UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Software Requirements Specification and Technology Neutral Process Design

Members:
Matthew Botha 14214742
Gershom Maluleke 13229908
Nathan Dunkley 14145759
Aiden Malan
Sello Thosago
Nsovo Baloyi 12163262
Matheu Botha 14284104

Team India

March 11, 2016
v0.1

1 Introduction

The purpose of this document is to outline the core functionality of the new computer science publication system in a technology neutral design specification. This design is not necessarily a depiction of what the final system will look like. During the development of the system it is possible that new functionalities are added, or old ones removed. This is merely the first iteration of the development cycle. Which can be revised in the future. As well as iterativly enhanced and detailed as the need arises.

2 Vision

This projects aim is to provide a system to facilitate the administration of postgraduate publications, including but not limited to journal articles and conference papers. It should be able to store information on Users, Authors and Publications as well as employ a authentication system to ensure only specified parties within the system have access to operations on the stored information. The purpose of keeping track of the publications is, in part, to keep track of the DoE and UP weighted units earned by Users and/or authors. The reason for this is that these users/authors are required to publish a certain amount of units each year in order to determine pay, research funding, and maintaining their research position.

3 Background

Currently the publications are kept track of with a complicated excel spreadsheet. The purpose of the spreadsheet is to keep track of the status of publications in order to generate a report detailing the units earned by a researcher. There are many drawbacks to this system. It is tedious to maintain the spreedsheet. Data is spread over multiple sheets. The larger the database gets the slower it preforms. The information is displayed in a raw textual format which makes it a daunting task to sit through and analyse. It is difficult to add new functionality and change or enhance existing functionality. When pulling in data from multiple sheets it becomes difficult to debug the Lookups.

4 Software Architecture Documentation

4.1 Architecture requirements

4.1.1 Architectural scope

4.1.2 Quality requirements

- Reliability - The reports generated from the data provided to the system should be

- Scalability - It is estimated that there will be approximatly one hundred users. The system should be able to handle each user being active at the same time even though it is unlikly.
- Cost - Generating a report for all data in the system will be the worst case scenario in this system and thus have a high cost. The system should however be able to produce subsets of the report at a relativly low cost. For example a report for all researchers under a specific research group.
- Security - There should be strict access right applied to each of the subsystems and their components. This means that if a user, by definition of the buisness rules, is not allowed to access a certain functionality, then they will be denied access.
- Auditability - Actions performed on the system should be logged in a manner that would allow one to trace back the history of actions performed on the system.

4.1.3 Integration and access channel requirements

The system will operate in a standalone fashion. In the future, however, it may require intergration with google calender. This service should be accessable via web based clients. i.e. web based clients,mobile browsers. As well as an Android application.

4.1.4 Architectural constraints

Client has not provided any architecture constraints.

4.2 Architectural patterns or styles

4.3 Architectural tactics or strategies

4.4 Use of reference architectures and frameworks

The system will incorporate the Ruby on Rails framework. It is easy to use this framework with front end frameworks. Ruby on Rails will be coupled with jQuery. The system will use jQuery because it is extensible and supported by most, if not all, browsers. Node.js will be used because it is a runtime environment which runs and handles all the server-side logic.

The system will make use of the Model-view-controller architecture. This architecture is ideal because the software application will be separated into three interconnected parts, which means each module can function as separate entities.

4.5 Access and integration channels

4.5.1 Access Channels

The user will be able to access the system, provided they are logged in, via the web either with a PC or a smartphone. It is important that the system has a user friendly interface, especially for mobile devices, to make it as easy to use as possible. The user will also be able to use the system via an Android application if they so choose.

4.5.2 Integration Channels

The system will integrate with external systems via a MySQL database.

