Functional Requirements And Software Architecture Specification
COS 301 Team Alpha Project
Version 2.0

Amy Lochner 14038600
Avinash Singh 14043778
Christiaan Nel 14029368
Christiaan Saaiman 12059138
Gerard van Wyk 14101263
Marc Antel 12026973
Themba Mbhele 14007950

https://github.com/AvinashSingh786/COS301-Alpha.git

University Of Pretoria

March 2016

Contents

1 Introduction

This document defines the Software Requirements Specification and Technology Neutral Process Design for COS 301 Team Alpha. The Computer Science Department has expressed a need for the creation of a system, which can allow researchers to keep track of publications which they are currently working on, or have already published.
The aim of this project is to follow a structured software development process in order to produce a product which provides the client with all the required functionality in an elegantly designed and built product. A collaborative and co-operative approach is required between all stakeholders who are involved in this project.
The information, specifications, and diagrams within this document are presented in order to provide testable requirements which correlate to the client's needs.

2 Vision

The client for this project, Ms. Vreda Pieterse who is a member of the Department of Computer Science at the University of Pretoria, has solicited us to develop a system. The purpose of this system is to record and oversee all publications of staff members or research groups, within the Department of Computer Science. The system will assist the Head of Department to track the progress on any papers which are in the process of being written; as well as determining whether or not a staff member is under performing thus allowing the Head of Department the opportunity to provide advice to these staff members.

3 Background

3.1 The client's problem

The client has solicited us to develop a system that has specific usability goals, as well as certain user-experience goals. Currently, the Department of Computer Science does not have a system with which to monitor the progress of staff member's publications, nor to keep track of how many publications an author is working on. Furthermore, it is necessary to determine which publications will be presented at different conferences, as well as the reporting capabilities, and the ability to remind users of deadlines. The system should provide all these requirements in a secure, flexible and intuitive manner.

3.2 Future business/research opportunities

It is desired that this system will encourage authors to collaborate with other authors on similar topics and to expand the users base knowledge of ongoing research projects.

4 Architecture Requirements

This section will be expanded on and developed further in the next phase of the project design and will be mentioned only briefly here.

4.1 Access Channel Requirements

As it is possible that the system may have concurrent users, it is favourable that there be interfaces such as a web application or website, as well as a mobile platform applications for the various different mobile devices through which the system can be accessed.

4.2 Quality Requirements

Specify and quantify each of the quality requirements which are relevant to the system

- Performance - How well the system can cope with extreme load.
- Security - Minimising the possibility of leaking information, maintaining data integrity, and avoiding session hijacking.
- Maintainability - Can the system be managed without downtime.
- Scalability - Can the system be used for large amount of users without it affecting performance.
- Efficiency - Can the system be optimized to produce faster and better results.
- Flexibility - Can the system be easily changed or modified.
- Reliability - Is the system able to cope with the load in order to provide constant access, ensure the system is always active and can provide all functionality.
- Integrability - Will the system be able to integrate with other technologies.
- User Friendly - Does a user easily understand how to use the system.
- Concurrency - Can multiple users perform actions at the same time.
- Low Cost, Reduced data usage - Is it suitable for users with low budget and capped Internet.
- Updatability - Can the system have version updates to introduce new features or functionality whilst maintaining old data.

4.3 Integration Requirements

This section specifies any integration requirements for any external systems. This may include

- The different Web protocols used.
- API - UML Interfaces.
- Google Calender and Email integration.
- Mobile Scalability and functionality Integration.
- Venue and Publication integration.

