Post-Doctoral Application Management System

Software Documentation Document

May 18, 2014

Version 0.9

Iteration 1

Prepared for Ms. Cathy Sandis (UP Research Office)
by SoftServe Group

Group members

Kgothatso Phatedi Alfred Ngako (12236731)

Tokologo "Carlo" Machaba (12078027)

Mathys Ellis (12019837)

Change log			
Date	Version	Description	Person
10/02/2014	v 0.0	Document created.	Alfred Ngako

Contents

List of Figures

1 Document description:

1.1 Document purpose

This document provides the documenting for the software architecture infrastructure which the application functionality is deployed and executed.

1.2 Documentation methodology

The documentation and software development methodology used by the project adhere to the guidelines set out by the agile method. Thus this document has undergone and will undergo various iterations that may extend or reduce the contents of the document.
The document was compiled using a software architecture specification document template provided by Dr Fritz Solms as an alternative to the Kruchten 4 + 1 approach to documenting software.
This document was created using the requirement elicitation techniques and requirement definitions as specified by Klaus Pohl's book Requirements Engineering: Fundamentals, Principles, and Techniques [Dr.Phol, K., 2010]. The requirements, vision and scope were elicited from the following sources:

- Numerous interviews with the client.
- On-line research into UP Post doctoral applications.
- Correspondence with the UP IT department.
- Collecting and analysing various documents such as:
  - The initial project request document
  - Application forms
  - Renewal forms
  - CV templates
  - Approval and recommendation forms

1.3 Document conventions:

- Documentation formulation tool: LaTeX

1.4 References:

- Dr.Phol, K., 2010, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer, Heidelberg.

- http://www.javaworld.com/article/2074488/enterprise-java/enterprise-application-integration-using-j2ee.html

2 Architecture Requirements

This section discusses the software architectural requirements from the software requirements such as:

- Architectural Scope,
- Quality Requirements,
- Integration and Access Channel Requirements, and
- Architectural Constraints.

All the above mentioned topics are put in place to address the non-functional requirements that were illicited from Ms. Cathy Sandis (UP Research Office).

2.1 Architectural Scope

The scope that the architecture needs to cover include:

- A persistence infrastructure (Database) to facilitate domain objects (e.g CVs, DRIS information, and Applications). This will also allow the implementation of the Audit trail required by Ms. Cathy Sandis (UP Research Office)and the centralised point for all the required documentation shared among stakeholders.
- A session oriented infrastructure to assist in realizing the security requirements of authenticating participants and their actions.
- A web access channel which will provide the client, and other stakeholders, with an interface to the underlying system.
- An infrastructure for the generation of reports.
- A mailing competent infrastructure.

2.2 Quality Requirements

The following are the requirements around the quality attributes of the systems and the services it provides:

2.2.1 Availability:

The system's availability on designated networks will depend on the availability of the University of Pretoria's servers that host the system. If the University of Pretoria's servers hosting the system are active and provide access over a designated network then the system must be available over that designated network. The designated networks are defined as the internet and the campus network of the University of Pretoria.

2.2.2 Security requirements

The system will need to be fully secured since the system deals with confidential information such as person information, application statuses, financial data and meeting information. Also since the systems main goal is to provide stable and audible application and renewal process flow the system may not be vulnerable to data tampering or any tampering whatsoever.

The system will have to provide different security roles to the registered users on the system. Any number of roles should be assignable to any user by a administrator with the correct role to allow for flexibility. But in essence a stakeholder may only have access to their section of the application process. The system administrator should be able to view all the sections in the system and should be able to modify them except where they may not.

2.2.3 Scalability requirements

The current aim is to create a scalable system that can support 500 to 1000 applicants per year with possible growth. This is in line with the figures given by the client and the growth in the research sector of the university.

The system needs to be scalable in regard to the following factors:

- **Performance:** This is regarded as the speed and responsiveness of the system. The system needs to be handle report queries in less than 10 seconds. It should be able to handle any application section processing in less than 3 seconds.
- **Storage:** This is regarded as the growth and shrinking of the data that is stored. The system will need to be able to handle a database that is in the range of 1 GB to 15 GB that has the potential to grow even larger. The reason for this stems from the requirement that the system will support archival functionality and archived data will store the data for long periods of time.

- **Concurrency:** This regarded as the amount of active users on the system at the same time. The system will need to support at least 100+/- concurrent users efficient and effectively since the system requires multiple stakeholders to part take in the application process while there can be multiple applications occurring at the same time.

2.2.4 Testability:

The system must be testable. This will be done using unit testing and following the test plan that will be laid out in the testing document of this project.