4.6 Technologies

5 Functional Requirements and Application Design

5.1 Use case Prioritisation

5.1.1 User Subsystem

Register User: Critical
- Update User: Important
- Login User: Critical
- Logout User: Critical
- Add to Publication: Important

5.1.2 Publication Subsystem

Create Publication: Important
View Publication: Nice-to-Have
Update Publication: Important
Remove Publication: Important
Add Author: Important
Remove Author: Important

5.1.3 Conference Subsystem

Add Conference: important
Update Conference: important
Delete Conference: important

5.1.4 Journal Subsystem

Add Journal: important
Update Journal: important
Delete Journal: important

5.1.5 Thesis Subsystem

Add Thesis: important
Update Thesis: important
Delete Thesis: important

5.1.6 Research and Authors Subsystem

Create research paper: important
Update research paper: important
Remove research paper: important
View research paper: nice to have

5.1.7 Report Subsystem

Log In to System: critical
View Available Reports: important
Select Filters to Apply to Reports: important
Select Output Type of Report: nice-to-have
Show Report: important

5.2 Use case/Services contracts

5.2.1 User Subsystem

Pre-conditions:

Register user: The user can not be registered to the system already and the user registering a new user has to be admin.

Update user: The user has to be logged in first before they can update their profile and they can only update their own profile.

Login user: The user has to be registered to the system first.

Logout user: The user has to be logged in first.

Add to publication: The user has to be logged in first and the publication has to exist in the system

Post-conditions:

Register user: The user is now registered and added to the database.

Update user: The user's profile is updated and the changes are saved to the database.

Login user: The user is logged into the system.

Logout user: The user is logged out of the system.

Add to publication: The user's name is now added to the author's of the selected publication.

5.2.2 Publication Subsystem

Use Cases:

3.1 Create Publication:

Pre-conditions: The user must be logged into the system

Post-conditions: The publication is created

3.2 View Publication:

Pre-conditions: The user must be logged onto the system and the publication must be created before hand

Post-conditions: The user will be able to view and edit(Add authors etc.) to the publication.

3.3 Update Publication:

Pre-conditions: The user must be logged onto the system and the publication must be created before hand

Post-conditions: The publication is updated

3.4 Remove Publication:

Pre-conditions: The user must be logged onto the system, the publication must be created before hand and the user must confirm the removal of the publication

Post-conditions: The publication will be removed

3.5 Add Author:

Pre-conditions: The user must be logged onto the system and the publication must be created before hand

Post-conditions:A new author is added to the publication

3.6 Remove Author:

Pre-conditions: The user must be logged onto the system, the publication must be created before hand and the user must confirm the removal of the selected author

Post-conditions: The author is removed from the publication

5.2.3 Conference Subsystem

Add Conference:
Pre-conditions: The user must be logged in.
Post-conditions: The conference is added.
Update Conference:
Pre-conditions: The user must be logged in and a the conference must be added prior to this use case.
Post-conditions: The conference is updated.
Delete Conference:
Pre-conditions: The user must be logged in and a the conference must be added prior to this use case.
Post-conditions: The conference is deleted.

5.2.4 Journal Subsystem

Add Journal:
Pre-conditions: The user must be logged in.
Post-conditions: The journal is added.
Update Journal:
Pre-conditions: The user must be logged in and a the journal must be added prior to this use case.
Post-conditions: The journal is updated.
Delete Journal:
Pre-conditions: The user must be logged in and a the journal must be added prior to this use case.
Post-conditions: The journal is deleted.

5.2.5 Thesis Subsystem

Add Thesis:
Pre-conditions: The user must be logged in.
Post-conditions: The thesis is added.
Update Thesis:
Pre-conditions: The user must be logged in and a the thesis must be added prior to this use case.
Post-conditions: The thesis is updated.
Delete Thesis:
Pre-conditions: The user must be logged in and a the thesis must be added prior to this use case.
Post-conditions: The thesis is deleted.

