Software Design Document

for

Club Management System

Prepared by

Nur Fatin Nabilah binti Mohd Shofe

Mohamad Danial Effendi

Nur Fakhira binti Rosdi

Muhammad Haris

LeBron Familia

10 December 2018

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this software design document is to describe the architecture and system design of the Club Management System (CMS). The CMS is designed to create a much simpler software for the management of a club inside any learning institutes.

1.2. Scope

This document describes the implementation details of the Club Management System software. The software will consist of two major functions. First to ease the registration of new clubs as well as their instructors and members, the second is to ease the instructors in managing the clubs including the activities they will do in the future. This document will not specify testing of the software.

1.3. Audience

The intended audience for this software is the students, teachers, and lecturers from all different kinds of learning institutes. This software is to make the instructors of the club job become easier in handling the club.

1.4. Definitions and Acronyms

Terms	Definition
CMS	Club Management System
UPM	University Putra Malaysia
Instructors	The instructor for the clubs in the system
Members	The members of the clubs in the system
Mb	Megabytes

1.5. References

1. IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.
2. Software Requirement Specification for Club Management System, May 25 2018
3. Software Design Document of Event Driven DIS PDU Logger (EDDIS system), October 28, 2002
4. 433-340 Software Engineering Project Manual, 2002
5. Schach, Stephen R, \Classical And Object-Oriented Software Engineering with UML and Java. 4th Edition", 1999, WCB/McGraw-Hill
6. Software Design Document for a specific implementation of ‘BCI2000’, July 2004

2. System Overview

CMS project provides and manages various club activities such as member registration, registration for various regular and vacation batches and more. The CMS software is a net-built system that manages the entire club activities and provides respective functionality for various types of visitors (for the students specifically). This system is built with respect managing all the clubs offered in Universiti Putra Malaysia (UPM). It allows normal users to avail for club membership, book the ground at for desired days and even enroll for various activities in the club that they found interesting. The CMS is built keeping in mind various daily activities of all club and the software automates all these club functionalities for easy operation for the students. There are actually a numerous amount of club and organizations offered in UPM, but the students wouldn't enroll because they weren't that happy signing up using the manual way, which is the paper-based operations.

Currently, the process of managing the club is file based and manual. These obsolete management systems slow down functionality of the club. For example, a new user wants to enroll in a training batch he or she must visit the club and fill up the registration form. The form then passes through a hierarchy of club members before approval. It takes time as well as effort form a user's perspective. This is just a single case. Same problem persists in all the major operation of the club. To solve this problem, we come up with a system name CMS. It is a web-based application. This system uses Java Server Faces as a programming language. This system interacts with user through the webpage. The scope of the project is to design an interface for student to choose clubs and activities that they want to join. Besides, this project also provide student to register the club they want without system down. This system will use Persistence API (JPA) to support the database.

3. Design Consideration

3.1. Assumptions and Dependencies

Assumption

- Client or members of CMS has an active internet connection or has access to one to view the website
- Clients or members runs an operating system which support internet browsing.
- Clients or members runs latest browsing software to access our website.

Dependencies

1. User need the connection of internet at minimum of 500Kb per second.
2. User need to run an operating system that support internet browsing.
3. Our system only relies on the student Id which is matric no. and password to log in, there is no alternative way to log in into our system.

3.2. General Constraints

1) Hardware or software environment

There is a web server requirement. Windows 2003 is the preferred operating system due to the security requirements. ASP .Net 2.0 Framework needs to be installed.

2) End-user environment

Administrator, instructor, and members should have network and internet connectivity. They will need to login with their user credentials to be able to use the system.

3) Standards compliance

None

4) Interoperability requirements

None

5) Interface/protocol requirements

Network connectivity and TPC/IP support are required.

6) Data repository and distribution requirements

Data will be stored in the database and Web services will be used to store the encrypted data. Stored procs will be used in some functions. By doing all the connections via stored procs can limit the access to the databases to the stored proc level.

7) Memory and other capacity limitations

5MB/10MB HDD space is required.

3.3. Goals and Guidelines

Development Environment

The application development environment must remain consistent. This minimizes negative impacts to interoperability and quality.

