

Software Engineering Group 1 (2008-2009)

Software Design Description

Lieven Govaerts

20 November 2008

Group members (alphabetically):

	How to reach us
Software Engineering Group 1	se1@tinf.vub.ac.be
Mathias Becquaert	Mathias.Becquaert@vub.ac.be
Jeffrey Geysens	Jeffrey.Geysens@vub.ac.be
Steve De Ridder	Steve.De.Ridder@vub.ac.be
Maarten Deville	Maarten.Deville@vub.ac.be
Robrecht Dewaele	Robrecht.Dewaele@vub.ac.be
Lieven Govaerts	Lieven.Govaerts@vub.ac.be
Stijn Neirinckx	Stijn.Neirinckx@vub.ac.be
Ken Tanaka	Ken.Tanaka@vub.ac.be
Gilles Vanderveken	Gilles.Vanderveken@vub.ac.be

Abstract

This document, the Software Design Description lays down the architecture and detailed design of the Calendar software, to be conceived in the context of the annual project for the *Software Engineering* course. It took place under guidance of Prof Dirk Vermeir and assistant Damien Trog during the 2008-2009 study year at the Vrije Universiteit Brussel (Brussels, Belgium).

Document history:

version	date	author	description
v 0.1	20 Nov 2008	Lieven Govaerts	Draft initialization

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	3
1.3	Reference materials	3
1.4	Definitions and acronyms (listed alphabetically)	3
2	System Architecture	4
2.1	Overview	4
2.2	Three-Tier Architecture	4
2.3	Discussion of Alternative Designs	4
2.4	Survey of Technologies Used	4
2.4.1	Presentation Layer	4
2.4.2	Business Layer	5
2.4.3	Data Layer	5
2.5	Physical Configuration	5
2.6	System Interface Description	6
3	Component Design	7
3.1	Main Servlet	7
3.2	Notification Service	7
4	Software Features	7
4.1	Subscribe	7
4.2	Login	8
4.3	Create Event	9
4.4	Display Events	9
5	User Interface Design	10
5.1	Web pages tree	10
5.1.1	Overview	10
5.1.2	Page Description	10
5.2	User Interface	10
6	Database Design	10
7	Additional Material	10
8	Requirements Traceability Matrix	10

1 Introduction

1.1 Purpose

This document specifies the entire software architecture and design for the calendar software. These design decisions directly relate to the functionalities, performances, constraints, attributes and interfaces of the system.

1.2 Scope

This document describes the software architecture and design for the initial release of the calendar software, version 1.0. The intended audience of this document exclusively includes the designers, the developers and the testers of the software.

This SDD document mainly follows the standards of the IEEE 1016-1998 standard..

1.3 Reference materials

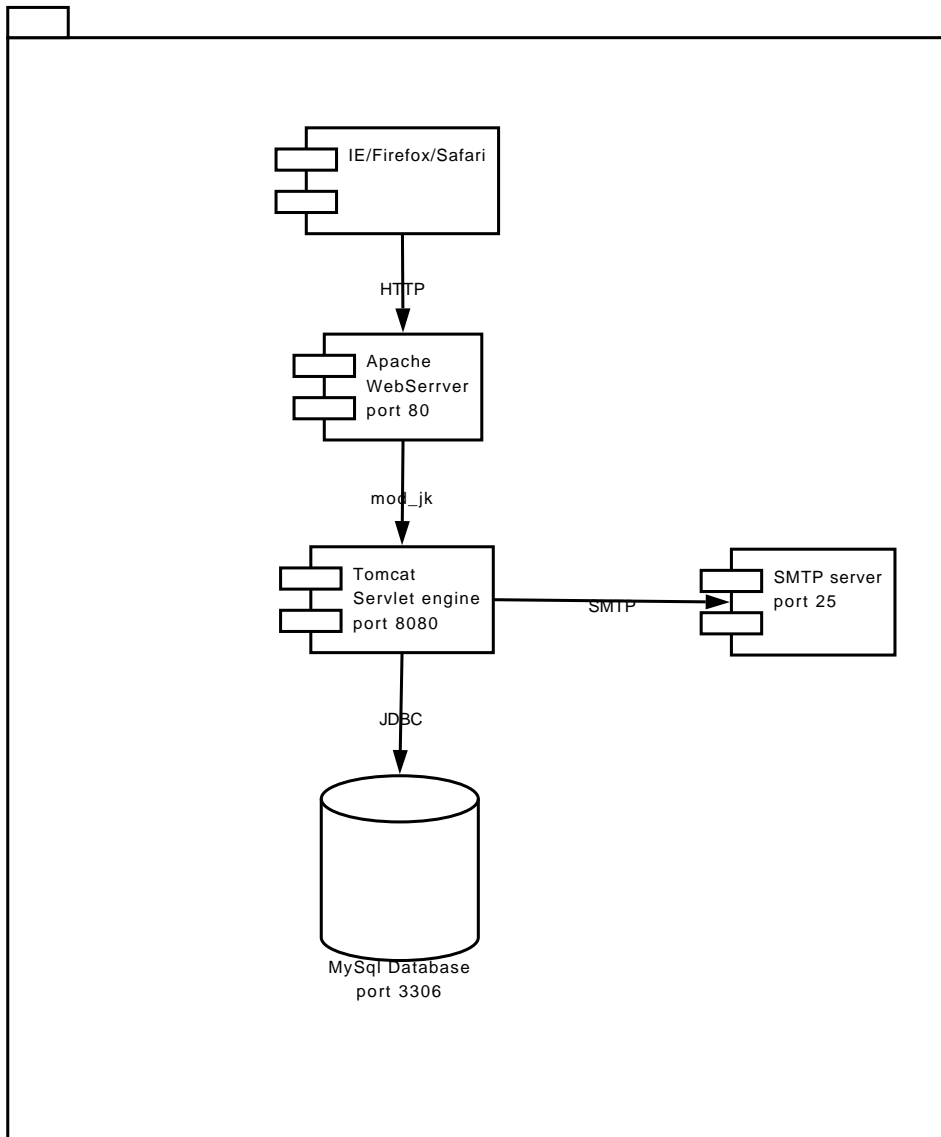
- IEEE Std. 1016-1998 - Software Design Descriptions
- http://tinf2.vub.ac.be/~dvermeir/courses/software_engineering/slides.pdf [23 Oct 2008].
- Eric J. Braude , *Software Engineering - An Object-Oriented Perspective*, John Wiley and Sons, 2001.

