SVN[4] service would use it to know which accounts can access a certain project. The Linux service would use it to set the list of administrators for a machine.

[3] The Indico tool allows you to manage complex conferences, workshops, and time-tables of meetings.

[4] Subversion is an open source version control system used to host CERN projects.

Advantages of having e-groups for e-mail lists

It’s strongly recommend that every CERN 'official' mailing list be registered as one e-group. Here are some of the advantages of having an e-group, compared to a private list in some user's address book:

	List in address book	E-groups
Availability	When the list owner is away, the list is unavailable.	The list is always available.
Data quality	It's the responsibility of the list owner to keep the list up to date.	The list can be updated automatically using the CERN central database (either to follow users' preferred e-mail addresses or to re-create the list following some selection criteria).
Data consistency	If two people want to share a list (e.g. to overcome the availability problem), consistency can be compromised.	The list is always stored in one single place.
List management	By hand	Web interface

Table 4.2: Advantages of having e-groups for e-mail lists

Advantages of having e-groups for Indico

Indico could use e-groups for internal management. For example, if a meeting is required to be scheduled for a particular department, only those staff members belonging to that department have the authority to change the settings. Indico could use a dynamic e-group to know all the possible participants and use another dynamic e-group to know who is allowed to change the settings for that particular meeting.

Advantages of having e-groups for SVN

Subversion is an open source version control system used to host CERN projects. SVN could use e-groups for internal management too. Each CERN project hosted on SVN would set different privileges to different groups of accounts. So, with e-groups, one can define an IT Department dynamic e-group with all accounts that can browse and define a team project dynamic e-group with the privilege to read and write on the project.

E-groups Customization

New specific additional features to e-groups can be required from CERN computing services. For example, the mail service required a feature that would allow changing the e-mail properties of any e-group. The solution for additional features comes with the development of the specific feature by the computer service in question and then a new tab is added to the e-groups application, containing an HTML IFrame to the feature. Layout and style have to fit the e-groups application and, it is important to say that those specific features are the entire responsibility of the services. For the mail server service, the CRA Web application adds a tab with an HTML IFrame containing the feature, passing as a parameter the e-group name in question.

The link between e-groups and features has to be transparent for the end-user. It wouldn't be pleasant for the user to login again to use a specific feature. So, to solve this and similar problems, a common project called "single sign on" has been developed and applied to all computing services at CERN. Thus, for the users, the e-groups application and specific features appear to be the same application even if some features belong to other computer services.

Database structure

For the purpose of synchronization, the database structure underwent some changes. Triggers on *cra_egroups*, and *cra_egroup_selfsubscriptions* will call the *cra_egoups_package.egroup_modified* procedure which will set the last modification date of the e-group in *cra_egroups_modifications*. If a membership changes, this procedure will be called from the *cra_egroups_package* and will recursively add all e-groups which include the modified membership. The server that informs the services of the changes can record the outcome of the SOAP call in *cra_egroup_notifications*. Using these two tables, the server can determine which changes need to be communicated to which services. If an e-group no longer exists in the *cra_egoups* table, it means that a *deleteEgroup* operation must be invoked, otherwise the server must invoke the *synchronizeEgroup* operation.

The following figure represents the e-groups database structure:

