UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Software Requirements Specification
and
Technology Neutral Process Design

Name of project

Version: 1.0

Maree, Armand
12017800

Watt, Brenton
14032644

Meyers, Charl
14024633

Tom, Elton
13325095

Tswene, Kamogelo
12163555

Molefe, Keletso
14222583

Spazzoli, Lorenzo
13304862

March 2, 2016

Contents

1 Introduction

In this document it would be specified how a system would be developed for the department of computer science at the University of Pretoria in order for them to replace their current, not so efficient Microsoft Office Excel spreadsheet, system with a more concurrent and reliable option.

2 Vision

The client requires a system that would allow them to retrieve and submit meta-data about academic papers published by the department of computer science at the University of Pretoria. Certain users would be able to create new papers and the project leader would then be able to control certain properties of the project, like the current progress of the paper. It should also have a web interface where changes can be made and also an Android app.

3 Background

What lead to the project is the fact that that the client is facing a problem with the current system they are using. The spreadsheet which does fulfill its task of storing details about user's publications, does not have a user friendly interface, and does not allow researchers to see any changes made to the system immediately, it is also not a concurrent system and is deemed less reliable. The client wants a system that would improve the manner the users would be able to view their submissions and be able to modify the system to their liking, with changes immediately visible to researchers. Another aspect which lead to the development of a new system which would simplify the collaboration between the University of Pretoria users and users from other universities, is that instead of having a system which is local; a web interface could be used which is more accessible. The lack of ability to easily access previously entered user information data such as oneâĂŹs username or contact details also was a factor that lead to this project, as it would be more convenient and efficient to have to have an interface that would for example have a drop down list of all the user information data on the system.

4 Architecture Requirements

4.1 Access channel requirements

The system requires 2 interfaces:

- Web interface
- Android app (Could also be a mobile website)

4.2 Quality requirements

The following quality aspects needs to be addressed:

- **Scalability:** Less a 100 users will use the system.
- **Security:** Passwords will be hashed with the SHA512 hashing algorithm.
- **Reliability:** An automatic partial dump backup will be done everyday at 03:00 and each back up will be kept for a week. On top of this a full dump will be done every 3 days and will be kept for 2 weeks. Alternatively the backup will be done by an external program.
- **Audibility:** Every change should be logged and the person responsible for that log will be recorded. This will allow the administrators to track which users change what.
- **Maintainability:** Data that is deleted will only be moved to another database that might be a little slower. This will help alleviate some pressure off the main database.
- **Cost:** The total hours is estimated to be 120 at a cost of R500 per hour.

4.3 Integration requirements

The system should allow universities to connect to each other using an API. This would allow them to collaborate on projects and also to track the papers that are being written. The Android app should also be able to connect to the server via the API.

The protocol that will be used to transfer the data between the server and the clients will be the Hyper Text Transfer Protocol Secure (HTTPS). And the third parties that integrate with the system will access it either through the web interface, the mobile app or the provided API.

Some quality requirements that have to be considered are:

- **Security:** The data that is sent over the internet should be encrypted and it has been decided to use the Secure Socket Layer (SSL) encryption algorithm.
- **Reliability:** The reliability of the transfer of the data is dependent on the reliability of the internet connection.

4.4 Architecture constraints

There are currently no architecture constraints that the client mentioned.

5 Functional requirements and application design

5.1 Use case prioritization

Critical:

- There must be some web hosting server in order to be able to host a web interface, this interface will be the main access channel to the system as it makes it easier to access the system on a computer or non-Android device. If there is no web-server then there will be no web interface that can be accessed from research groups all over the country. This defeats the whole purpose of the system.
- The web interface needs to be accessible from the Internet preferably so that other research groups outside the University are able to access the system.
- The system needs to be able to save data in some way. If there is no possible way to store data then the system might as well not exist, because the purpose of the system is to store meta data about research papers.

Important:

- Users must be able to download a bibliography of papers released in a selected period.
- An Android app must be available that can access the system to make it easier to access the system on the go.
- Users must create their user profile manually and will not be able to link the system with a web service in order to create their profile.
- No item should ever be deleted in the system. Everything is kept. Old unaccessed data may be archived but never deleted. There needs to be some sort of recovery or backup system that enables a user to restore accidently deleted data.
- The system must be able to handle a max of at least 50 concurrent users as there is a possiblility that at least 50 users may access the system all at once. There will also be an estimate of at least a 1000 users. The system must be able to handle that amount of users.

Nice-to-have:

- The system must be lightweight.
- The system must show the completion of a paper in percentage that a user specifies after editing the paper.