Ease of Use

The new features must be easy to use and provide a strong user experience. New features cannot impact existing functionality from a user perspective.

Extensibility

The proposed features must be extensible. Features can be enabled as needed or required by the users.

Organized

The system is organized to make it easier for the user to register into new clubs as the administrator will be the one to approve the user registration, so the instructor can focus more on organizing activities.

3.4. Development Methods

The basics of a good architecture is to layer the application into multiple autocratic and autonomous applications that can be replaced individually and allow us to keep the application running while we are working on a specific layer.

Availability

The architecture should support a high availability environment. Infrastructure redundancy is required. This ensures the solution is available if multiple servers or an entire data center fail. The current availability of the solution per the hosting providers service level agreement is 99.999% availability.

Security

Solution architecture should expose only the minimal amount of code possible. Most of the back-end pieces should be hidden away. In addition to that, security of each system should be multi-layered.

Extensibility

Architecture must be able to swap out modules, change layers, and add pieces to the application without having to worry about the underlying data contracts in place.

4. System Analysis

4.1. Product Function

4.1.1. Register

After the user browses our system, the user must register to the system to use it. There will be a register button in the login page which will be redirected to the user registration page. User needs to fill all the required fields in the interface. After the user registered to the system, the user can proceed to login to use the system.

4.1.2. Login

The user that has registered in the system can log in to the system using their username and password.

4.1.3. Update activity

An instructor can update the activities for the club including the name of activities, date, location, and detail of the activity. After adding the and update the activity, the activity will be listed in the activity column.

4.1.4. Update profile

Both instructor and member can update their profile based on the field provided in the update profile interface.

4.1.5. Access payment record

The members of the club need to make annual payments when they join a certain club. The instructor can view the payment made by the members to follow-up which members have already made the payment or still not made the payment.

4.1.6. Generate report

Instructor generates a report for the club based on the name, date, attendance for the club and details about the activity that has been conducted.

4.1.7. View activity

Members of the club can view the activity that has been added by the instructor.