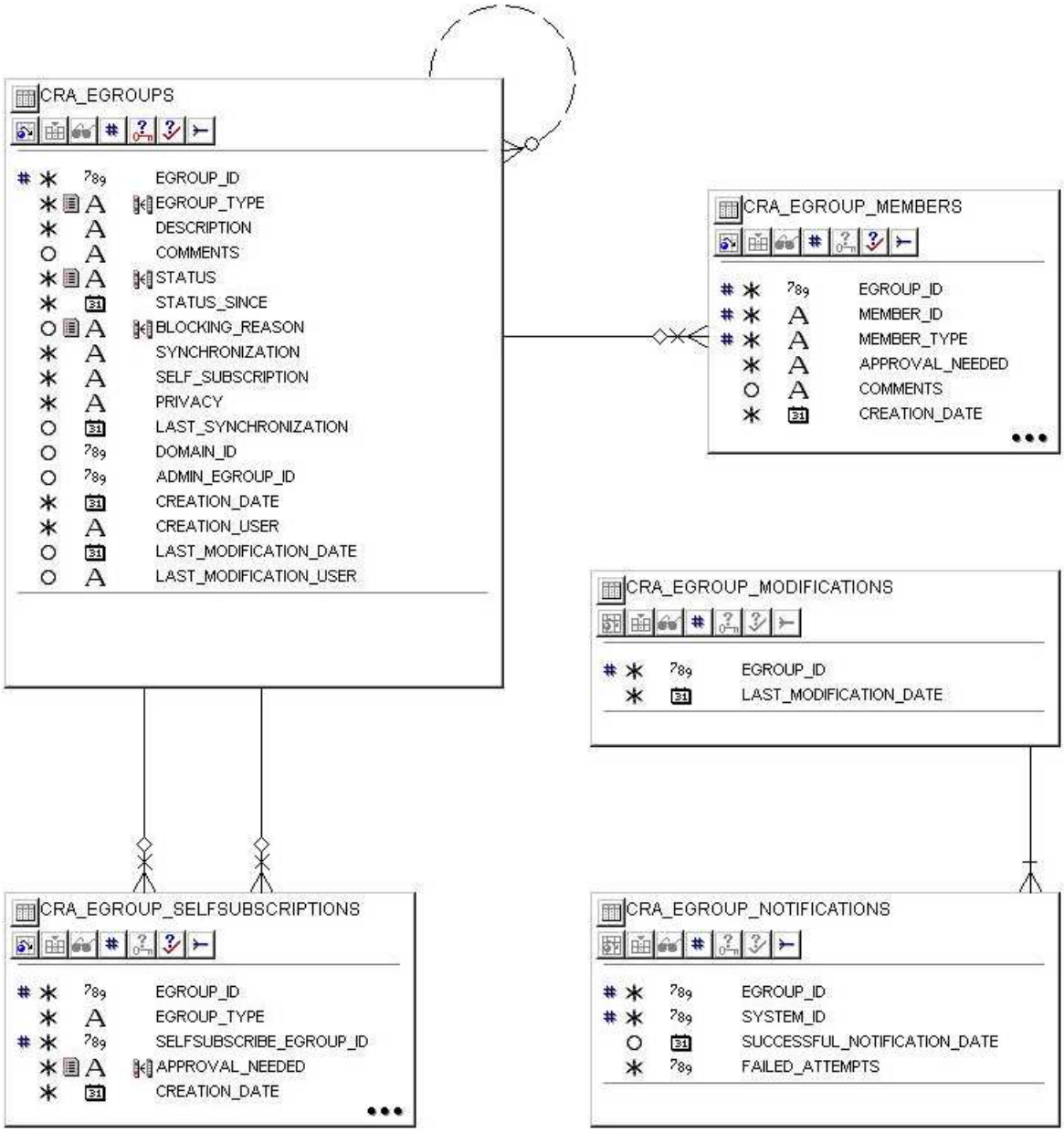

Figure 4.1: CRA e-groups database structure

4.2.2 CERN External E-mail Accounts

First it is important to say that CERN External e-mail account are managed by the mail service, and the reason why this object is synchronized with CRA is because it can be used in any e-group as an external member.

CERN External Accounts allow external users who have no regular CERN account to register online simple credentials (e-mail and password), then they can use them to access CERN applications. Such External accounts only grant limited access to some CERN applications (and web sites).

The database structure needed a few changes:

- a system parameter was added to the CRA database to control the sequence number of the last external account change;
- a new method was added to the *cra_egroups_package* to update the e-mail of the external members in e-groups.

4.3 Server Class Diagram

The following class diagram shows the generic server classes, their interrelationships (including inheritance and aggregation), and the operations they perform:

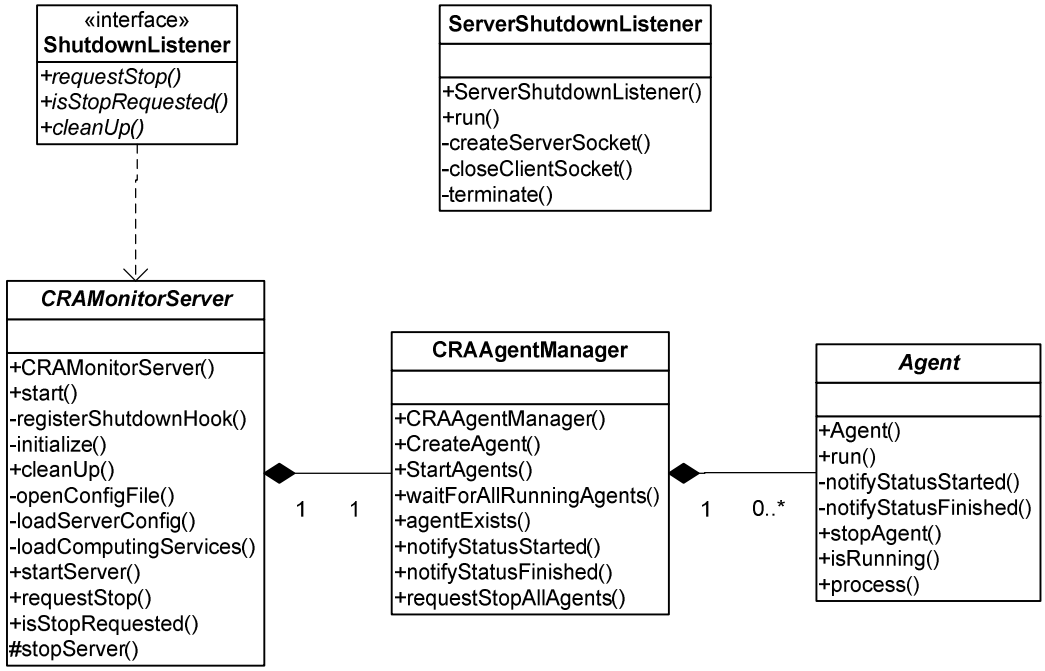

Figure 4.2: Server class diagram

The generic server is composed by the following classes:

