Post-Doctoral Application Management System

Architecture requirements specification

August 1, 2014
Version 1.0
Iteration 2

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
06/03/2014	v 0.0	Added quality requirements in old SRS document	Mathys Ellis
12/05/2014	v 0.0	Document created	Mathys Ellis
13/05/2014	v 0.1	Transferred quality Requirements, Access Channel requirements to document	Carlo Machaba
14/05/2014	v 0.2	Added Integration requirements	Carlo Machaba
15/05/2014	v 0.3	Added to Glossary	Carlo Machaba
15/05/2014	v 0.4	Added Document description, purpose, methodology, conventions and references	Carlo Machaba
17/05/2014	v 0.5	Added more Quality Requirements	Carlo Machaba
20/05/2014	v 0.6	Added more Quality Requirements	Carlo Machaba
22/05/2014	v 0.7	Added and improved Quality Requirements	Mathys Ellis
22/05/2014	v 0.8	Transferred architectural scope to this document	Mathys Ellis
22/05/2014	v 1.0	Edited, formatted and added some details. Finalised document for first iteration	Mathys Ellis

Contents

1 Project Repository **4**

2 Document description: **5**

2.1 Document purpose: 5

2.2 Documentation methodology 5

2.3 Document conventions: 6

2.4 References: 6

3 Architecture requirements **7**

3.1 Architectural Scope 7

3.2 Access channel requirements 7

3.3 Quality requirements 8

3.3.1 Usability requirements 8

3.3.2 Security requirements 9

3.3.3 Audit-ability: 9

3.3.4 Scalability requirements 10

3.3.5 Availability: 11

3.3.6 Robustness: 12

3.3.7 Testability: 12

3.3.8 Flexibility: 13

3.3.9 Re-usability: 13

3.3.10 Maintainability: 14

3.4 Integration requirements 15

3.5 Architecture constraints 16

4 Glossary: **17**

List of Figures

1 Project Repository

https://github.com/mox1990/Project-Postdoc.git

2 Document description:

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

2.1 Document purpose:

The Architecture Requirements Specification provides the architectural requirements that the project will need to follow as well as a set of quality requirements that must be met by the system. The quality requirements have been quantified in order to provide the SoftServe group and the client with a means to test the system for non-functional compliance. This allows the SoftServe group to set up tests in the non-functional testing document that can be used at the end of the development cycle to verify that the system complies with the architecture and quality requirements specified by this document. Architecture in this document's context refer to the technological basis and software development infrastructure and styles of the project. Thus this document serves as a contract between SoftServe and the client, Mrs Cathy Sandis of the DRIS of the University of Pretoria in terms of what technologies the project should incorporate as-well as what quality the system should be of once the project is complete.

2.2 Documentation methodology

The documentation and software development methodology used by the project adhere to the guidelines set out by the scum agile methodology. Thus this document has undergone and will undergo various iterations that may extend or reduce the contents of the document.
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 quality requirements and aspects of the architectural requirements, 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

2.3 Document conventions:

- Documentation formulation tool: LaTeX

2.4 References:

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

3 Architecture requirements

3.1 Architectural Scope

This section discusses the scope that the software architecture of the system needs to cover:

- A persistence infrastructure using a DBMS to facilitate the storage of the various domain objects (e.g CVs, DRIS information, and Applications). So to allow the system to store and centralise the various data that the system will be handling.
- A multi-user session infrastructure to assist in realizing the security requirements of authenticating users and their actions.
- A pipeline infrastructure for processing applications.
- A RESTful web service infrastructure, so to allow lightweight data communication between the users and the server and support for mobile platforms.
- A report generation infrastructure in order to support the data retrieval requirements of the client.
- An email infrastructure for the system to use in order to provide notification support.
- A action logging infrastructure to track user activity on the system, in order to provide audit-ability.

3.2 Access channel requirements

The only two access channel requirements specified by the client are as follows:

- **All stakeholders:** Users of the system will access the system through a HTTP and HTTPS web browser client that is locally installed on a user's computer system or mobile platform. Support for mark up language HTML 4.0.1 and 5 will be enabled through this. This will improve availability and usability of the system due it making it more compatible with older internet enabled systems. The web interface will allow different stakeholders access to different sections of the system based on the roles assigned to their accounts.
- **System Administrator:** The system administrator will require the same as the above but will need to be able to conduct any administration duties through some form of web portal using the HTTP and HTTPS protocols. Above this the system administrator will need to have access to the MySQL database employed by the system. This is mainly for maintenance reasons but also as an alternative if certain unexpected system failures or system generated errors occur. This will be accessible through the use of software which uses the appropriate technologies.