4.2. ER-Diagram

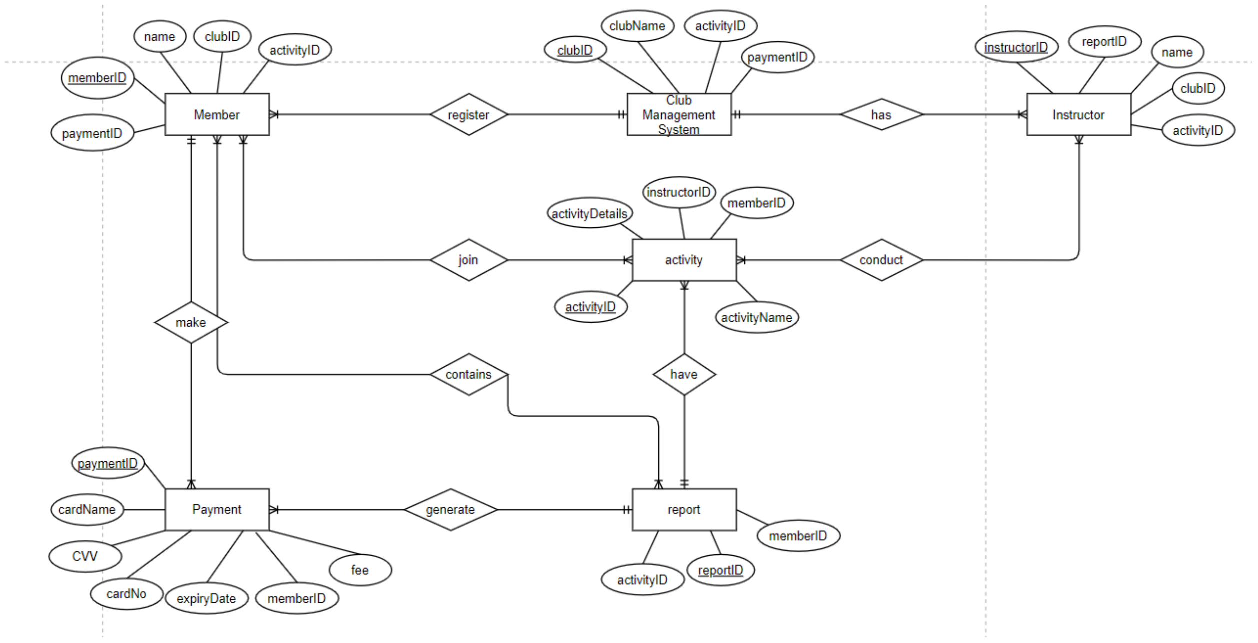

Figure 4.2: Entity-Relationship diagram for Club Management System

5. System Detail Design

5.1 Deployment Diagram

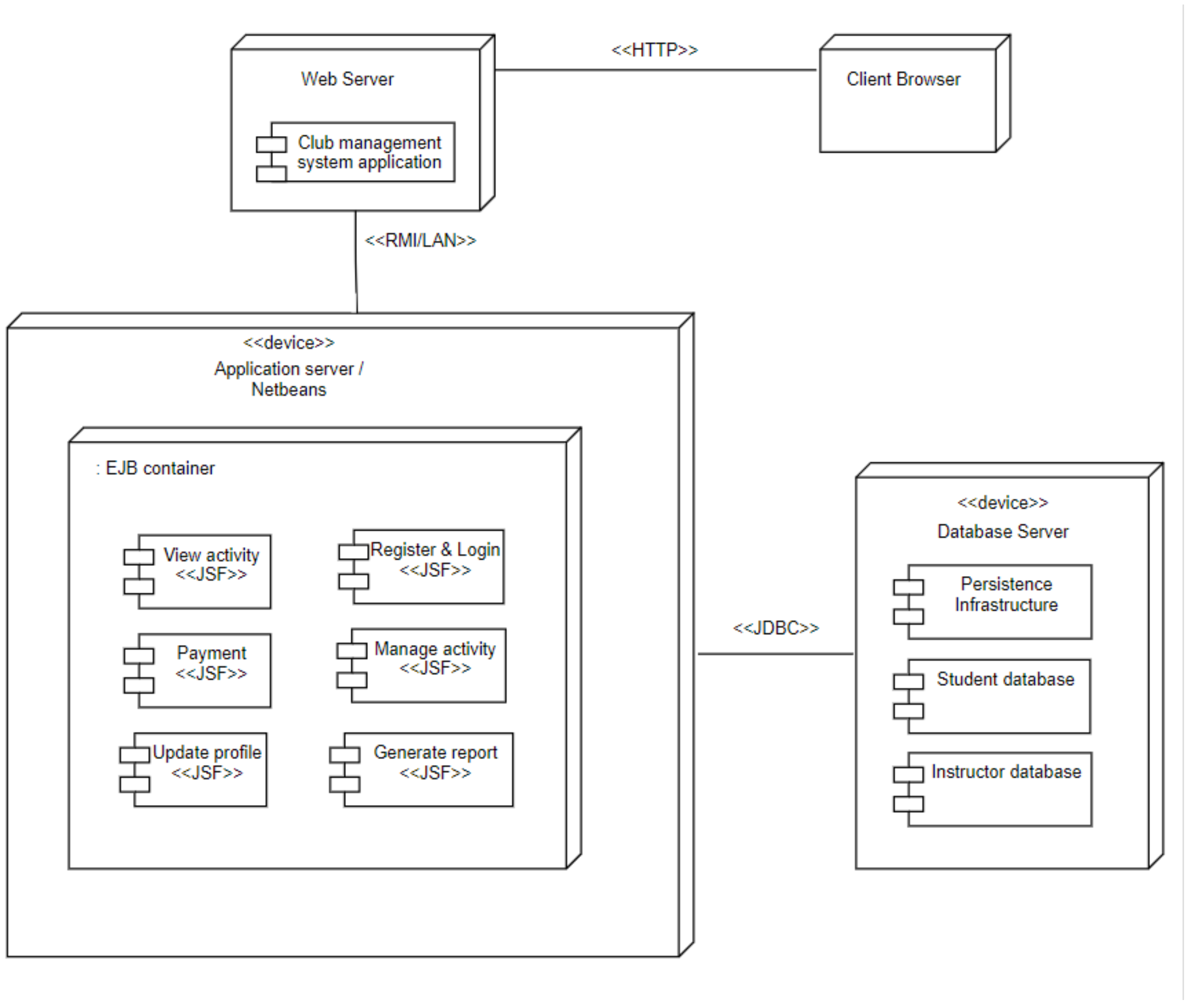

Figure 5.1: Deployment diagram for Club Management System

The deployment diagram above shows the structure of the run-time system. it will capture the hardware that will be used to implement the system and links between different items of hardware. It will also show the model physical elements and the communication paths between them. it will be used to plan the system architecture. For client browsers it will go through the HTTP to open the web server. Then, from the web server it will go to the application server through RMI. In the application server it will have an EJB container that consists of the JSF page