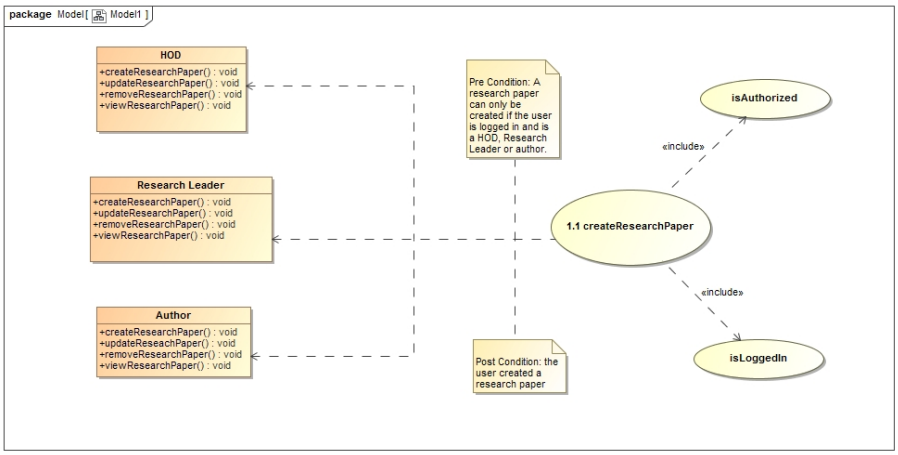


Figure 1: Pre and Post conditions create function

5.2.6 Researchers and Authors Subsystem

3.1 Create Research Paper:

Pre-conditions: The user must be logged into the system. The user must also be either HOD, research leader or author.

Post-conditions: The research paper is created.

3.2 Update Research Paper:

Pre-conditions: The user must be logged onto the system and the research paper must exist.

Post-conditions: The user will be able to view and edit(Add authors etc.) to the research paper.

3.3 Remove Research Paper:

Pre-conditions: The user must be logged onto the system and the research paper must exist.

Post-conditions: The the research paper is deleted.

3.4 View Research Paper:

Pre-conditions: The user must be logged onto the system, the research paper must exist.

Post-conditions: The publication can be viewed

5.2.7 Conference, Journal and Thesis Subsystem Request and Results Data Structures

5.2.8 Report Subsystem

Pre-Conditions: User must be logged in to be able to generate a report.
Post-Conditions:

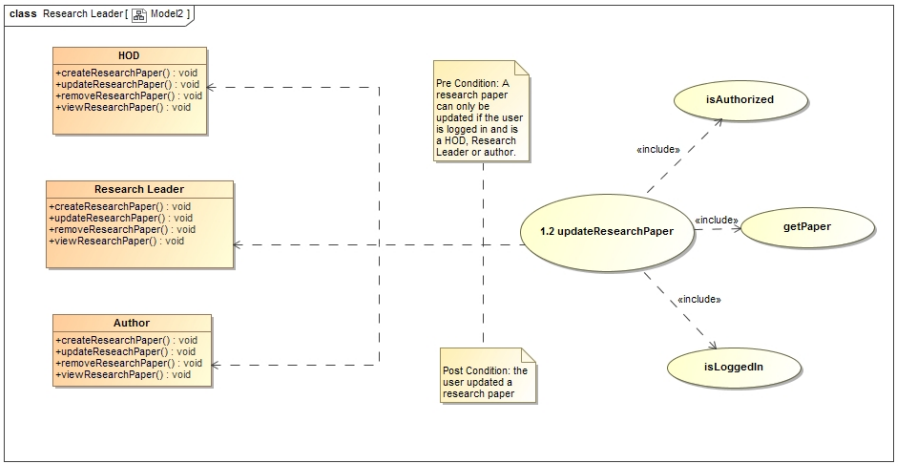


Figure 2: Pre and Post conditions update function

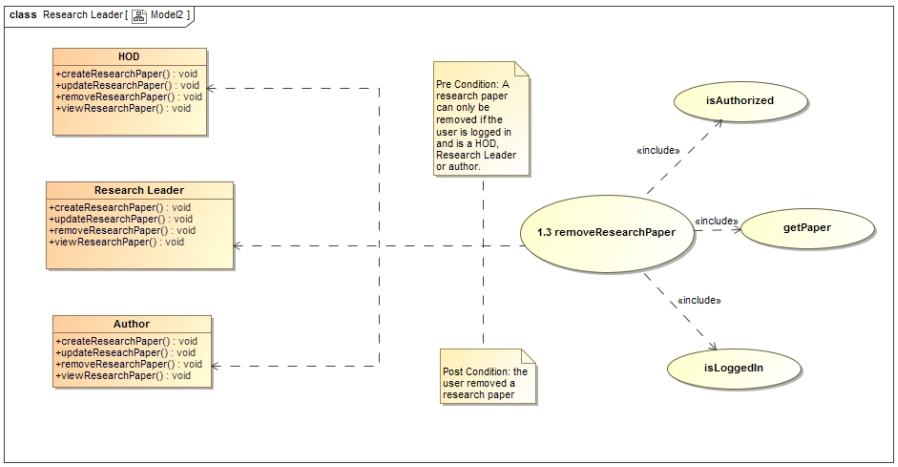


Figure 3: Pre and Post conditions remove function

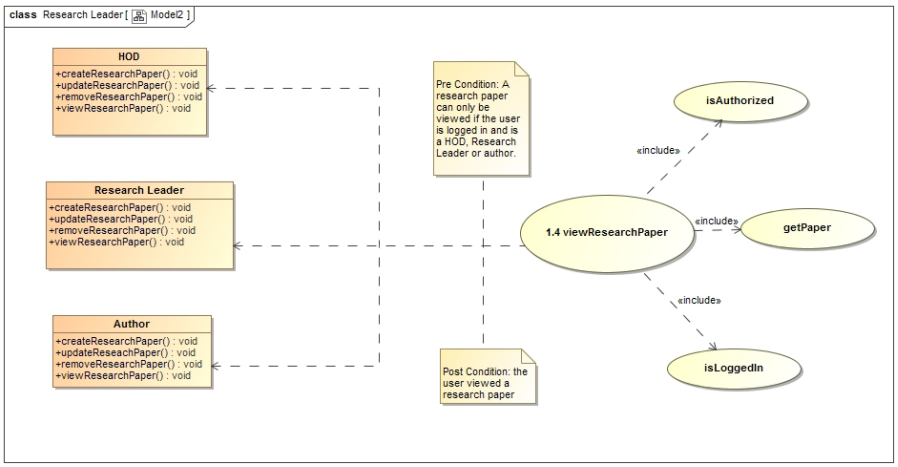
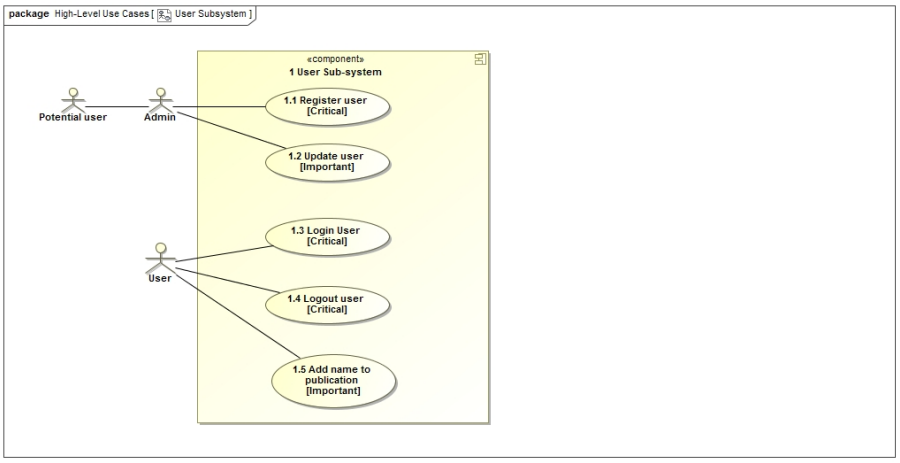


Figure 4: Pre and Post conditions view function

- A simple but comprehensive report must be generated in the format chosen by the user.
- The report must only include papers that the current user is associated with.