4.4 Architecture Constraints

This specifies any constraints, the client may require, to be placed on the system architecture. Such constraints may be:

- HTML (Hypertext Markup Language)
- AJAX (Asynchronous JavaScript and XML)
- JavaEE (Java Platform Enterprise Edition)
- JavaScript (Functionality to HTML)
- PHP (Server Side Scripting)
- MySQL (Database Manager)
- Android (Mobile Devices)
- IOS (Mobile Devices)
- Apache (Web Server)
- Linux/Windows (Operating System)

5 Software Architecture Documentation

This document defines

5.1 Architecture requirements

In this section extract the architectural requirements from the software requirements including

5.1.1 Architectural Scope

Architectural responsibilities that need to be addressed by the software architecture are as follows:

- Providing a persistence infrastructure - being a database stored on a dedicated server
- Providing a reporting infrastructure
- Providing an infrastructure for process execution
- Providing a backup infrastructure (e.g. making use of RAID)
- Providing a user interface through which the user can interact with the website/app
- Providing a means of Session management
- Providing a method of prevention against SQL injections
- Provide a log in system
- Provide a means with which to query the database in an interactive way
- Provide a means by which users can be notified about reminders

5.1.2 Quality requirements

- Performance

  HPerformance of a system refers to the behaviour of the system in terms of repsonse time and throughput. Changes and modifications to the system should be done in the most cost and time effective manner to provide a good user experience. One method of ensuring the performance is visible to the user is to make use of feedback tools to ensure the user is aware that the system is performing as expected or to notify the users should something have gone wrong.
- Security

  Minimising the possibility of leaking information, maintaining data integrity, and avoiding session hijacking. This is very important as personal information will be stored in the database(e.g. email). The system will also have different user types with different access permissions hence security will be needed to ensure that a specific user only has the capabilities afforded to them.

  We will achieve security by making use of 3 methods, namely: prevention, detection, and recovery. We will ensure that we have put means in place through which we may be able to detect any threats to the system for example validation any information that is required to be sent to the server. Preventing threats to the system can be enforced through limiting the access channels, making use of authentication, not allowing any external sources to be accessed through the system etc. Recovery will be achieved through cancelling requests, maintaining a working back up state.

- Maintainability

  This refers to how well the system can be modified to accommodate new functionality, access channels, fix bugs and improve the performance of the system. As it is likely that multiple people will be working on the system the code has to be easy to read and understand. We will achieve this by commenting our code to allow for a quick study of the code. We will employ a means by which to allow pluggability of our code to ensure that large amounts of code will not be needed to be changed/removed.

  The system should not only be maintainable in terms of issues with the system but it should also be maintainable in terms of usability. i.e user permissions should be allowed to be changed etc.

- Scalability

  Scalabilty refers to the system's ability to handle increased traffic or workload. This will be implemented by ensuring that users can make the same requests simultaneously.

  We will need to ensure that resources are managed wisely in order to avoid lost updates, uncommitted data and inconsistent retrievals.

- Reliability

  As the system is required to support a fairly large user base it should allow for effective and safe concurrent use of the system. The system should ensure that no users can perform tasks that their user type does not afford to them.

  As access channels are via a web browser/android app it is essential that these access channels always have access to the server and thus database and ensure that the connection is stable, reliable and safe at all times. The system should be maintained in order to ensure that the system can perform all the required tasks in an effective and efficient manner. We will achieve this through thorough unit testing, making use of concurrent resource locks and eliminate single points of failure.

- Auditability

A requirement of the system is that all actions or changes to the should be logged to enable users with the appropriate permissions to be able to view all actions or changes to the system, who they were made by and when. This is essential in ensuring that the system can be rolled back to a stable state should something undesirable happen.

We will implement this by have log file running at all times which makes use of timestamping to indicate when changes were made, the system should allow to rolled back to a stable state should it be deemed neccessary. ACID Properties will be applied to ensure that the database maintains and stable, reliable and current state.

- Integrability

  Will the system be able to integrate with other technologies.

- Usability

  Usability will be achieved through the implementation of an intuitive, easy to use, easy to understand and an aesthetically appealing interface through which the user will be able to perform all system functionality afforded to them through their user type.

  We will ensure that the user interface performs all tasks in the most efficient and direct means. The interface should not cause the user any irritation in the form of colour schemes, delayed functionality, over complication of tasks.

5.1.3 Integration and access channel requirement

As it is possible that the system may have concurrent users, it is favourable that there be interfaces such as a web application or website, as well as a mobile platform applications for the various different mobile devices through which the system can be accessed.

5.1.4 Architectural constraints

It is desired that this system will encourage authors to collaborate with other authors on similar topics and to expand the users base knowledge of ongoing research projects.