component. Lastly, from the application server it will go to the database server through JDBC path. For the database we used a database in netbeans with JPA (Java Persistence API) as the medium between application server and database server.

5.2 Package Diagram

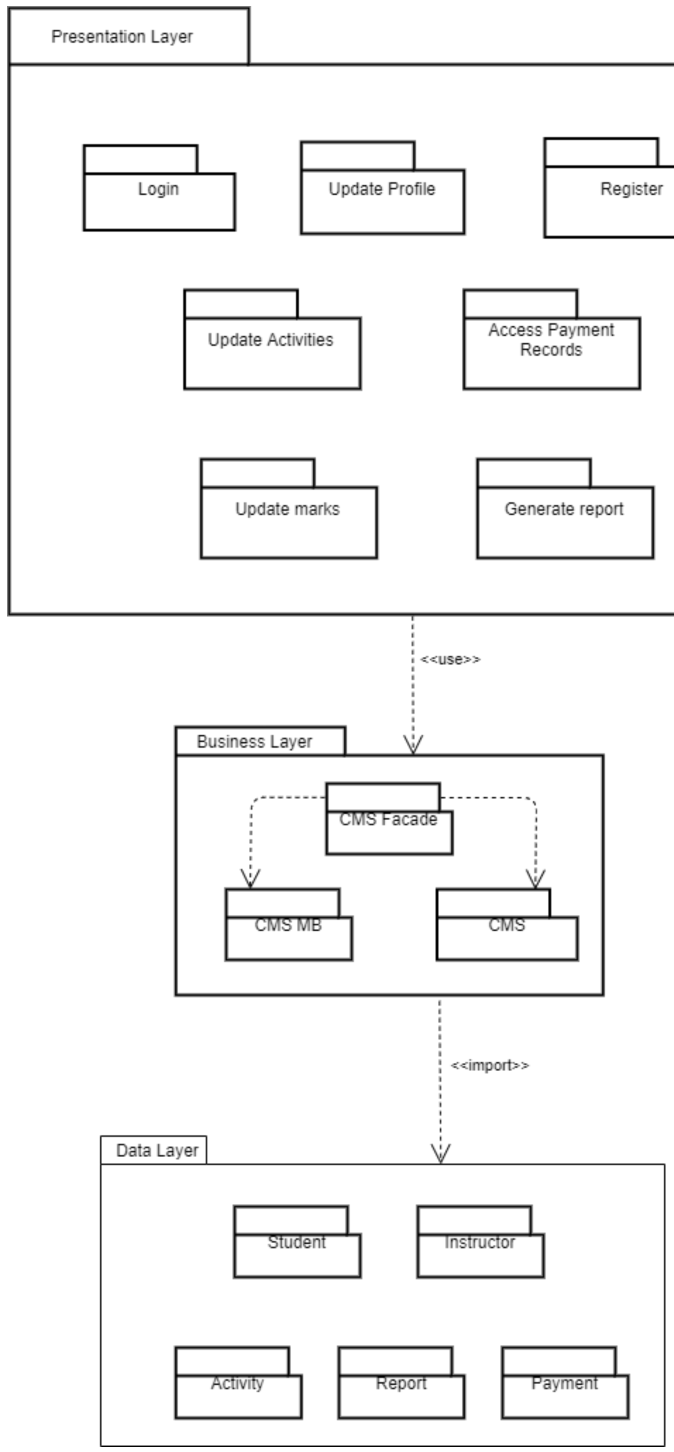

Figure 5.2: Package diagram of Club Management System

Figure 5.2 shows a package diagram for the Club Management System which consists of three layers. In the presentation layer users can view and use the functionality of the system such as sign up to the system, update their profile, access payment records and other. All the controllers of the system are being packaged into a business layer. The controllers are CMS Facade, CMS MB and CMS. Data layer is to store all the data that is input from the user either as a student or instructor.

