Universita' degli Studi di Messina
Dipartimento di Matematica e Informatica

Programming II project

NetLayer

11 June 2015

Authors:
Vittorio Romeo

Professors:
Massimo Vilari

http://vittoriomeo.info

http://unime.it

Contents

Part I

Project specifications

The following part of the document describes the project and its design/development process without exploring its implementation details.

The part begins with a synthesis of the **client request**. After a careful analysis of the request, a **Software Requirements Specification** (SRS) was written.

Writing a correct and informative SRS is of utmost importance to achieve an high-quality final product and ensuring the development process goes smoothly.

The SRS will cover the following points in depth:

- **Scope and purpose**.
- **Feature and functions**.
- **External interface requirements**.
- **Functional requirements**.
- **Example use cases**.
- **Non-functional requirements**.
- **Analysis models**.

Chapter 1

Client request

The client requests the design and implementation of a **forum creation/management framework** and a **modern responsive web forum browsing/management application**.

The client intends using the requested forum framework **to build communication platforms** for various projects, both for internal employee usage and interaction with the public.

It is imperative for the system to allow administrators to easily well-organized create **content-section hierarchies** and **user-group hierarchies**.

Administrators also need to be able to **give groups specific permissions for every section**.

Some sections will only be visible and editable to employee groups (e.g. internal discussion), some sections will be visible but not editable by the public (e.g. announcements), and others will need to be completely open to the public (e.g. technical support).

Being able to **keep track of user-created content** is also very important for the client.

Initially, tracking the date and the author of the content will be enough, but the system has to be designed in such a way that inserting additional creation information (e.g. browser/operating system used to post) will be trivial.

In the future, additional content types (e.g. videos, attachments) may be added to the system and their creation will have to be tracked as well.

Users and moderators will also need to be able to track user content through a **real-time notification system** directly from the web application interface.

This data needs to be independent from the contents, in order to easily allow administrators and project managers to gather statistical data on forum usage.

The web application has to be extremely simple but flexible as well. Administrators need be able to perform all functions described above through a responsive admin panel.

Content consumers and creators should be able to view and create content from the same responsive interface.

Moderators and administrators should be able to edit and delete posts through the same interface as well. User interface controls will be shown/hidden depending on the users permissions.

Chapter 2

Software Requirements Specification

1. Introduction

1..1 Software engineering

Software engineering is the study and an application of engineering to the design, development, and maintenance of software.

The Bureau of Labor Statistics' definition is Research, design, develop, and test operating systems-level software, compilers, and network distribution software for medical, industrial, military, communications, aerospace, business, scientific, and general computing applications.

Typical formal definitions of software engineering are:

- The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software.
- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.
- An engineering discipline that is concerned with all aspects of software production.
- The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.

1..1.1 Background

The term **software engineering** goes back to the '60s, when more complex programs started to be developed by teams composed by experts.

There was a radical transformation of software: from **artisan product** to **industrial product**.

A software engineer needs to be a good programmer, an algorithm and data structures expert with good knowledge of one or more programming languages.

He needs to know various design processes, must have the ability to convert generic requirements in well-detailed and accurate specifications, and needs to be able to communicate with the end-user in a language comprehensible to him comprehensible.

Software engineering, is, however, a discipline that's still evolving. There still are no definitive standards for the software development process.

Compared to traditional engineering, which is based upon mathematics and solid methods and where well-defined standards need to be followed, software engineering is greatly dependent on personal experience rather than mathematical tools.

Here's a brief history of software engineering:

- **1950s**: Computers start to be used extensively in business applications.
- **1960s**: The first software product is marketed.

  IBM announces its unbundling in June 1969.
- **1970s**: Software products are now regularly bought by normal users.

  The software development industry grows rapidly despite the lack of financing.

  The first software houses begin to emerge.

1..1.2 Differences with programming

- A programmer writes a complete program.
- A software engineer writes a software component that will be combined with components written by other software engineers to build a system.
- Programming is primarily a personal activity.
- Software engineering is essentially a team activity.
- Programming is just one aspect of software development.
- Large software systems must be developed similar to other engineering practices.

1..2 SRS

This **Software Requirements Specification** (SRS) chapter contains all the information needed by software engineers and project managers to design and implement the requested forum creation/management framework.

The SRS was written following the **Institute of Electrical and Electronics Engineers** (IEEE) guidelines on SRS creation.

1..3 Purpose

The SRS chapter is contained in the **non-technical** part of the thesis.

Its purpose is providing a **comprehensive description** of the objective and environment for the software under development.

The SRS fully describes **what the software will do** and **how it will be expected to perform**.

1..4 Scope

1..4.1 Identity

The software that will be designed and produced will be called **veeForum**.

1..4.2 Feature extents

The complete product will:

- Provide a framework for the **creation and the management of a forum system**.
- Allow its users to **deploy and administrate** multi-purpose forums.
- Give access to a **modern responsive web application** to setup, browse and manage the forum.

veeForum, however, will not:

- Provide infrastructure or implementation for a complete blog/website. The scope of the software is forum building.
- Implement instant private messaging - user-to-user chat is beyond the scope of the project.