1.4 Definitions and acronyms (listed alphabetically)

SCMP	Software Configuration Management Plan
SDD	Software Design Document
SMTP	Single Mail Transfer Protocol
SPMP	Software Project Management Plan
SQAP	Software Quality Assurance Plan
SRS	Software Requirements Specification
STD	Software Test Document
VCS	Version Control System
QA	Quality Assurance
MDA	Mail Delivery Agent

2 System Architecture

2.1 Overview



2.2 Three-Tier Architecture

Many traditional web-based enterprise software systems have used the three-tier architecture since the 1990s. This client/server architecture model promotes the separation of responsibilities into three distinct layers. Each layer of the system depends only on the layer directly below it; layers may not depend on layers above them. The tiers are the presentation layer, business layer, and data layer. Components in the presentation layer handle interacting with the user, validating input data and localization of data to display. Business layer components provide business logic and processing of data. Data layer components provide access to the database. Benefits of using this architectural style include increased performance, flexibility, maintainability, reusability, and scalability.

2.3 Discussion of Alternative Designs

2.4 Survey of Technologies Used

2.4.1 Presentation Layer

- JSP JavaServer Pages (JSP) is a technology from Sun that allows easy development of GUIs using web-browser clients. JSP helps in making a clear distinction of layout and code concepts, so that a developer and graphical designer can independently work on the same webpages with little interference.