Unit testing will test each unit in regard to:

- **Preconditions**
- **Post conditions**

The project will also have two phases of testing:

- **Offline:** This is the initial phase of testing and debugging which will be done with pseudo data.
- **Online:** This is the final phase of testing and debugging which will be done with active real time data.

2.2.5 Auditability:

The system needs to provide an audit trail of all critical actions that occur in the system. Critical actions are considered: user account management operations, login action, logout action and any operation by a user that leads to a change in application data of a particular prospective fellow.

The Audit trail will be in the form of a read-only table stored in the database. It can only be viewed by a user with the correct security role. The system is the only entity that can modify the audit trail where this modification can only be the addition of entries.

2.2.6 Usability requirements

The primary language of the system will be South African English. Any other language support is not considered part of the requirements but the system will be designed to allow for such development in the future.

The system's UI will only consider 2 types of user categories with regard to usability:

- **Trained user:**

  This type of user will have to have training in understanding how the system functions and how to use it. Their computer skills will be assumed to be in the range of basic to intermediate. Thus the user interface can allow for certain complexities but these complexities must be kept at a minimal. This user will be regarded as a system administrator. The stakeholders who fall under this category is the DRIS staff members overseeing the application process.

- **Normal user:**

  This type of user will have no training. Their computer skills will be assumed to be none or minimal. Therefore the UI that they will have access to will be simplistic and will be as user friendly as possible. The stakeholders that fall under this category will be Prospective fellows, Grant holders, HODs, Deans and Post-doctoral committee members.

2.3 Integration and Access Channel Requirements

This section discusses

1. the requirements for the different channels through which the system can be accessed by humans and other system, and
2. the integration channels which must be supported by this system.

2.3.1 Access Channels

The system must be reachable securely through a web browser.

2.3.2 Integration Channels

Upon completion the system must be integratable with the University of Pretorias PeopleSoft system.

2.4 Architectural Constraints

Ms. Cathy Sandis (UP Research Office)did not specify any Architectural constraints other then the fact that the system must be integratable with the University of Pretoria, packaged software solution, PeopleSoft System.