1..4.3 Benefits and objectives

Deploying veeForum will give its users a number of important benefits and will fulfill specific objectives.

- Companies and individuals making use of veeForum will have access to an **easy-to-install** and **easy-to-use** forum creation and management platform.
- Users and moderators of the deployed forums will be able to **easily create, track and manage** content and other forum users.
- Forum administrators will be given **total control** of the forum structure, users and permissions through an **easy-to-use** responsive administration panel.

2. General description

2..1 Product perspective and functions

The product shares many basic aspects and features with existing forum frameworks such as **phpBB** or **vBulletin**: flat/threaded discussion support, nested sections, user attachments, etc.

veeForum improves on existing forum frameworks in the following ways:

- Provides a responsive web interface without postbacks.
- Allows users and moderators to subscribe and unsubscribe not only to posts, but to users and sections as well.
- Has a powerful **real-time** Facebook-like notification system that notifies users when tracked content has been added or edited.
- Gives administrator the possibility to **design and manage complex permission hierarchies** for user groups and single users.

2..2 User characteristics

veeForum needs to target both users that **only consume the content offered by deployed forums**, users that **actively create and manage content in deployed forums**, and users that **build and deploy forum instances**.

User-friendliness is essential for every target, but all the required functionality is effectively exposed to different user groups.

It is therefore required to have clear interfaces that do not negatively affect the user experience by being either too complex or too simple (all features need to be exposed).

3. Glossary

- **User**: the system requires user registration and authentication to be used. An **User** represents a person using the system. He/she needs to be registered to the system in order to sign in. An **User** produces **Content** and always belongs to a single **Group**. **Users** can subscribe to **Content** using **Subscriptions**, to receive **Notifications**.
- **Group**: a **Group** is a collection of 0 or more **Users**. It defines **Privileges** for the **Users** in the group and **Permissions** related to every section.

- **Privilege**: a **Privilege** is a bit representing the permission to perform system-wide operation or access a system-related area. Possible values are:
  - **Superadmin**: can access any area of the system. Overrides all **Privileges** and **Permissions**.
  - **Manage sections**: can create/remove **Sections**.
  - **Manage users**: can create/edit/remove **Users**.
  - **Manage groups**: can create/edit/remove **Groups**.
  - **Manage permissions**: can create/edit/remove **Permissions** from **Groups**.
- **Permission**: a **Permission** is a bit representing the permission to perform an action, and it is always related in context to a specific **Section**. Possible values are:
  - **View**: the target **Section** can be accessed in read-only mode.
  - **Post**: **Posts** can be created in the target **Section**.
  - **Create Thread**: **Threads** can be created in the target **Section**.
  - **Delete Post**: **Posts** can be deleted in the target **Section**.
  - **Delete Thread**: **Threads** can be deleted in the target **Section**.
  - **Delete Section**: the target **Section** itself and subsections can be deleted.
- **Content**: represents any data created by **Users**. It can be specialized in **Attachments**, **Threads** and **Posts**. **Content** can have zero or more **Tags**.
- **Tag**: a string that labels one or more **Content** instances.
- **Section**: a collection of zero or more **Threads** and zero or more subsections. It represents the basic building block of the forum structure.
- **Thread**: a collection of zero or more **Posts**.
- **Post**: a container of textual content and zero or more **Attachments**.
- **Attachment**: a downloadable file. The internal data is stored in a **File data**.
- **File data**: binary storage for a file.
- **Subscription**: created by an **User**, the subscriptor, who targets one of the following entities: **User**, **Thread** or **Tag**. When **Content** related to the targeted entities is created, **Notifications** are generated.
- **Notification**: represents an alert that new **Content** related to an existing **Subscription** has been created. Used to notify **Users** of tracked content changes.

4. Specific requirements

4..1 External interface requirements

External interface requirements identify and document the interfaces to other systems and external entities within the project scope.

4..1.1 User interfaces

The product will provide both a desktop and a mobile user web interface.

- **Web interface**: it is required to provide a modern responsive web interface, compatible and tested with the most popular browsers (Internet Explorer 10+, Google Chrome, Mozilla Firefox). The web interface will give forum access to users and moderators, and administrator access to forum management staff.
- **Mobile interface**: it is required to provide a modern mobile application for the major platforms (Android, iOS, Windows Phone). The mobile application will allow browsing and content management of forums created with the product.

4..1.2 Software interfaces

The **open-source policy** of veeForum will allow framework users to expand or improve existing functionality and to interact with other existing technologies.

Accessing and modifying forum data (assuming permission requirements are satisfied by the user) will be possible through **RESTful** requests, returning and accepting **JSON** (Javascript Object Notation).

4..2 Functional requirements

In software engineering, a **functional requirement** defines a function of a system and its components.

Functional requirements may be **calculations**, **technical details**, **data manipulation and processing** and other specific functionality that define what a system is supposed to accomplish.

Behavioral requirements describing all the cases where the system uses the functional requirements are captured in **use cases**.