5.3 Component Diagram

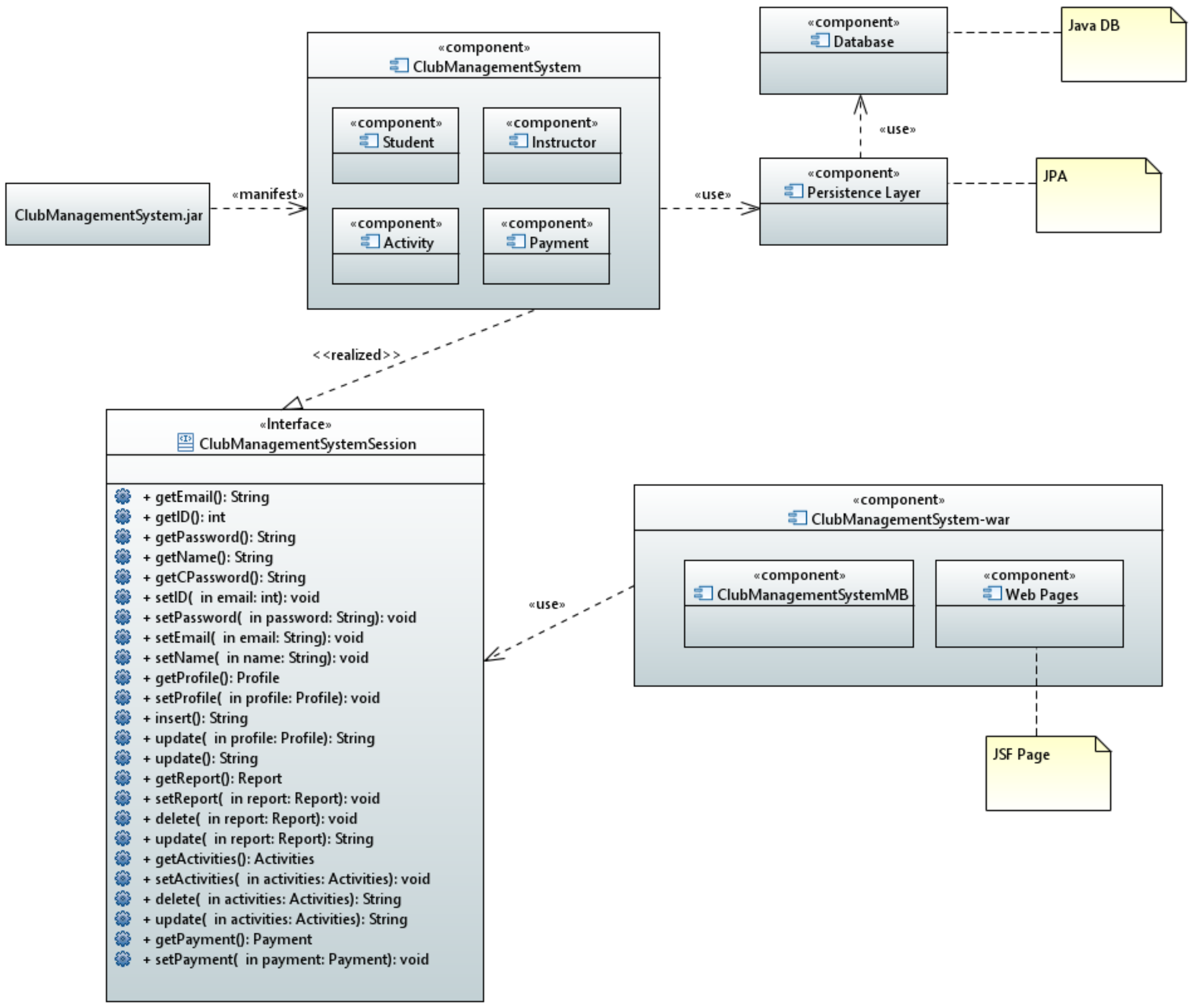
«component»
Database
Java DB
«component»
ClubManagementSystem
«component»
Student
«component»
Instructor
«component»
Activity
«component»
Payment
«use»
«component»
Persistence Layer
JPA
ClubManagementSystem.jar
«manifest»
«use»
<<realized>>
«Interface»
ClubManagementSystemSession
+ getEmail(): String
+ getID(): int
+ getPassword(): String
+ getName(): String
+ getCPassword(): String
+ setID(in email: int): void
+ setPassword(in password: String): void
+ setEmail(in email: String): void
+ setName(in name: String): void
+ getProfile(): Profile
+ setProfile(in profile: Profile): void
+ insert(): String
+ update(in profile: Profile): String
+ update(): String
+ getReport(): Report
+ setReport(in report: Report): void
+ delete(in report: Report): void
+ update(in report: Report): String
+ getActivities(): Activities
+ setActivities(in activities: Activities): void
+ delete(in activities: Activities): String
+ update(in activities: Activities): String
+ getPayment(): Payment
+ setPayment(in payment: Payment): void
«use»
«component»
ClubManagementSystem-war
«component»
ClubManagementSystemMB
«component»
Web Pages
JSF Page