3.3 Quality requirements

These are the quality requirements as specified by the client and are also in the order of precedence. Where the first takes the highest precedence.

3.3.1 Usability requirements

This is first and foremost quality requirement stipulated by the client. 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. These user will be regarded as system administrators or specialised users of the system. The stakeholders who fall under this category is the DRIS staff members overseeing the application process and any technical or maintenance support users.

- **Normal user:**

  This type of user will have no training with regard to the system. 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, Research fellows, Referees, Grant holders, HODs, Dean's office members and Post-doctoral committee members.

In order to quantify the above the SoftServe group will need to provide the system with a uniform user experience that is as simplistic and as visually oriented as possible without negating the below requirements severely. In addition to this any form of help text or instruction should be written in as natural, uncomplicated and unambiguous manner as possible. A further quantification of this requirement can be seen as finding the least number of clicks a user has to perform in order to perform any operation. Thus to meet this requirement it should be considered local tasks should take a maximum of five clicks and distant tasks should take up to a maximum of 8 clicks. Where local refers to tasks that are related to the current section begin accessed with in the system. These clicks should not include the clicks need to enter various lengthy form data.

3.3.2 Security requirements

The system will need to be fully secured since it 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. It should be noted that this requirement conflicts with the Usability requirement above as it is known that the more secure a system becomes the less user friendly it becomes. Therefore the SoftServe group will try to balance the two requirements in such a way that neither of the two is negated.

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 system administrator to allow for flexibility.
There will also need to be predefined roles so to allow a stakeholder from list of stakeholders in the vision and scope document, to only have the ability and access to perform the actions as defined by the vision and scope document.
The system administrator should be able to access all the sections in the system and should be able to modify them except where only the system itself has that right.
Another security requirement is that the audit trail must not be allowed to be modified by any human user and only allow the system to be able to insert data into the audit log.
Further the audit table should only allow be query-able by a user with the correct security role.
To further quantify this requirement the system must provide some means of a centralised user authentication that acts as a gateway for all users of the system. Also every time some action is made by the user the user needs to be re-authenticated to insure that the user does have the correct privileges and if any user account changes where made it takes immediate affect.
Any form of account retrieval token or on-demand user account creation tokens must expire within a specified time period. For account retrieval token this should be 1 hour.
For on-demand user account creation this should be about 2 days. This requirement will

influence the usability requirement negativity as there will be need for more user forms and interaction. But the impact of this requirement must be minimized as much as possible.
Further this requirement will also negativity impact performance as it will result on more code command execution per action that needs to be secured.

3.3.3 Audit-ability:

This is seen as the third most important requirement according to the client. The system needs to provide some form of an audit trail of all critical actions that occur in the system.
Critical actions are considered: user account management operations, login action, logout

action as well as any operation by a user that leads to a change in the data stored by the system. The Audit trail will be in the form of a read-only table stored in the database as stipulated in the security requirements above.
To quantify this requirement, the system will need to have an active audit log that can capture all the actions, that meet the list of critical actions, of each user that uses the system. With regards to usability, the feature needs to run in the background and not be noticeable by any user while they are using the system except if it is deliberately queried.

This requirement will have a negative impact on performance as it will be adding more overhead the executions of audit-able actions. But since not all the actions are audit-able it should not have a to significant impact. Though the audit logging action should be made as light weight as possible. Also adequate storage will be need since such entries can easily grow exponentially which can lead availability and other performance issues.

3.3.4 Scalability requirements

As research is still viewed as small field that is slowly growing by the client this is only the fourth most important requirement for the system. The current aim is to create a scalable system that can support 50 to 500 applicants per year with possible growth to a 1000 or more. This is in line with the figures given by the client and the growth, over the next ten to twenty years, in the research sector of the University of Pretoria.

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

- **Performance:** This aspect of the system regards the speed and responsiveness of the system. The system needs to be handle large report queries in less than 10 seconds. It should be able to handle any application approval service processing in less than 3 seconds. It should be able to send emails instantly from the system side. Note this may be difficult to measure due internet bandwidth and receiving mail server constraints. Login, logout and user authentication should be handled in less then a second by the system this does not considering the transfer of data to and from the server which is depended on the network connection speed.

- **Storage:** This aspect of the system regards the growth and shrinking of the data that is stored by the system. 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 fact that the system will need to store a large amount of active data over a period of about 3 to 8 years before it can be archived.
  Another requirement of the system is that it will need to support archival functionality and archived data that will eventual store all the old data from the active database for as long as time period as possible. Since the archival database should only grow it will need to be about 15 times larger than the active database.