- The system may link with Google Calender to synchronise due dates and alerts for the paper a user is busy working on.
- The web interface can be designed to work on mobile phones for those who want to access the system on the go but do not have an Android device.

5.2 Use case/Services contracts

- **Use Case:** Add a paper.

  Primary Actor: Registered author or HOD.

  Brief: Only an author or head of department can add new papers.

  Postconditions:

  - The article is saved to the database.
  - All authors are notified of their new paper.
  - The changes are logged.
  - Paper is displayed.

  Triggers: The user invokes the Add New Paper request.

  Basic flow:

  1. Present the actor with a new paper form.
  2. Actor fill in the form and clicks submit.
  3. The data is sent to the server and added to the database and logs the event.
  4. Actor is presented with a success message.
  5. Paper is presented to the actor.
  6. Authors are notified.

- **Use Case:** Editing a paper.

  Primary Actor: Author of paper or HOD.

  Brief: Only an author involved with the paper or head of department can edit the paper.

  Preconditions:

  - Author is presented with the paper in edit mode.

  Postconditions:

  - The article is saved to the database.
  - All authors are notified of the changes.

- The changes are logged.
- Paper is displayed.

Triggers: The user invokes the Edit Paper request.

Basic flow:

1. Present the actor with the paper in edit mode.
2. Actor makes all desired changes and clicks submit.
3. The data is sent to the server and added to the database and logs the event.
4. Actor is presented with a success message.
5. Paper is presented to the actor.
6. Authors are notified.

- **Use Case:** Viewing a paper.

  Primary Actor: Author of paper or HOD.

  Brief: Only an author involved with the paper or head of department can view the paper.

  Preconditions:

  - Author is presented with the paper in read-only mode.

  Triggers: The user invokes the View Paper request.

  Basic flow:

  1. Present the actor with the paper in read-only mode.
  2. When the actor is done viewing they simply navigate away. No changes or events has to be logged.

Request and Results Data Structures: See figure 1.

5.3 Required functionality

- Add a paper Use case: See figure 2
- Edit a paper Use case: See figure 3
- View a paper Use case: See figure 4

5.4 Process specifications

- Activity diagram to add a paper: See figure 5
- Activity diagram to edit a paper: See figure 6
- Activity diagram to Login: See figure 7
- Activity diagram to Request document details: See figure 8

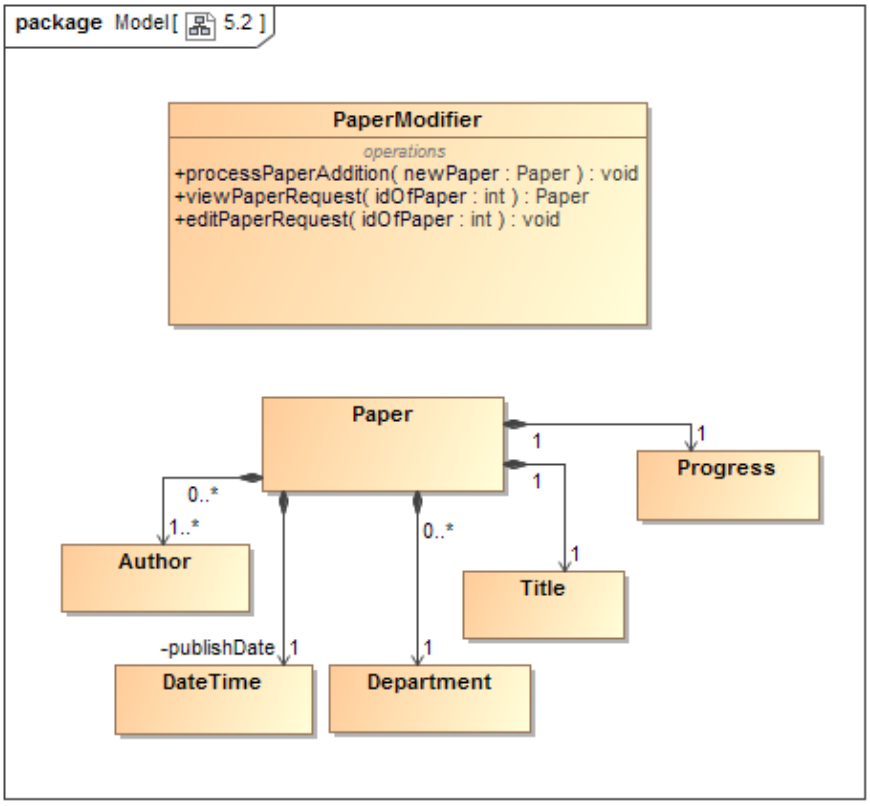

Figure 1: UML diagram for Request and Results Data Structure.