6 Architectural patterns or styles

7 Architectural tactics or strategies

In this section, we will discuss certain strategies that will be used to achieve the quality requirements stated in section 4.2 and were further expanded upon in section 5.1.2.

- Hashing and Salting Passwords

  To ensure that sensitive information of the users of the system is not leaked in the event of a database compromise, all passwords will not be stored in plain text. Instead, all passwords will be hashed with a strong one-way hashing algorithm, such as the sha256, so that in the event of a database compromise, it will be near impossible to compute the actual passwords from the hashes. To make the hashes more effective, we will add random data, known as salt, to the passwords before hashing them. This will make it extremely difficult for intruders to use pre-hashed tables to perform a look up to find a matching password.

- Prepared Statements

  To prevent the illegal extraction of data and/or the destruction of data from the database through SQL injection and blind SQL injection attacks, we will make use of prepared statements. Prepared statements also have other benefits other than security. Even though prepared statements execute a statement multiple time, they reduce parsing time as the preparation of the query is only done once. This will improve the efficiency and performance of the system. Also, because prepared statements bind values to parameters, the bandwidth on the server is minimized because you only need to send the parameters to the server and not the whole query. This also improves the performance of the system.

- Caching

  Because the system relies heavenly on databases, there will be a significant number of access attempts to the the databases. This has the potential of creating a bottleneck that gets worse as the number of concurrent users increases. This means that the system will scale poorly and the performance of the system will decrease drastically. The best way to combat this is to limit accesses to the databases were possible. To accomplish this, we will make use of a distributed cache system. A distributed cache will particularly be effective in read operations and will provide large performance gains as it reduces the processing times of applications and it limits database accesses. This will improve the scalability of the system drastically and thus will accommodate a large number of concurrent users.

- Database Normalization

  To make the system flexible, we will create the database tables and link them to one another were it is appropriate to do so according to rules that protect the data and makes it flexible. This process is known as normalization. Normalization eliminates redundancies and inconsistent dependencies. This makes it simple to makes changes to the system and also makes it easy to add new components to the system.

8 Use of reference architectures and frameworks

9 Access and integration channels

Specify and quantify each of the quality requirements which are relevant to the system

- The different Web protocols used.
- API - UML Interfaces.
- Google Calender and Email integration.
- Mobile Scalability and functionality Integration.
- Venue and Publication integration.

10 Technologies

This specifies any constraints, the client may require, to be placed on the system architecture. Such constraints may be:

- HTML (Hypertext Markup Language)
- AJAX (Asynchronous JavaScript and XML)
- JavaEE (Java Platform Enterprise Edition)
- JavaScript (Functionality to HTML)
- PHP (Server Side Scripting)
- MySQL (Database Manager)
- Android (Mobile Devices)
- IOS (Mobile Devices)
- Apache (Web Server)
- Linux/Windows (Operating System)

11 Functional Requirements and Application Design

This section discusses the application functionality and service contracts required by the users.

11.1 Use Case Prioritization

The Use Case Prioritisation is specified for each use case in the next section.

11.2 Use Case/Services Contracts

Use Case Prioritisation and Service Contracts are described below

11.2.1 User Login

Description: A user is required to log into the system before the user can make use of any functionality.
Prioritisation: Critical

Pre-conditions

- A user must have a connection to the server.
- A user must be registered as a user by a person with HOD or Admin permission.
- The user must enter the correct information in order for the authentication to be successful.

Post-conditions

- The user has specific access to the server on which all data is stored, i.e add/edit authors and search for authors
- The user is able to use all the user functionality provided by the system
- The user may log out of the system when they wish to.

Figure 1: Service Contract: User Login

11.2.2 User Registration

Description: In order to be able to log into the system and make use of the functionality provided by the system the user must be registered on the system.
Prioritisation: Critical

Pre-conditions

- The user must be part of the staff of the Computer Science Department of the University of Pretoria.
- The administrator or HOD of the system must have registered the user on the system.
- The user must decide on their login credentials.

Post-conditions

- The user's login information is securely stored in the system database.
- The user can log into the system.