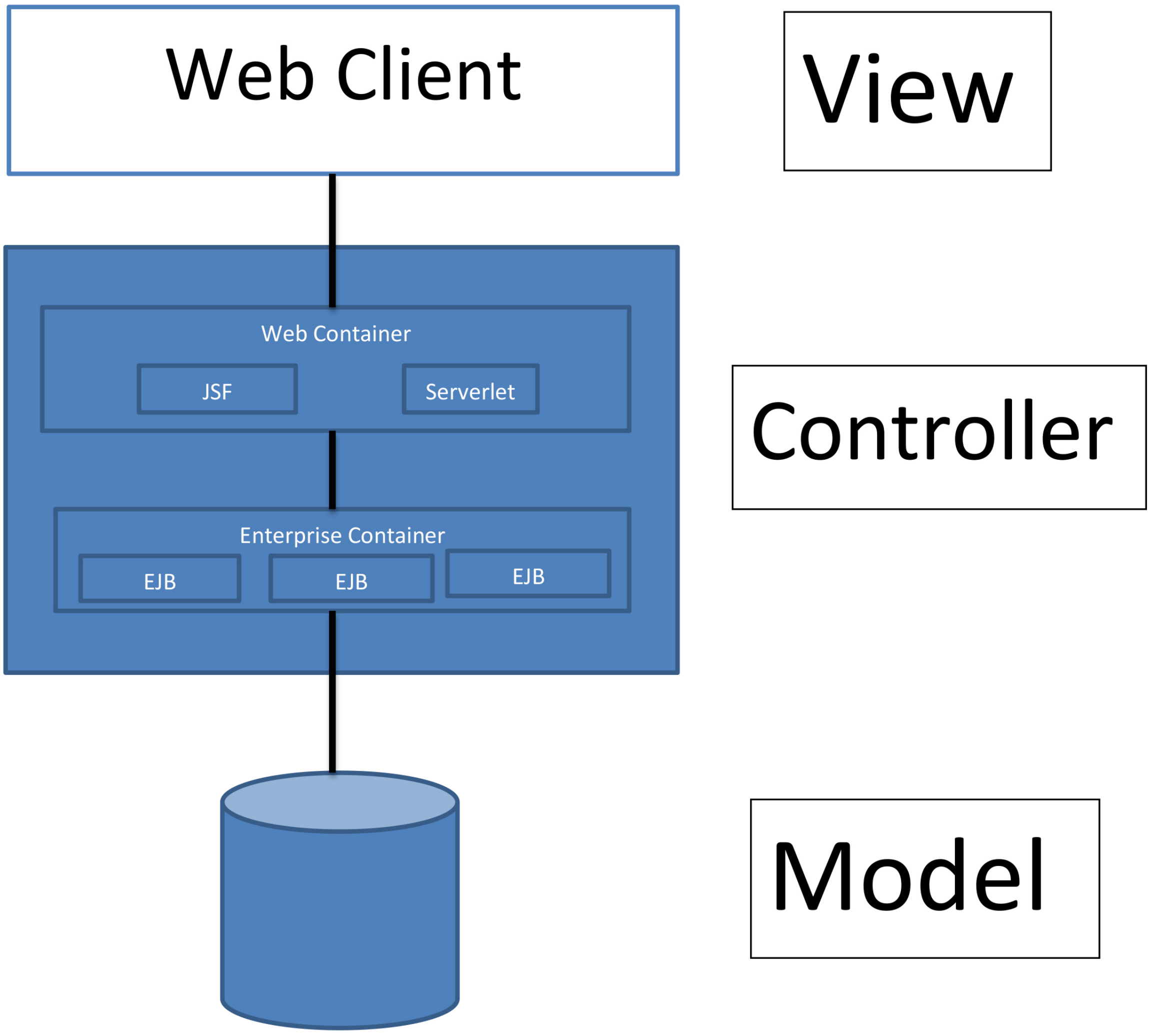

Figure 1: Model View Controller of Architecture

3 Architectural Patterns and Styles

The system will maintain a MVC architectural pattern specific to how Java EE is structured. Which is a layered structure that automatically decouples the client from the server.

The Java EE structure is designed to satisfy the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. Which is reason to why it suits its use in the development of the system. The following diagram provides the abstracted architectural structure which will be followed:

View

This component provides an interface to the under laying system to the user. This client is the web browser. It also incorporates components which are in the Web container(controller), such as Java Sever Faces.

Controller

This component is made up of two containers, the web container and the Enterprise container.

Model

This component is the actual database on which the system runs on. Which will be used as the centralised point of access with regards to applications being processed by the system as required by Ms. Cathy Sandis (UP Research Office).

4 Architectural Tactics and Strategies

This section describes techniques which will be used to satisfy the quality requirements. It the time of printing this document no architectural tactics/strategies have been identified as the architectural patterns and styles should be sufficient to satisfy the requirements.

5 Use of Reference Architecture and Framework

Java EE provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications. It also provides an architecture for implementing services as multitier applications that deliver the scalability, accessibility, and manageability needed by the system. This is reason to why it is ideal to use it in the development of the project.

6 Access and Integration channels

This section discusses the requirements for the channels through which the system can be accessed by humans and other systems. Also making mention about the integration channels which need to be followed.

6.1 Access Channels

The system will be accessible by humans through the recent versions of the following browsers:

1. Mozilla Firefox,
2. Google Chrome,
3. Microsoft Internet Explorer

4. Apple Safari, and
5. Opera

The mobile counterparts of the above mentioned browsers will also be catered for and so no other access channel (such as Android/Apple apps) is to be considered.

6.2 Integration Channels

Upon successful complementation the system must be integratedable with the University of Pretoria's PeopleSoft system. Therefore it is sensible to implement the system in Java EE as it is one of the many components that form part of the multiple tiers that build up PeopleSoft.

The use of build tools such as Maven will provide a central piece of information with regards to the projects build, reporting and documentation. This central point of information will assist in the integration phase since it is implemented through a standardised build approach which can come into play at a later time through the use of Enterprise Application Integration.

The EAI will be achieved via the logical integration architecture of Direct point-to-point integration. This means that the application management system will make direct JDBC (Java Database Connection) calls to the universities databases tables (which needed to be setup to cater for our system at that point). The Integration method will be pushed-based data-level integration (or if all else fails UI-Level integration).

7 Technologies

The System will use the following technologies:

- Java Enterprise Edition 7
- MySQL database
- Netbeans 8.0
- Glassfish server

Integrated Development Environment

The system should be buildable independent of an IDE but it will be developed on Netbeans 8.0 to allow for uniformity amongst the development team, with regards to coding style, and provide easy integration with the tools that will be used such as Javadoc, to generate API documentation in HTML format.

API

- JavaMail
- JavaBeans Activation Framework
- Java Database Connectivity
- Java Naming and Directory Interface
- Java API for XML Web Services
- SOAP with Attachments API for Java
- Java Authentication and Authorization Service

Build Tools

- Apache Maven (for reasons explained in the integration channels section).

Operating System

The system will be deployable on:

- Windows 7 and 8.
- Linux based operating systems (specifically Kububuntu 13).

8 Glossary:

- **API** - Application Programming Interface
- **Application** -Both renewal applications or new fellowship applications are seen as applications by this project.
- **CV** - Curriculum Vita
- **HTML** - Hyper Text Mark-up Language
- **Java EE** - Java Enterprise Edition
- **UP** - University of Pretoria