5.5 Domain Model

6 Open Issues

- To be determined

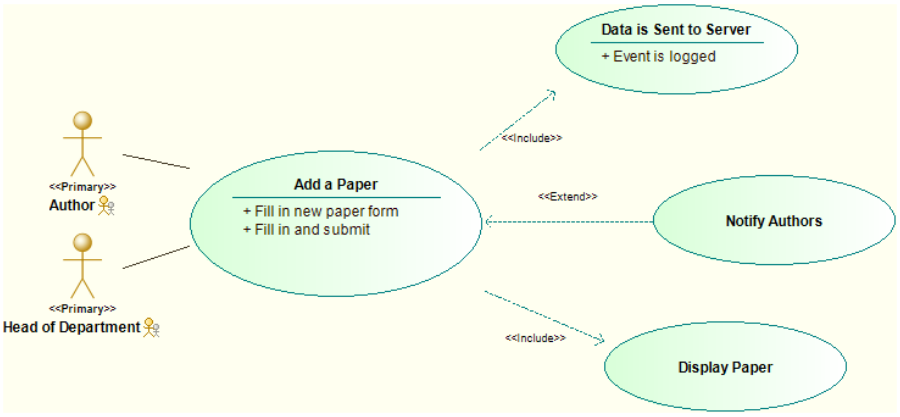

Figure 2: Use case diagram for adding a paper

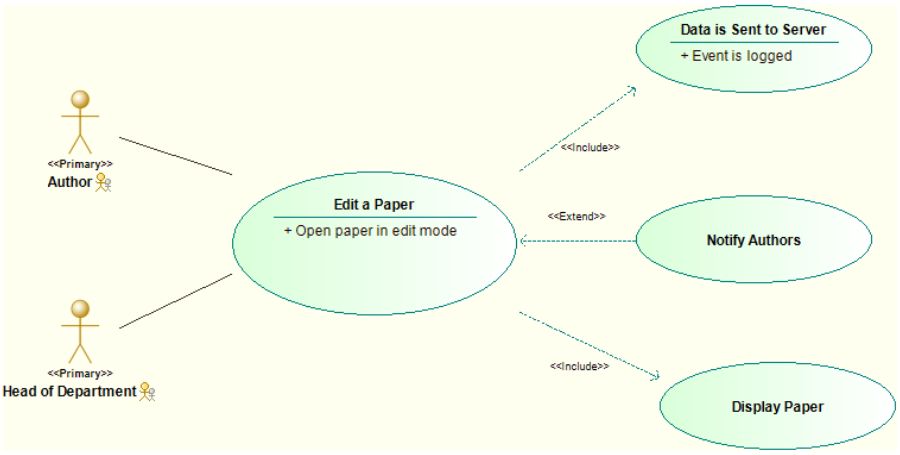

Figure 3: Use case diagram for editing a paper

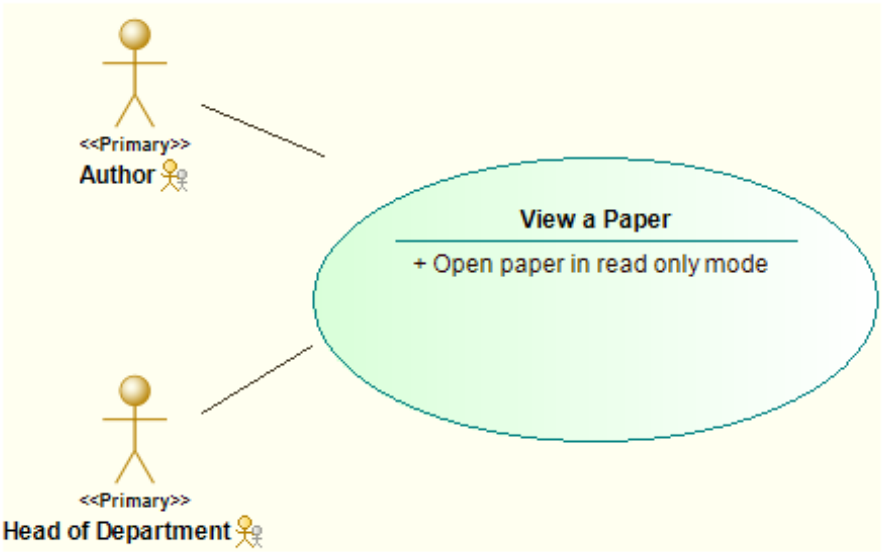

Figure 4: Use case diagram for viewing a paper