Due these reasons the calculation of how much space should be required will only be possible once the system is implemented and practically tested.
The suggested linear formula to the above analysis is as follows:

$$A \times ((Xna \times Nna) + (Xra \times Nra)) = B \tag{1}$$

A = Number of years for which archival or active support is intended
B = Number of bytes needed on average for A years
Xna = Average number of bytes per new application
Xra = Average number of bytes per renewal application
Nna = Number of new applications a year
Nra = Number of renewal applications a year

- **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.

This requirement will require that the code undergo optimization and as well as certain functionality which can reduce the maintain-ability of the code itself also it can lead to reduction in terms of system robustness.

3.3.5 Availability:

The system's availability on designated networks will depend on the availability of the University of Pretoria's servers or which ever hosting service's server that will used to host the system. It should be noted that the former will only be possible if the project is considered acceptable by the UP's IT department.
In order to quantify this it should be seen as follows: If the hosting service's servers hosting the system are active and provide access over a designated network then the system must be available over that designated network. Where the designated networks are defined as the internet and/or the local network where in the server of the hosting service is located.

3.3.6 Robustness:

This requirement was identified by the SoftServe group as being very important since the system will be receiving and handling large amounts of sensitive data. Thus the integrity of the data must be insured. To do this the system needs to be robust in in terms of its data validation routines, data storage routines, security routines, and system integrity checks. These four aspects of robustness are quantified as follow:

- **Data validation**: This aspects refer to the high level data validation that occurs at a user interface level. All data that is manually imputed by a user, such as text, numbers, emails, passwords, etc. Must be checked to see if they meet constraints imposed on them through the use of data type checking and regular expression analysis. Any section in the system that requires certain data to completed by the user before performing a action must be checked for completeness before allowing the user to perform the action. Where completeness means the data is not empty and validate according to constraints imposed on it. Data is only sent to lower levels of the system after the data has been validated.
- **Data storage**: When the data is actually stored the data needs to be validated against a set of low-level storage rules and constraints. Many DBMS packages allow this type of validation through data typing and allowing the database administrator to a specify constraints. Note that this is validation is independent from the validation that occurs at the user interface level.
- **Security**: As mentioned above in the security requirements, the re-authentication of users is seen as a robustness check. Another robustness check for security is that a particular user may only have one of either a account retrieval token or on-demand user account creation token active at a time else it is regarded as fault.
- **System Integrity**: If any malfunctioning modules are detected within the system any user action will not be completed and the user will be notified immediately with the appropriate error message. So prevent any data corruption due to system faults.

3.3.7 Testability:

The system must be testable in order to show that the system is meets the clients expectations. This will be done using user acceptance testing, integration testing and unit testing. The test plan will be laid out in the Non-Functional and Functional testing documents of this project. Each of which will provide a set of tests to be performed in order to verify the systems compliance. Therefore each non-functional and functional requirement needs to be quantified in a manner that allows the SoftServe group to test for it in the system. This requirement will help ensure the other requirements are met and all in all improve the system as a whole. Below are the suggested methods quantifying the tests for each type of test:

Unit testing will test each module and its unit in regard to:

- **Pre-conditions** of each unit contained in a module.
- **Post-conditions** of each unit contained in a module.
- **Result and request objects** of each unit contained in a module.

The integration tests will test the system with regard to:

- **Module communication** between units and if their interfaces match the expected interfaces.
- **Functionality** of the system in contrast to the required functionality.

User acceptance testing will test the whole system in regard to:

- **Non-functional requirements**: If the system meets the non functional requirements as stipulated by the architectural requirements documents.
- **Functional requirements**: If each of the expected system functional requirements are met as stipulated in the functional requirement and application design document.
- **Client acceptance**: If the system is at a level that client accepts the system and wishes to deploy it.

3.3.8 Flexibility:

The system should be designed in such a way that adding new features can be done without extensive code restructuring or reverse engineering. The process of adding new features should require the programmer working on the system to easily create a new module for the new functionality. For this to be realised the system will should use appropriate design patterns that allow this. For example the strategy, dependency injection and command design patterns.

3.3.9 Re-usability:

The system's components should be designed in such away that they can be reused in different sections in the system or even in external systems. For example the user access control module should be portable to other systems that can be developed.

3.3.10 Maintainability:

The system should be maintainable in regards to 2 aspects:

- **System housekeeping**: This aspect revolves around the actually administration of the system once it is deployed. The system should prevent the system administrator from unnecessary worrying about automated tasks. But the system should allow the system administrator to be able to manage user data, perform backups manually, archival routines and system configuration manually.