- *CRAMonitorServer* – Abstract class that reads the configuration file, loads the computer services and opens a socket to receive the "shutdown" command;
- *CRAAgentManager* – This class creates one agent per computer service, starts and requests to stop the agents;

- *Agent* – Abstract class that extends the *Thread* class for polling, failure handling and the alert mechanisms.
- *ShutdownListener* – Interface class to stop the server and clean up connections to the database;
- *ServerShutdownListener* – This class opens a socket to wait for the "shutdown" command;

If the server is used for e-groups, it has the following class diagram:

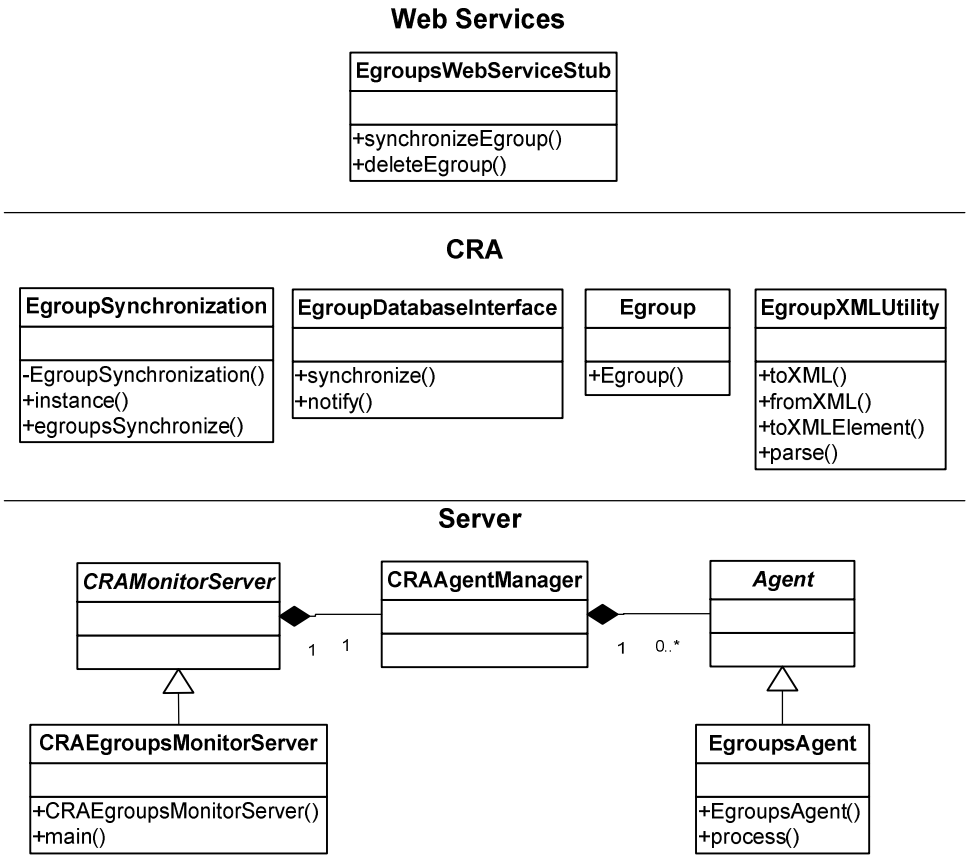

Figure 4.3: E-groups server class diagram

The server part is composed by the following additional classes:

- *CRAEgroupsMonitorServer* – This class extends *CRAMonitorServer* class and specifies the name of the configuration file and keywords to be used;
- *EgroupsAgent* – This class extends the *Agent* class and calls the egroupsSynchronize method in the *EgroupsSynchronization* class;

The CRA part shows the most important classes for the synchronization of e-groups:

- *EgroupsSynchronization* – This class calls the synchronize method in the *EgroupDatabaseInterface* class to know which e-group will be the next to be processed. Then, if the e-group no longer exists in the *cra_egoups* table, it means that a *deleteEgroup* method must be invoked, otherwise the server must invoke the *synchronizeEgroup* method. Both operations are defined on the Web service Client;

- *EgroupDatabaseInterface* – This class communicates with the database using the JDBC driver. It has the *synchronization* method to get the next e-group to process, and the *notify* method to register successful/unsuccessful synchronizations;
- *Egroup* – Defines the e-group object;
- *EgroupXMLUtility* – This class permits to convert an e-group object to an XML document and the reverse.