Figure 2: Service Contract: User Registration

11.2.3 Update User

Description: Personal details as well as log in credentials can be changed if and when necessary.
Prioritisation: Important

Pre-conditions

- The administrator or HOD must be logged into the system.
- The user to be updated must already exist in the system.
- The administrator or HOD must have the new information.

Post-conditions

- The user's information is updated on the system's database.

Figure 3: Service Contract: Update User

11.2.4 Remove User

Description: Should a user no longer belong to the Department of Computer Science the person should be removed from the system.
Prioritisation: Important

Pre-conditions

- The person should be a user on the system.
- The person should no longer belong to the staff of the Department of Computer Science.
- The administrator or HOD must be the logged in.

Post-conditions

- The person is removed from the system's database.
- The person can no longer gain access to the system.
- The person history still remains in the log file.

Figure 4: Service Contract: Remove User

11.2.5 Create Publication

Description: Adding a paper to the system
Prioritisation: Critical

Pre-conditions

- The user(may be the administrator or HOD) should be logged into the system.
- The user, if not the administrator or HOD, must be a contributor to the paper.
- All authors who contributed to the paper should be available on the system.

Post-conditions

- The paper is added to the system

Figure 5: Service Contract: Create Publication

11.2.6 Update Publication

Description: Allows a user/administrator/HOD to change a publication's meta-data
Prioritisation: Critical

Pre-conditions

- The user(may be the administrator or HOD) should be logged into the system.
- The user, if not the administrator or HOD, must be a contributor to the paper.
- The paper must already be in the system.

Post-conditions

- The paper's meta-data is updated

Figure 6: Service Contract: Update Publication

11.2.7 View Publication

Description: Allows a user/administrator/HOD to view the meta-data of a paper that lies within their permissions
Prioritisation: Important

Pre-conditions

- The user(may be the administrator or HOD) should be logged into the system.
- The user - if not the administrator, HOD or research leader - must be a contributor to the paper.
- The paper must already be in the system.

Post-conditions

- The paper's meta-data is displayed to the user

Figure 7: Service Contract: View Publication

11.2.8 Add Publication Type

Description: Allows the administrator or HOD to add a publication type
Prioritisation: Important

Pre-conditions

- The user(must be the administrator or HOD) should be logged into the system.
- The publication type to add must not already be in the system.

Post-conditions

- The new publication type is added to the system.

Figure 8: Service Contract: Add Publication

11.2.9 Update Publication Type

Description: Allows the administrator or HOD to update a publication type's details
Prioritisation: Important

Pre-conditions

- The user(must be the administrator or HOD) should be logged into the system.
- The user - if not the administrator, HOD or research leader - must be a contributor to the paper.
- The publication type to be updated must already be in the system.

Post-conditions

- The selected publication type will be updated with the new information.

Figure 9: Service Contract: Update Publication

11.2.10 Add Reminder

Description: Allows the user to create a reminder for the deadline of the publication.
Prioritisation: Nice-To-Have.

Pre-conditions

- The user(can be the HOD) should be logged into the system.
- The user must be a contributor to the paper.
- The publication must already be in the system.

Post-conditions

- A reminder via Mail or Calendar will be set with the selected publication and deadline.

Figure 10: Service Contract: Add Reminder

11.2.11 Create Research Group

Description: Allows the administrator or HOD to create a new Research Group
Prioritisation: Critical

Pre-conditions

- The user(must be the administrator or HOD) should be logged into the system.
- The Research Group you wish to create must not already be in the system.

Post-conditions

- The Research Group will be created and stored in the database.
- Users will be able to join the Research Group.
- Research Leader will be able to see all Publications made by the Research Group's members.

Figure 11: Service Contract: Create Research Group

11.2.12 View Research Group

Description: View the details of a Research Group
Prioritisation: Nice-To-Have

Pre-conditions

- The user(must be the administrator or HOD) should be logged into the system.
- The Research Group must already exist in the database.

Post-conditions

- The user will be able to view the details of the Research Group.

Figure 12: Service Contract: View Research Group

11.2.13 Update Research Group

Description: Allows the administrator or HOD to update an existing Research Group
Prioritisation: Important

Pre-conditions

- The user(must be the administrator or HOD) should be logged into the system.
- The Research Group must already exist in the database.