2.4.2 Business Layer

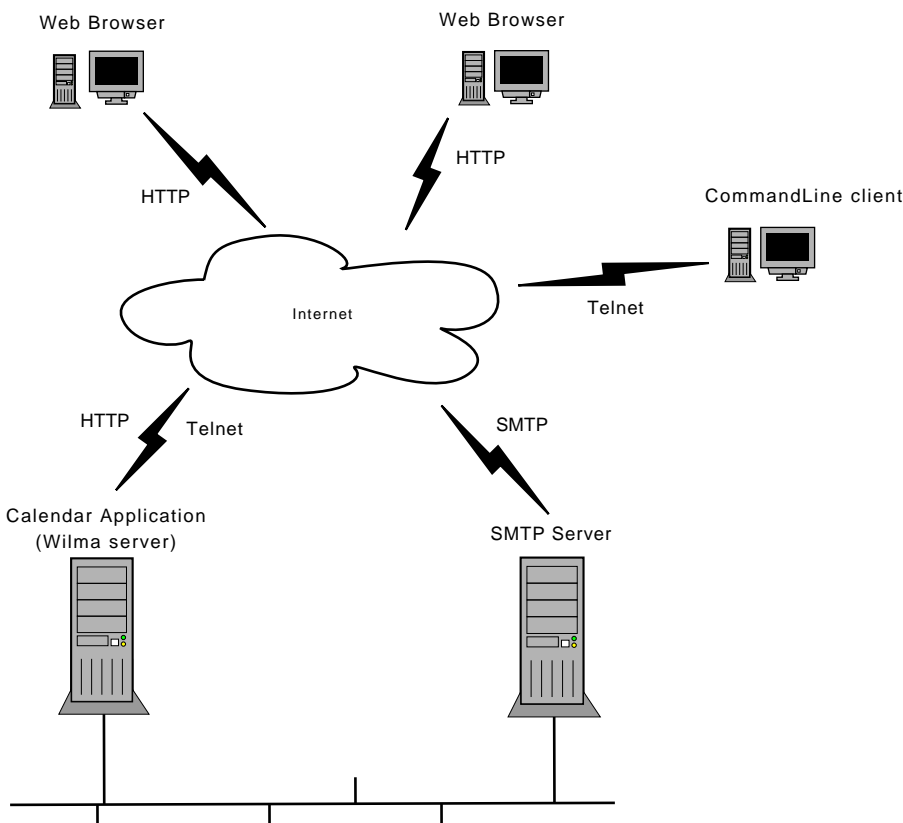
- Apache is used in over 50% of all websites worldwide (according to Netcraft). It's stable, scalable and performs really well. In our current setup we don't really need the apache server in front of Tomcat. Tomcat is perfectly capable to act as a frontside webserver. The reasons why we chose to add apache in the architecture are:
 - Because this is the normal production setup for Tomcat applications.
 - It will allow us to serve static pages faster,
 - It will help us to grow the site more easily in the future
 - We can use apache's built-in feature set to ensure security of our website.
- Tomcat is a multi-thread Servlet container. It's written in Java and thus requires a Java Virtual Machine, separate from the apache server process. Tomcat will be the host container for our application's presentation and business layer, including the interface with the data layer. While Tomcat itself is multi-threaded, each individual request from a user's webbrowser is handled by one single thread. This gives us a simple development model, while ensuring performance and scalability. To bridge the apache process (C code) and the tomcat process (Java code), the mod_jk module for apache is required.

2.4.3 Data Layer

- For the database multiple options were available: MySQL, PostgreSQL, Oracle, MS SQL Server. The two commercial products were not selected because of the high associated cost and the non-functional requirement to use open source building blocks for the software. The choice between PostgreSQL and MySQL was difficult to make based on criteria as robustness and scalability, as both have proven themselves time over time in much bigger scenario's as what we need. In the end we chose MySQL because this is the database software we're most familiar with. JDBC Hibernate

2.5 Physical Configuration

The software is to be installed on the Wilma server at the VUB datacenter. Wilma is accessible over the internet through web browsers and command line clients. The software relies on the availability of an SMTP server on the local LAN of the VUB, so that we do not have to include MDA (Mail Delivery Agent) functionality in the software.



2.6 System Interface Description

3 Component Design

3.1 Main Servlet

[[[Main Servlet dispatches all page requests to the correct JSP's...]]]

3.2 Notification Service

[[[Notification Service sends emails to site subscribers...]]]

[[[Hier dieper ingaan op het Object Model]]]

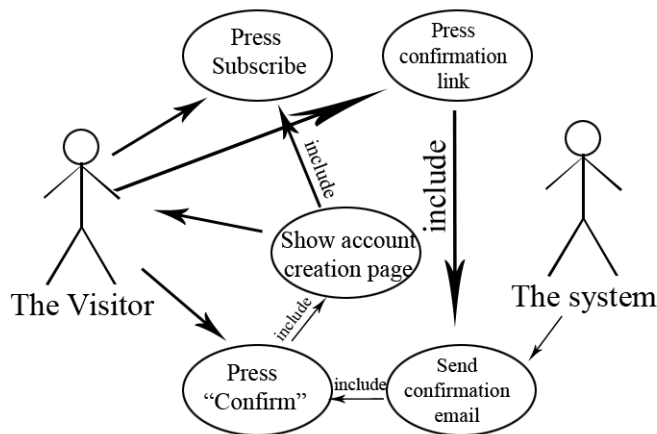
4 Software Features

[[[Hier all use cases toevoegen...]]]

4.1 Subscribe

- UML Diagram

Subscription uml diagram



- Brief description

This use case is based on FR1 of the SRS. Visitors can subscribe themselves.

- Actors

The visitors

The system

- Pre-conditions

The visitor has an e-mail adress.

- Basic flow

This use case starts when an actor wishes to subscribe.

1. The visitor selects the "Subscribe" link on the home page of the calender website.
2. The visitor is shown a screen where he can create an account.
3. The visitor must enter his name, password, second name, age, sex, profession, phone number, address and email address.
4. The visitor selects "Confirm".
5. The system sends a confirmation email to the visitor.
6. The visitor clicks on a link in the confirmation email.

7. The visitor is now activated.

- Alternative flow

If the email address is already in use, the visitor is given a notification on the account creation page. The visitor must then enter a different email address to continue the subscription process.