There is one Web services client defined to synchronize e-groups:

- *EgroupsWebServicesStub* – Represents the Web service client and contains the *synchronizeEgroup* and *deleteEgroup* methods. These two methods synchronize and delete e-groups on the computing services (e.g. mail service);

If the sever is used for External email accounts, it has the following class diagram:

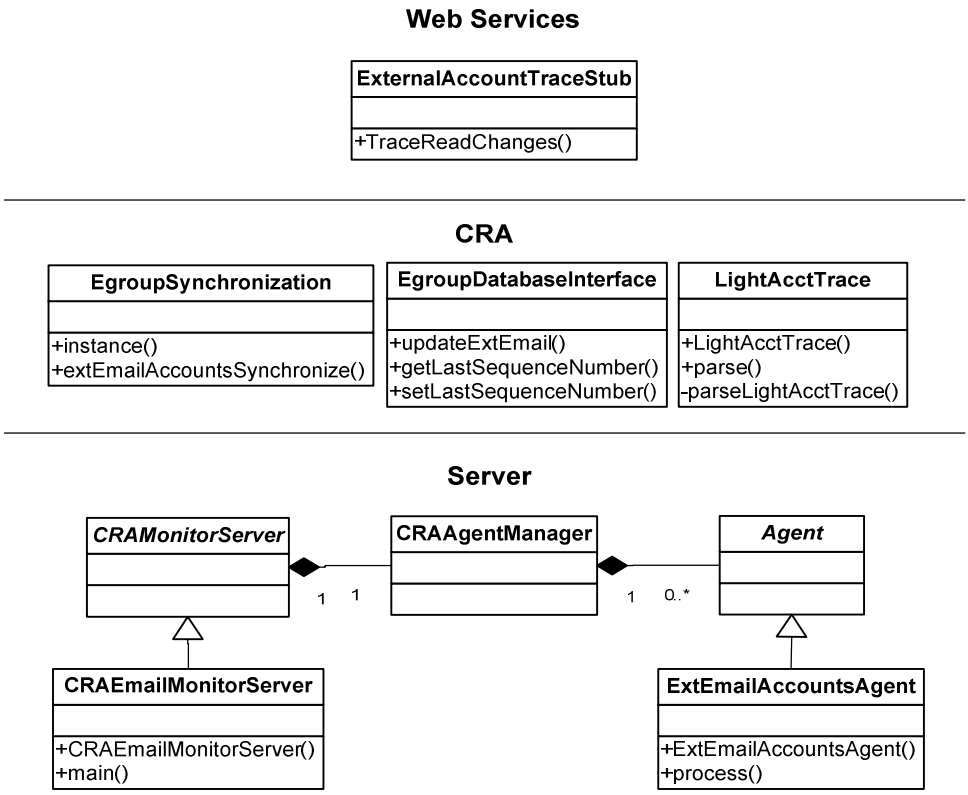

Figure 4.4: External accounts server class diagram

The CERN External Email Accounts can be referenced as Light Accounts and represent the external members in e-groups.
The server part is composed by the following additional classes:

- *CRAEmailMonitorServer* – This class extends *CRAMonitorServer* class and specifies the name of the configuration file and keywords to be used;
- *ExtEmailAccountsAgent* – This class extends the *Agent* class and calls the extEmailAccountsSynchronize method in the *EgroupsSynchronization* class.

The CRA part shows the most important classes for the synchronization of external accounts:

- *EgroupsSynchronization* – This class calls the *TraceReadChanges* method from the ExternalAccountTraceStub class to receive the changes made in the CERN External

Email Accounts. Then the *updateExtEmail* method in the class *EgroupDatabaseInterface* is called to update the e-mail of the external members in e-groups;

- *EgroupDatabaseInterface* – It has the *updateExtEmail* method to change the external members in the e-groups. The *getLastSequenceNumber* and *setLastSequenceNumber* methods allow to keep track of the last modification in the CERN External Accounts;
- *LightAccountTrace* – Defines the external account or light account object and allows to converts from an external account object to an XML document and the reverse.

There is one Web services defined to trace the changes made in external accounts:

- *ExternalAccountTraceStub* – Represents the Web service client and contain the *TraceReadChanges* method. This method permits to keep track of the changes made in external accounts in the mail computing service;

4.4 Server Configuration

In the CRA Server the configuration means all the necessary variables and files needed to start and set the "behavior" of the server have for the moment been split into several files:

- CRA.properties (mandatory);
- (AgentType)ServerConfig.properties (e.g. EgroupsServerConfig.properties) (mandatory);
- log4j_server.properties.

The CRA properties file contains a set of properties such as:

- Connect string for the database;
- Path to the SQL query templates;
- URL of the Computing Rules;
- Parameters for automatic expiration;
- Nice users accounts allowed to call the CRA Web services;
- Web service URLs.

There is a server configuration properties file per CRA object:

(CRA object)ServerConfig.properties file contains a set of properties such as:

- The interval between subsequent synchronizations, in seconds;
- The maximum interval, in seconds, of subsequent retries;
- Which services will be used;
- The URLs for the services that are used;
- The list of e-mails of administrators.