- **System development**: This aspect revolves around the developers of the software. The system must be developed in such a way that the source code is easy to maintain and highly readable such that debugging and patching difficulty can be reduced. Also this ensures that the code can easily given to a third party if need be. The coding documentation standard that must be adhered is that stipulated by the JavaDoc standard. The coding style convention must be defined in such away that the code is easy to read and structured in a logical manner. The identifiers used must be named appropriately and must follow the camelCase standard except where the identifiers are constants in which case the entire name will be in capital letters. The style suggested is a variant of the Allman style:

```
public void run()
{
        /*This is a multi line commment*/
        //This is a single line comment

        for (int i = 0; i < x; i++)
        {
                int x = 0;
                int y = 0;
                while (x == y)
                {
                        if (x == y)
                        {
                            something();
                        }
                        else
                        {
                                somethingElse();
                        }
                }
        }
}
```

3.4 Integration requirements

The client's primary integration requirement is with regards to the import and export of data that can be used by the PeopleSoft management system of UP as well as other parties. The client's secondary integration requirement is with regards to the complete integration of the system with UP's PeopleSoft management system. With the regards the vision and scope document it should be noted that the knowledge the SoftServe group has of the Peoplesoft system is very limited due to UP's IT department's prerequisite for the project. Above that their are a few other integration requirements that SoftServe in conjunction with the client have purposed. The system has to integrate with the following systems as follows:

- Must be able to create exportable data packages containing the information of prospective and current research fellows that can easily be loaded into the PeopleSoft system or used by other parties. The technology interchange format purposed is CSV. This is to meet the primary requirement of the client. In terms of quality requirements:

  1. **Performance:** The system should be able to handle the export and importing of application information at a reasonable rate of about at least 10 full application data packages in a second. The system should be able to import and generate on-demand user accounts at a rate of 50 per second. For exporting it should be able to do so at 60 accounts per second.
  2. **Scalability:** The system should be able to export or import as much data as specified by the user as long as it does not exceed the size of the database or the size of users secondary storage.
  3. **Audibility:** Any export and import should be recorded by the system.

- Must be able to query PeopleSoft system to retrieve the account information of internal individual or group Stakeholders to create a uniform login experience. As specified by the vision and scope document this will not be possible at the time of writing. Therefore no API can be given. But the system will most likely make use of the above in point to achieve this requirement.

- Must assign Prospective fellows with a id number in such a way that is integrable with UP Emplid. SoftServe suggests: "f" + 8 digit code.

- Stakeholders that have a Peoplesoft account must use their UP emplid id as a user name. The password may vary.

- Must be able to provide access to the NRF researcher ratings list but not utilize it as the client has stipulated human error is possible in maintaining of those ratings.

The above integration requirements except for the second requirement does need to make use of any protocols or another system's API.

3.5 Architecture constraints

The following architecture constraints have been selected by SoftServe group as being suitable for the system. Please note due to the client's inexperience with software development and the technologies that exist she has allowed the SoftServe group to use their expertises, to decide on the appropriate set of architectural constraints that can meet the above requirements.

1. Database technology: MySQL
2. Programming paradigm: Object-Oriented
3. Programming language: Java
4. Development platform: Java-EE 7
5. System architecture: Java-EE
6. Architectural frameworks: JSF, PAM, Java Persistence API, JAX-RS
7. Architectural pattern: MVC, Layered (Multi-tiered)
8. Development technologies: EJB, JSP, Expression Language, JDBC, JTA, Java EE Connector Architecture, JAF, JavaMail, JNDI, JAAS, Concurrency Utilities for Java EE, JAXB, PrimeFaces
9. IDE: Netbeans 8.0
10. Build environment: Apache Maven 3
11. Web server software: GlassFish 4.0 server
12. Web interface protocol: HTTPS
13. Client web browser: Microsoft Internet explorer 9+, Google Chrome 30+, Mozilla FireFox 20+, Opera, Safari
14. Client device operating systems: Windows, iOS, Android, Linux.

4 Glossary:

- **NRF** - National Research Foundation
- **Spreadsheet** - A special type of computer document that is used to represent data in rows and columns.
- **GlassFish** - GlassFish is a web server software package that is very flexible and compatible with Java EE applications.
- **HTML** - Hyper Text Mark-up Language
- **HTTPS** - Hyper Text Transfer Protocol Secure is a higher level network oriented communication rule set that is highly secure and is used by all web browsers.
- **Java EE** - Java Enterprise Edition
- **MySQL** - Is a relational persistence database package that provides all the necessary management tools to run and manage a database server.
- **Object-Oriented** - A programming language style that encapsulates everything as an object instance of a particular class of attributes and methods.
- **Application** - Both a renewal and new fellowship are seen as applications.