5.3 Required Functionality

5.3.1 User Subsystem Use Case



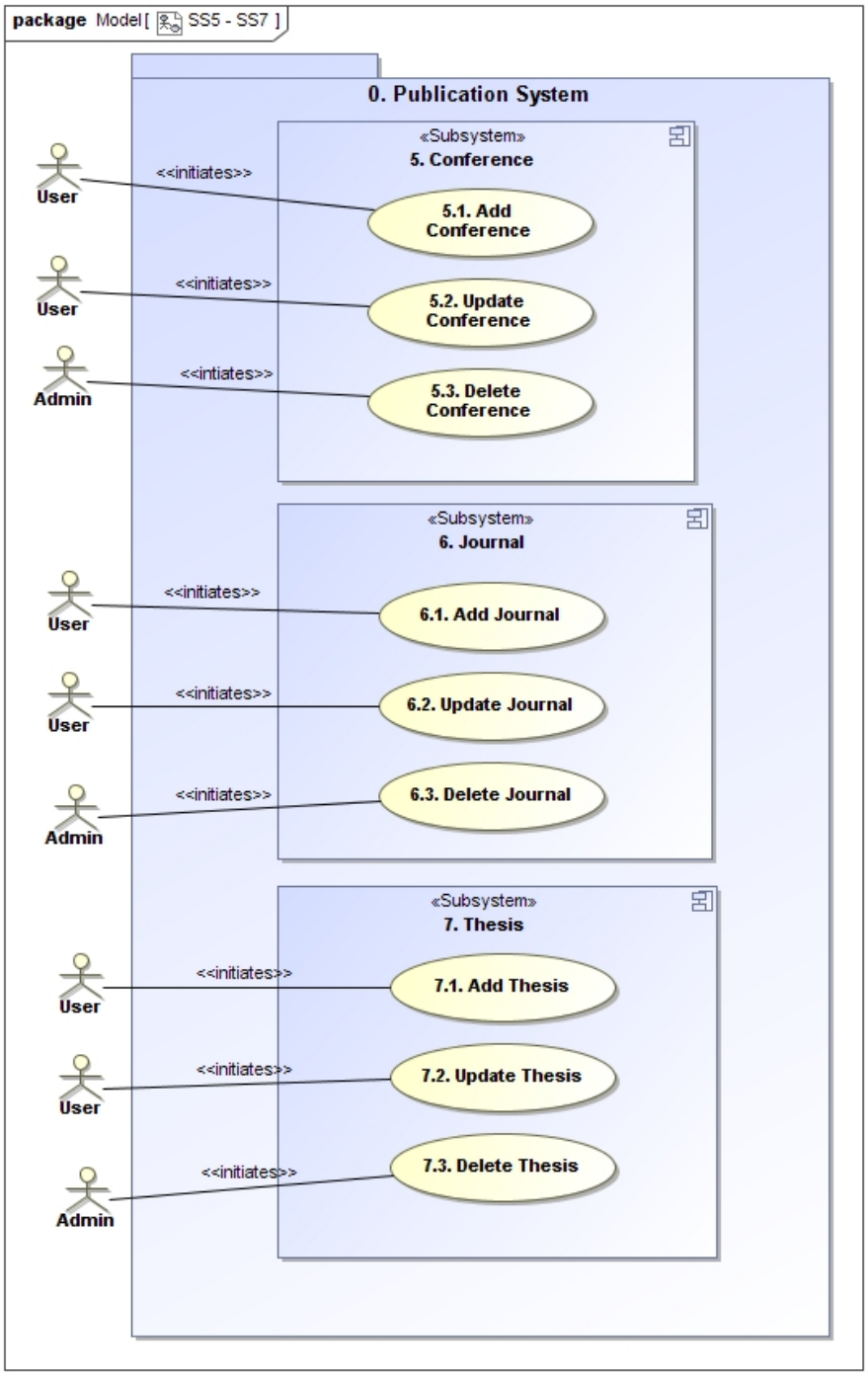
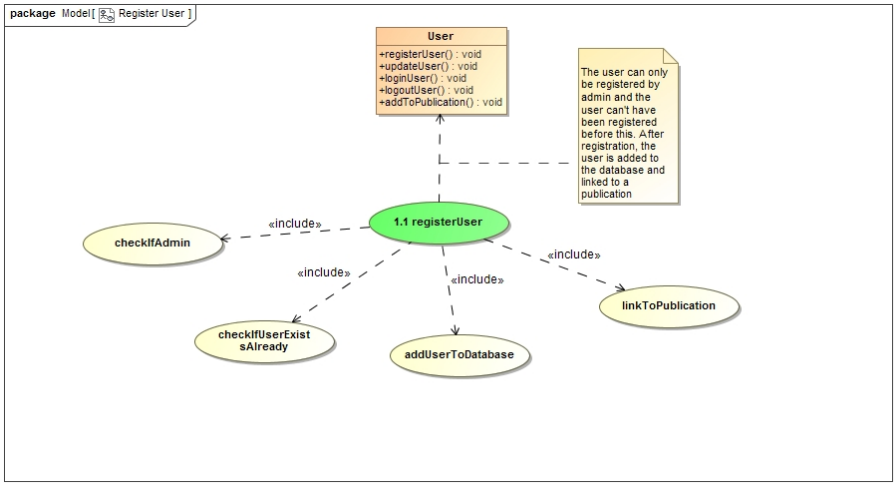