There is only one log4j properties file for all the CRA object instances of the server. Nevertheless, different logging files are created per instance of the server. Due to this the CRA object name is passed to the log4j properties file as a system parameter by the CRAMonitorServer class. For example, if one starts two instances of the server, one for e-groups and another for external e-mail accounts, egroup_server.log and email_server.log logging files will be created.

The log4j_server.properties contains all the configuration properties used in the runtime logging.

4.5 CERN computing services

In order to understand better which computing services [CER07] the CRA server synchronizes with, various services are described below:

AIS

The Administrative Information Services.

MAIL

This service allows the access to mail services when the user has a local mailbox at CERN. The user is also able to setup the desired mail address.

NICE (Windows PCs)

This service allows users to connect to the Windows PC network around CERN and gives access to the DFS file system on Windows.

AFS/PLUS (Linux PCs)

AFS (the Andrew file system) is the file base used at CERN for the central Linux system. The PLUS service (Public Login User Service) is the interactive logon service to Linux for all CERN users.

PARC

This is a specialized account, for usage of engineering applications.

EDMS

The Engineering and Equipment Data Management Service provides CERN with an integrated and reliable Engineering and Equipment Data Information Systems.

REMEDY

REMEDY provides a problem and call tracking service, used by the HelpDesk.

4.6 Web Services

In context of this project and for communication purposes, it has been necessary to create three Web services, two for e-groups and another one for CERN External Accounts.

4.6.1 E-groups

For the e-groups Web service, it is necessary to first define the XML Schema of the e-group object and then the WSDL.

E-group XML Schema Definition

The use of XSD [XSD] brings many advantages like:

- XSD Schema supports Inheritance: one schema can inherit from another schema. This is a useful feature because it provides the opportunity for reuse;
- XSD schema provides the ability to specify data types;
- XSD Schema is an XML document, so there is no real need to learn any new syntax, unlike DTDs.

The reason why the e-group XML Schema definition has been externalized from the WSDL is because of the ability to reuse certain parts for the future account synchronization, e.g. the definition of the resource owner.

The XML Schema Definition that describes the owner of an e-group is given below:

```
- <xs:complexType name="UserType">
  - <xs:sequence>
      <xs:element name="CCID" type="xs:long" minOccurs="1" maxOccurs="1" />
      <xs:element name="Type" type="cra:UserTypeCode" minOccurs="1" maxOccurs="1" />
      <xs:element name="Name" type="xs:string" minOccurs="1" maxOccurs="1" />
      <xs:element name="ComputingRulesSigned" type="xs:boolean" minOccurs="1" maxOccurs="1" />
      <xs:element name="Pem" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="PrimaryGem" type="cra:GemType" minOccurs="0" maxOccurs="1" />
      <xs:element name="SecondaryGems" type="cra:SecondaryGemsType" minOccurs="0" maxOccurs="1" />
      <xs:element name="CernUnit" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="CernDepartment" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="CernGroup" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Telephone1" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Fax" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Building" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Floor" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Room" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="Mailbox" type="xs:string" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
```

An example of an XML document that conforms to this schema is given below:

```
- <Owner>
    <CCID>438064</CCID>
    <Type>Person</Type>
    <Name>Tiago Rodrigues Vieira Batista</Name>
    <ComputingRulesSigned>true</ComputingRulesSigned>
    <Pem>batista@mail.cern.ch</Pem>
  - <PrimaryGem>
      <EmailAddress>tiago.batista@cern.ch</EmailAddress>
      <DisplayInPhonebook>true</DisplayInPhonebook>
    </PrimaryGem>
  - <SecondaryGems>
    - <SecondaryGem>
        <EmailAddress>tiago.rodrigues.vieira.batista@cern.ch</EmailAddress>
        <DisplayInPhonebook>false</DisplayInPhonebook>
      </SecondaryGem>
    </SecondaryGems>
    <CernUnit>IT-AIS-FPF</CernUnit>
    <CernDepartment>IT</CernDepartment>
    <CernGroup>AIS</CernGroup>
    <Telephone1>7xxxx</Telephone1>
    <Fax>6xxxx</Fax>
    <Building>513</Building>
    <Floor>2</Floor>
    <Room>012</Room>
    <Mailbox>G01235</Mailbox>
  </Owner>
```

E-group WSDL

The creation of the WSDL document is useful for the automated client-server-side code generation in Java and .NET SOAP frameworks. In this project both types of client-server-side code generation are used: for CRA, the Java JDeveloper framework, and for the mail server service, the .NET framework.

For e-groups, it has been necessary to create two types of Web services, one to synchronize with all the CERN computing services and another one to upload e-groups. Until now, the Web service to synchronize e-groups has been deployed on the mail service machine and the upload Web service has been deployed on the CRA machine.

The upload Web service can be used as an automatic way of creating e-groups instead of creating them one by one on the CRA Web Interface. For instance the SVN service has already expressed the intention to use the upload Web service to automatically create e-groups.

Migration of Simba lists to E-groups

Simba list is an application that will be replaced by e-groups. The upload Web service will allow uploading of existing Simba lists, and converting them to e-groups.