Figure 5.3: Component diagram for Club Management System

Figure 5.3 shows the component diagram for the model layer of the Club Management System. The diagram shows the different components, such as Student, Instructor, Payment, and Activity in the model layer and how the controller layer component which is ClubManagementSystemSession interacts with these components. The diagram illustrates a database component, Java DB which interacts with the model layer with the support of the persistence layer, JPA. Additionally, the view layer component will be displayed using JSF web page.

6. Description of API and third-party component

6.1 Java Persistence API (JPA) and Third Party-Components

Java Persistence API (JPA) or now known as Jakarta Persistence is concerned with persistence, which loosely means any mechanism by which Java objects outlive the application process that created them. It is a Java application programming interface specification that describes the management of relational data in applications using Java Platform, Standard Edition and Java Platform, Enterprise Edition. The JPA is one possible approach to Object-Relational Mapping (ORM). Via JPA the developer can map, store, update and retrieve data from relational databases to Java objects and vice versa. For our system, the JPA is used together with the databases that we have created inside the Netbeans IDE.

The third party-component that we use inside our system is JCalendar. JCalendar is a Java date chooser bean for graphically picking a date. JCalendar is composed of several other Java beans, a JDayChooser, a JMonthChooser and a JYearChooser. For the JCalendar, we implement it in the Student Payment page, where the student needs to enter the date of the activity that they need to pay for and the expiry date for their credit/debit card. It is also implemented in the Instructor Activity page when the instructor wants to add or update an activity record. By using this third-party component, we can make it easier for the users to enter the date information as they only need to choose a date from the calendar given to them. So, there is no risk of users entering the wrong information such as letters and special characters in the field.

7. Glossary

ASP.net – an open-source server-side web application framework designed for web development to produce dynamic web pages

UC – *Use Case* a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language as an *actor*) and a system to achieve a goal.

UPM – *Universiti Putra Malaysia*

TCP – *Transmission Control Protocol* a standard that defines how to establish and maintain a network conversation via which application programs can exchange data.

IP – *Internet Protocol* a set of rules governing the format of data sent over the Internet or other network.

HDD – *Hard Disk Drive* mechanism that controls the positioning, reading and writing of the hard disk, which furnishes data storage.

IEEE – *Institute of Electrical and Electronic Engineering* Professional organization whose activities include the development of communications and network standards.

UML – *Unified Modeling Language is a* standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system.

8. Bibliography

1. IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

2. Software Requirement Specification for Club Management System, May 25 2018

3. Software Design Document of Event Driven DIS PDU Logger (EDDIS system), October 28, 2002

4. 433-340 Software Engineering Project Manual, 2002

5. Schach, Stephen R, \Classical And Object-Oriented Software Engineering with UML and Java. 4th Edition", 1999, WCB/McGraw-Hill

6. Software Design Document for a specific implementation of ‘BCI2000’, July 2004