Post-conditions

- The user will be able to edit and save the new information entered.

Figure 13: Service Contract: Update Research Group

11.2.14 Remove Research Group

Description: Allows the administrator or HOD to remove an existing Research Group
Prioritisation: Important

Pre-conditions

- The user(must be the administrator or HOD) should be logged into the system.
- The Research Group must already exist in the database.
- The Research Group must have no members in it and no Research Leader.

Post-conditions

- The user will be able to remove the Research Group from the database.

Figure 14: Service Contract: Remove Research Group

11.2.15 Create Author

Description: Allows the user to create a new Author
Prioritisation: Critical

Pre-conditions

- The user should be logged into the system.
- The Author must not already exist in the database.

Post-conditions

- The user will be able to create a new Author and save him/her in the database.

Figure 15: Service Contract: Create Author

11.2.16 Update Author

Description: Allows the user to update an existing Author
Prioritisation: Important

Pre-conditions

- The user should be logged into the system.
- The Author must already exist in the database.

Post-conditions

- The user will be able to edit and save the new information entered.

Figure 16: Service Contract: Update Author

11.2.17 Generate Report

Description: Allows the user generate a report based on certain data
Prioritisation: Nice to have

Pre-conditions

- The user should be logged into the system.
- There should be existing data in the system's database

Post-conditions

- The system will generate a report based on certain criteria and make it visible to the user

Figure 17: Service Contract: Generate Report

11.3 Required Functionality

11.3.1 User-Research System interaction

Description: The type of user indicates what privileges that user has in the Research system

Normal-user

- A normal user may log in to the system if registered on the system.
- A normal user may add publications to the system.
- A normal user must be an author of a publication should they want to add it to the system.
- A normal user may add authors to a publication.
- A normal user may change authors in a publication.
- A normal user may add a publication to a conference.
- A normal user may only view their own publications.

Head of Department

- The head of department may log in to the system.
- The head of department may add/edit/remove a user.
- The head of department may add/edit/remove an author.
- The head of department may add publication/edit a publication.
- The head of department may be an author on a publication.
- The head of department may add authors to a publication.
- The head of department may add/remove publications for conferences.
- The head of department may view all publications on the system.

Admin

- Admin users may log in to the system.
- Admin users may add/remove/edit users.
- Admin users may add/edit publications.
- Admin users may not be an author to any publication on the system.
- Admin users may add authors to a publication.
- Admin users may change authors to a publication.
- Admin users may add/remove publications to conferences.

- Admin users may view all publications on the system.

../Assignment1/Overview.jpg

Figure 18: Functional Requirements: Overview of Research System

../Assignment1/AccessNormaluser.jpg

Figure 19: Functional Requirements: Normal user access privileges

../Assignment1/AccessHODAdmin.jpg

Figure 20: Functional Requirements: Superuser(HOD and admin) access privileges

11.4 Process Specification

This section conatins UML activity diagrams that illustrate the sequences that will be followed for various use case scenarios.

`../Assignment1/AddUser.jpg`

Figure 21: Process Specification: Adding A User

../Assignment1/removeUser.jpg

Figure 22: Process Specification: Removing A User

`../Assignment1/UpdateUser.jpg`

Figure 23: Process Specification: Update A User

`../Assignment1/AddPublication.jpg`

Figure 24: Process Specification: Adding A Publication

../Assignment1/UpdatePublication.jpg

Figure 25: Process Specification: Update A Publication

`../Assignment1/ViewPublication.jpg`

Figure 26: Process Specification: View A Publication

11.5 Domain Model

../Assignment1/DomainModel.jpg

Figure 27: Domain Model of Research System

12 Open Issues

This section deals with issues that still need to be clarified, specified, assumed or have been discovered to include inconsistencies in the requirements, comprising of the following issues:

- Will a server be provided?
- Should the HOD be a separate entity or fall within the Admin entity?
- To where does the publication go to be reviewed?
- Should the reminder system be via Mail or Calendar notification?
- It is assumed that the Venue will have a deadline attached.