4.6.2 CERN External Accounts

For CERN External Accounts the WSDL has already been created along with the Web service which has been deployed on the mail server service machine. The client-side code generation was used for CRA through the JDeveloper framework.

The objective of the Web service is to keep track of all operations that users do on their external accounts (like update information, change e-mail, delete account, etc.).

Each operation has a sequence number. Calling the CERN External Account Web Service with the last processed sequence number will return a batch of the recorded operations that follow in the sequence. This information is processed to update the CRA external members e-mail in the database.

4.6.3 Web services Security

The developed Web services followed some security measures:

- The Web services obliges the computer services to authenticate with a NICE account;
- The list of allowed NICE accounts have to be specified;

4.7 Setup of the server environment

The setup of the server environment is composed by the following files:

- Binary folder
  - bin/**setenv.sh** – contains the environment variables of Java (JAVA_HOME), and Oracle (JDBC_JAR_PATH), etc.
  - bin/**server.sh** – set all the variables of the server
  - bin/**egroups-server-startup.sh –** starts the server to synchronize E-groups objects
  - bin/**egroups-server-shutdown.sh –** stops the server
  - bin/**email-server-startup.sh –** starts the server to synchronize CERN external e-mail accounts objects
  - bin/**email-server-shutdown.sh –** stops the server
  -
- Configuration folder
  - conf/**CRA.properties**
  - conf/**EgroupsServerConfig.properties**
  - conf/**EmailServerConfig.properties**
  - conf/**log4j_server.properties**
- Library folder
  - lib/**cra.jar –** Contains all the classes belonging to CRA
  - lib/**cra-server.jar –** Contains all the classes belonging to the Server
  - Other libraries necessaries to run Web Services
- Log folder
  - log/**egroups_server.log** – log file of the server to synchronize e-groups
  - log/**email_server.log -** log file of the server to synchronize external e-mail accounts

4.8 Deployment process

There are three types of running environments the development environment, the test and the production environment:

- In the development environment, each developer has a workstation with Oracle JDeveloper10g, a standalone OC4J, a CVS client and a connection to the test database. The developer creates and deploys components and runs tests without getting in the way of other developers or testers.
- In the test environment all the components are deployed to the Oracle Application Server and a connection to the Oracle test database is used. System tests are run in this environment.
- In the production environment all the tested components are deployed to the Oracle Application Server and a connection to the Oracle production database is used.

In the deployment process all application files are packaged as an archive file and transferred to the server where the application will run. CRA is delivered in “enterprise archive” format (“cra.ear”) and in the polling Server application as a compressed Java file (“server.jar”). The developed components have to follow the deployment cycle, from development to test environment and then from test to production environment.

Code quality and Tests

This chapter will discuss the quality of the written code, and the code inspections. A description and evaluation of the tests performed on the server will follow.

5.1 Code Inspections

A code inspection is carried out once every month to ensure that the developed code meets the requirements. It mainly checks for errors and also that the code respects the coding standards. Not only are these code inspections important for the quality of the written code but also, for the testing phase and maintenance: any one that participated can maintain the code afterwards.

5.1.1 Coding standards

Some facts:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.

In order to improve readability/understandability/changeability hence the maintainability of a piece of software the Coding standards have to address the following issues:

- Programming **style** or structure relates to the way in which program constructs are layed-out and grouped together. By properly aligning the different constructs and by grouping the different pieces of code ensures it is easier for the reader to see which pieces of code belong together thus, enhancing the understandability and changeability of the code. Two programs written in the same language but, with a completely different style are like two programs written in different languages. A person different from the author will have to familiarize him/herself with the style in order to understand the author's intent. A good program structure will also favor the printability of a piece of software.
- **Naming:** Naming program objects in a structured way will help a newcomer in understanding the purpose and usage of the objects as well as their scope. The latter plays a role when assessing the impact of a change made to the piece of software.
- **Modification:** Standards can help in the prevention of introducing errors when a piece of code needs to be inserted, updated, or removed.

In addition, the standards set a framework for newcomers that explain in some detail what is expected from them. Non-compliance to the standards can result in refusal of a produced piece of software. Adherence to the standards can be validated during an inspection session.

5.2 Performed Tests

First individual tests to the polling server were executed testing the triggers for modification in e-groups, the polling of the changes, failure handling and alert mechanisms. Then individual calls to the Web services were carried out to test the communication between CRA and the computing services. Finally the server was instantiated for e-groups and external e-mail accounts so that CERN computing services can be informed of the changes made in CRA.

5.2.1 Database Triggers

To test if the database triggers were well defined, all the possible changes to the e-group objects were performed, confirming that a new record was added or updated correctly on the modification table.

5.2.2 Server

In order to test the server a class *MonitorServerTest* had been created with the objective of polling the changes made in a dummy object. The goal of this test is to test the server in all possible different scenarios before it can be tested for real. In this phase different object and configuration options were used, major problems in the implementation could be found.