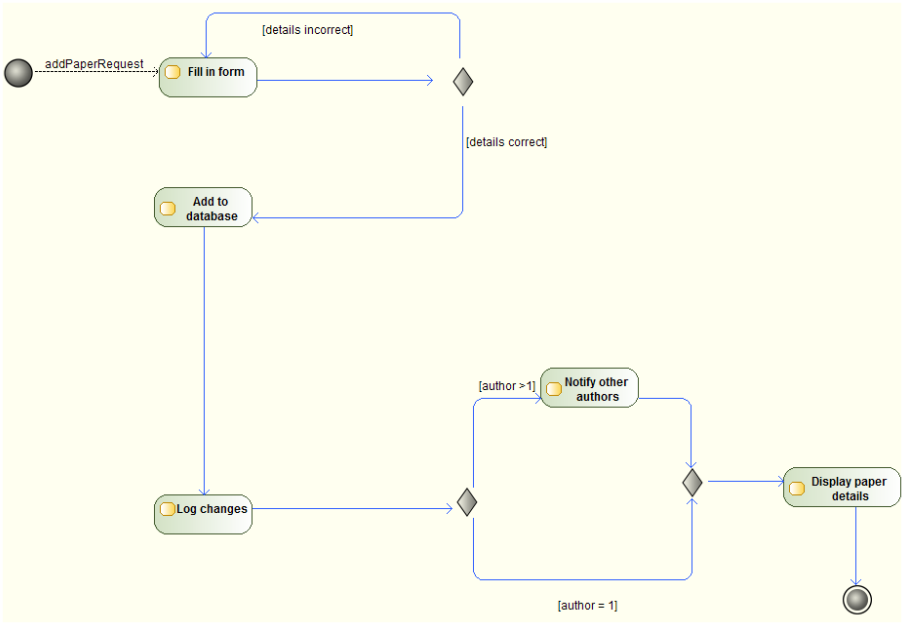

Figure 5: Adding a paper to the database

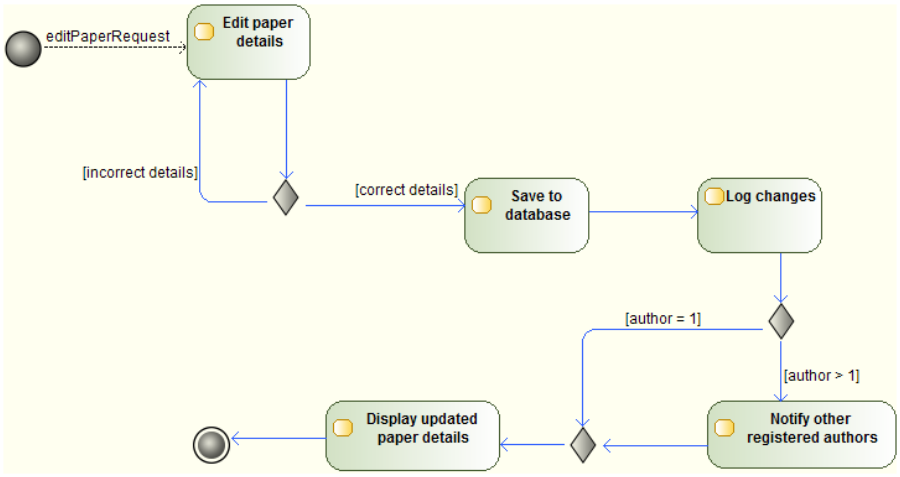

Figure 6: Editing a paper

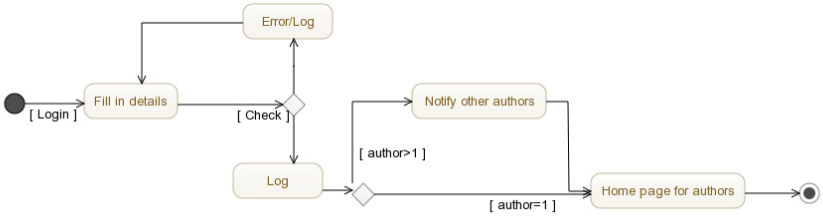

Figure 7: Login to system

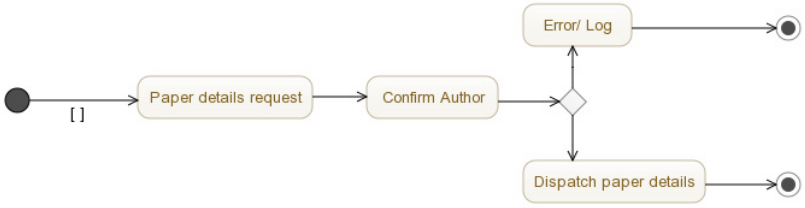

Figure 8: Request paper details