If not all profile entries are given by the visitor before selecting "Confirm", the visitor is given a notification on the account creation page. The visitor must then add the remaining entries to continue the subscription process.

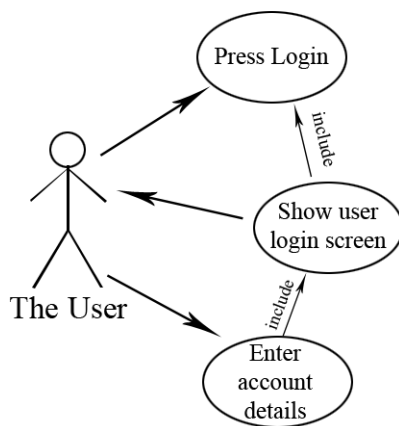
- Post-conditions

The visitor data is placed in the system database.

4.2 Login

- UML Diagram

Login uml diagram



- Brief description

This use case is based on FR4 of the SRS. A user can log onto his account.

- Actors

The user

- Pre-conditions

The user is not logged in.

The user is subscribed.

- Basic flow

This use case starts when an actor wishes to log onto his account.

1. The user selects the "Login" link on the home page of the calendar website.
2. The user is shown a screen where he can enter his name, password and function.
3. The user enters his account details.
4. The user selects "OK".

- Alternative flow

If the name-password-function combination is not correct, the user is denied access. The user will remain at the login page and a login failure notification is displayed.

- Post-conditions

The user is logged in.

4.3 Create Event

- Brief description

This use case is based on FR6 of the SRS. A user can create (schedule) an event in his agenda. This use case is a simple model for the actual FR6, which has some extra functionalities.

- Actors

The user

- Basic flow

This use case starts when an actor wishes to add an event to his own agenda. The system requests that the actor

1. enters a name for the event (type:string)
2. chooses a starting date (type:date) and time (type:time) for the event
3. chooses an ending date (type:date) and time (type:time) for the event

- Alternative flow

If the ending date precedes the starting date, the system displays an error message. The actor can choose to cancel the operation or to select a new starting and/ or ending date.

- Pre-conditions

The user is logged in.

- Post-conditions

If the Create Event was successful, the event is added to the agenda of the user. If not, the system state is unchanged.

4.4 Display Events

- Brief description

This use case is based on FR9 of the SRS. A user can display his own agenda. This use case is a simple model for the actual FR9, which has some extra functionalities.

- Actors

The user

- Basic flow

This use case starts when an actor wishes to display all events of his own agenda. The system requests that the actor pushes a button (message:'display calendar') The systems responds by displaying an ordered (in order of date and time) list of all events of his own agenda.

- Alternative flow

If the agenda is empty, a message (text:'there are no items in your calendar') will be displayed.

- Pre-conditions

The user is logged in.

- Post-conditions

If the agenda contains one or more items, a sorted list will be displayed. If the agenda is empty, a message will be displayed.

5 User Interface Design

5.1 Web pages tree

5.1.1 Overview

[[[Hier de rest van de pagina's toevoegen]]]



index.html



login.jsp



events.jsp



hotlist.jsp



profile.jsp



profile.jsp



logout.jsp

5.1.2 Page Description

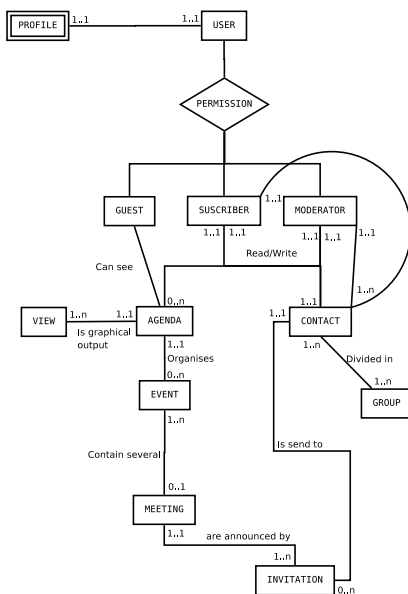
1. index.html: First page of the website, explains the product and allows the user to subscribe or log in.
2. product.html: Product description, more information to the user.
3. login.jsp: Each of the following pages will check whether the user was successfully logged in to the website. Login.jsp will
4. logout.jsp: Shows that the user was successfully logged out, and guides him/her back to the main page.
5. subscribe.jsp: A wizard (one or multiple steps) that explains the user how to subscribe and guides him/her through the process. [[[Hier de rest van de pagina's toevoegen]]]

5.2 User Interface

[[[screenshots van de belangrijkste pagina's.]]]

6 Database Design

[[[Hier Jeffrey's ERD tekst toevoegen.]]]



7 Additional Material

8 Requirements Traceability Matrix