The failure handling and alert mechanisms have proved to be a good solution to solve problems related with the unavailability of any Web service and in case of unexpected errors by addressing the problem to the person in charge of the instance of the polling server.

5.2.3 E-groups

For e-groups two classes were implemented: *EgroupXMLUtilityTester* and *EgroupsTester*. The first has the objective of parsing the java object to an XML document and the second for testing the synchronization, upload and delete of e-groups through Web services.

Following the unit tests the server could be instantiated for e-groups in the test environment. This phase firstly required the upload of all the e-groups to the mail service then the synchronization tests could start by changing things in the CRA e-groups Web application and by checking if they were replicated on the mail service.

5.2.4 CERN External Accounts

In order to test the synchronization of external e-mail account changes through the Web service, the class *ExtEmailAccountTester* was implemented.

The server was instantiated for external account objects and it was verified that all the changes made by the external users were replicated on the CRA external members.

5.2.5 CERN Computer Services

From the point of view of the CRA it is important that computer services follow correctly the synchronization process. This means that when CRA calls the computer service Web service to synchronize or delete e-groups this operation is correctly processed but, if the computer service returns false or an error occurs, the state of the computer service remains the same, so problems like trying to delete the same object twice doesn't happen. For example if the deleted operation of an e-group was called upon and the mail service deleted the e-group from the system but, somehow an error occurred, then the deletion of the e-group has to be rolled back onto the computer service.

5.2.6 Unit Tests

For unit test proposes, a test database with 820 e-groups and 144215 e-group members, and the production database with 794 e-groups and 146583 e-group members to synchronize with the mail service were used. The objective of the mail service was to create email lists from the e-groups so, only members with a CERN account and with CERN external accounts were considered in the mail service and the others had been discarded.

Conclusions and future perspectives

6.1 Role in the project

My role in this project was to replace the old polling mechanism between CRA and the computer services inherited from the Computer Center Data Base by a generic synchronization server through Web services. This involved the study the old mechanism, the CRA and the computer services. The design and architecture of the server and the system to keep track of the modifications in the CRA objects was already planned. I implemented the synchronization server, the necessary Web services, the server configuration, the notification mechanism for successful/unsuccessful synchronizations and the failure and alert mechanisms.

My role also consisted of a plan and the co-operation in the replacement of Simba lists for e-groups. This implied:

- studying e-groups and Simba lists interfaces
- using the synchronization server developed for e-groups and CERN external accounts.
- adding a link to the e-mail properties in the e-groups interface.
- creating a client Web service to synchronize e-groups from CRA to the mail service.
- creating a client Web service to synchronize CERN external accounts from the mail service to CRA.
- developing a Web service to upload e-groups from the computer services to CRA.

6.2 Project developed

As previously mentioned on section 4.2.1, Simba lists and e-groups run in parallel and the synchronization between them is complimented by a process that pushes all updated e-groups from the CRA database into files to the mail service. This takes place during the night. Once the data is transported to the mail service a process converts e-groups to Simba lists and puts them in the production area. One of the objectives of this project is the migration from Simba lists to e-groups, another objective is the substitution of the existing synchronization process.

The new synchronization process developed in this project was successfully implemented and tested for CRA and the mail service. It has proved to be a good solution to synchronize e-groups in a reliable and quick way. For the time being Simba lists will not be removed as a preventive precaution measure. Thus, both systems will run in parallel synchronized by the polling server through the Web services, so that when changes are made in e-groups they will be reflected in Simba lists in real time. Therefore, in the future Simba lists will be uploaded and converted to e-group objects and only one entry point will exist for the creation of e-mail lists.

This project has proved to be very useful for CRA and for the mail service by providing a solution to create static and dynamic e-mail lists in real time. This project fully complied with the specifications proposed in this report and at the same time it successfully passed the tests

which were submitted. This clearly confirmed that the strategic options which were taken contributed to the achievement of objectives.

6.3 Future Perspectives

As aforementioned on section 4.2.1, e-groups can be a useful tool for other computer services like Indico, SVN and Linux, meetings can be scheduled with the responsible IT sections, explaining the advantages of using e-groups. After the agreement between CRA and the responsible IT sections, the synchronization process will be similar to the one used for Simba lists. The trigger mechanism, the polling server, the e-group XSD and the WSDL will be the same. The main implementation will be on the computer service side. They should implement a Web Service based on the WSDL, permitting the synchronization with the polling server. After this implementation they can create, upload and delete e-groups by using the uploading e-groups Web service or the e-groups Web application and have all e-groups synchronized on their systems.

This project only addressed two kinds of objects; the e-groups and CERN external accounts, but other CRA objects can also be synchronized. So, the next step can be the synchronization of the NICE, PLUS and AFS accounts and passwords.

Currently the existing synchronization between CRA and NICE service is made by a polling mechanism inherited from the Computer Centre Database. Therefore, when a CERN User asks to reset the NICE password the new password will take on average 10 minutes before it can be used in the system. Even if the help desk warns the user of that fact and he/she keeps trying to access with the new password then the NICE account will end up being blocked. The new polling server will solve this problem for any account (e.g. NICE, PLUS, AFS) by informing the service right after the Helpdesk resets any account.

Concerning new CRA objects - first all the requirements will be collected by the IT section responsible for the computing services. Then the configuration files and the necessary additional classes to the server and Web services will be implemented. However, adding new CRA objects will be much easier because the polling mechanism is already done.

List of acronyms and SI units

AIS	Administrative Information Services
AFS	Andrew file system
API	Application Programming Interface
CASE	Computer-Aided Software Engineering
CRA	Computing Resources Administration
CVS	Concurrent Versions System
DTD	Document Type Definition
EDMS	Engineering Data Management Service
EGEE	Enabling Grids for E-science
EJB	Enterprise JavaBeans
FPF	Finance, Procurement & Foundation section
HR	Human Resources
HTML	Hyper-Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IDE	Integrated Development Environment
IT	Information Technology
J2EE	Java 2 Platform, Enterprise Edition
JDBC	Java Database Connectivity
JSF	JavaServer Faces
JSTL	JavaServer Pages Standard Tag Library
LHC	Large Hadron Collider
LEP	Large Electron-Positron collider
MVC	Model-View-Controller
OC4J	Oracle Containers for Java
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
RDBMS	Relational Database Management System
SOAP	Simple Object Access Protocol

SPS	Super Proton Synchrotron
SQL	Structured Query Language
PLUS	Public Login User Service
PL/SQL	Procedural Language/Structured Query Language
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Style sheet Language Transformations
B	The byte is a unit that measures of information storage, most often consisting of eight bits.
k	Kilo 10^3
M	Mega 10^6
P	Peta 10^{15}
m	The metre is the basic unit of length. It is the distance light travels, in a vacuum, in 1/299792458th of a second.

References

[Abo07] , 2007, Jul.. About CERN. Available at http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/AboutCERN-en.html

[Apa07] , 2007, Aug.. Apache log4j. Available at http://logging.apache.org/log4j/1.2/index.html

[Apa08] , 2008, Jun.. Apache Struts, 2008 . Available at http://struts.apache.org/

[Bru02] B. Eckel, *Thinking in Java (3rd Edition)*. 2002.

[CER04] , 2004, Feb.. CERN IT Department. Available at http://it-div.web.cern.ch/it-div/what-we-do/

[Cer05] , Nov. 2005. Cern Computer Newsletter. Available at http://cerncourier.com/cws/article/cnl/23547

[CER07] , 2007, Sep.. CERN Computing Services. Available at http://it-div.web.cern.ch/it-div/AccountsandpasswordsatCERN.htm

[CER08] , 2008, Jul.. CERN Document Server. Available at http://cdsweb.cern.ch/

[CER081] , 2008, Mar.. CERN IT-AIS Group. Available at http://ais.web.cern.ch/ais/mandate.html

[Chu] B. K. Chuck Musciano, *HTML: The Definitive Guide, Third Edition*.

[Chu02] C. Cavaness, *Programming Jakarta Struts*. 2002.

[CRA08] , 2008, Jan.. CRA Home Page. Available at http://cra.web.cern.ch/cra/

[CRA081] , 2008, Mar.. CRA Project page. Available at http://ais.web.cern.ch/ais/projs/cra/welcome.html

[Dan04] D. Goodman, *JavaScript Bible*. 2004.

[Gul02] Gulcu, *The complete log4j Manual*. 2002.

[Hus02] H. &. G. F. &. C. D. &. D. Winterfeldt, *Struts in Action*. 2002.

[Jas01] J. H. &. W. Crawford, *Java Servlet Programming, Second Edition*. 2001.

[Jen03] J. Vesperman, *Essential CVS*. 2003.

[Mod] , 1999, Dec.. Model-view-controller architectural pattern. Available at http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html

[Mod08] , 2008, Jul.. Model View Controller. Available at http://struts.apache.org/primer.html

[Ora] Oracle. Available at http://www.oracle.com/global/eu/rd/fs/presentation-cern.html

[Ora1] Oracle Designer. Available at http://www.theconnectpartnership.co.uk/downloads/white_papers/case_tool_benefits.pdf

[Ora2] Oracle Application Server. Available at http://its.unm.edu/oracle/iAS/

[PLS] PL/SQL. Available at http://www.oracle.com/technology/tech/pl_sql/index.html

[SOA07] , 2007, Apr.. SOAP. Available at http://www.w3.org/TR/soap/

[Wik08] Wikipedia. , 2008. Web services. Available at http://en.wikipedia.org/w/index.php?title=Web_service&oldid=224106059

[Wik081] Wikipedia. , 2008. HTML. Available at http://en.wikipedia.org/w/index.php?title=HTML&oldid=224978186

[WSD01] , 2001, Mar.. WSDL. Available at http://www.w3.org/TR/wsdl

[XML08] , 2008, May. XML. Available at http://www.w3.org/XML/

[XSD] XSD. Available at http://www.15seconds.com/issue/031209.htm