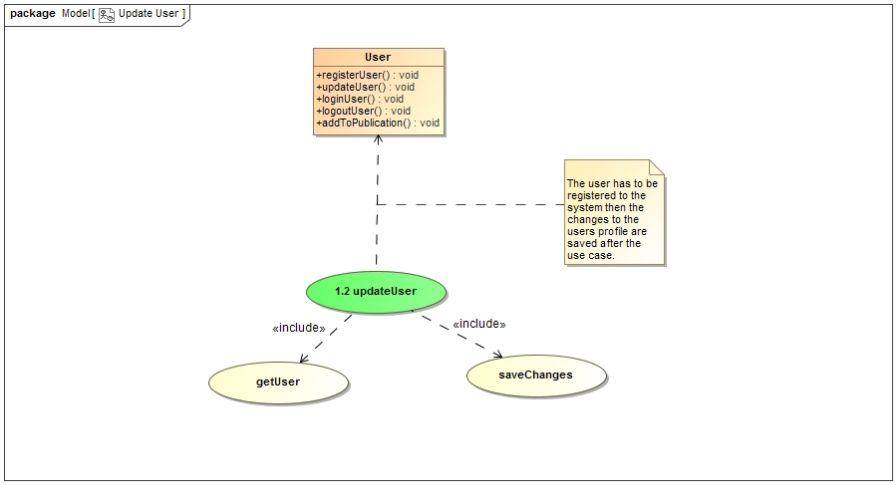
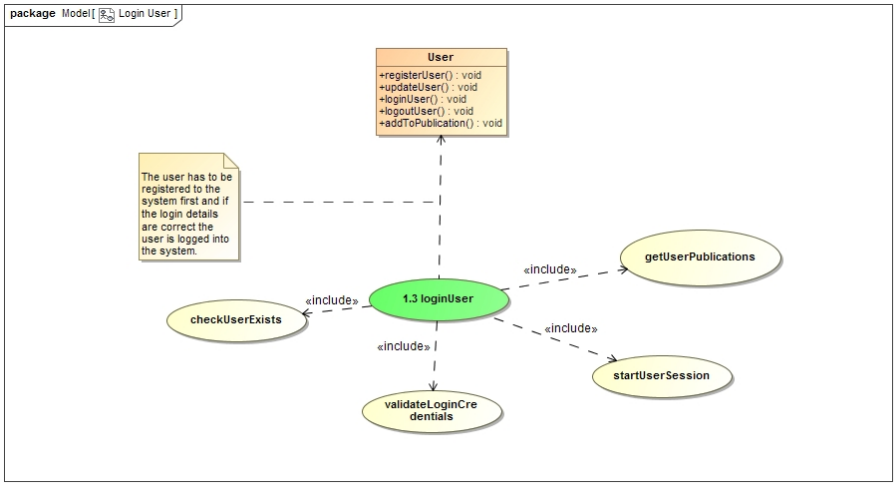
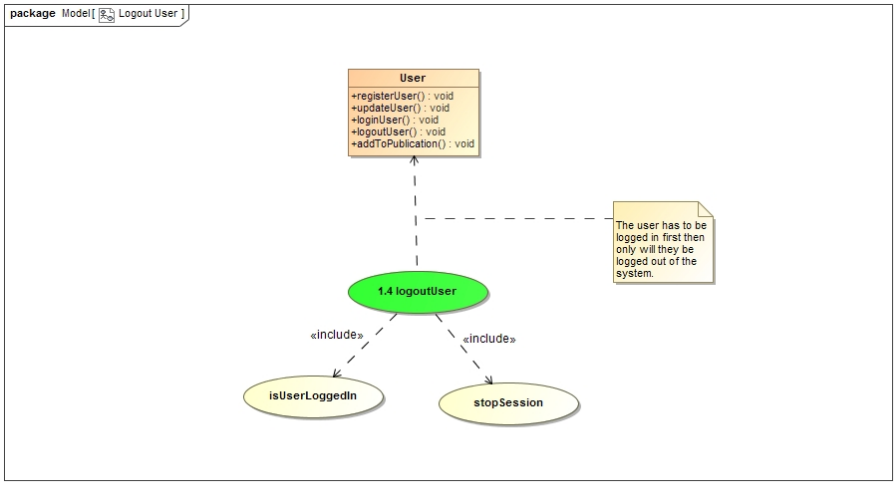


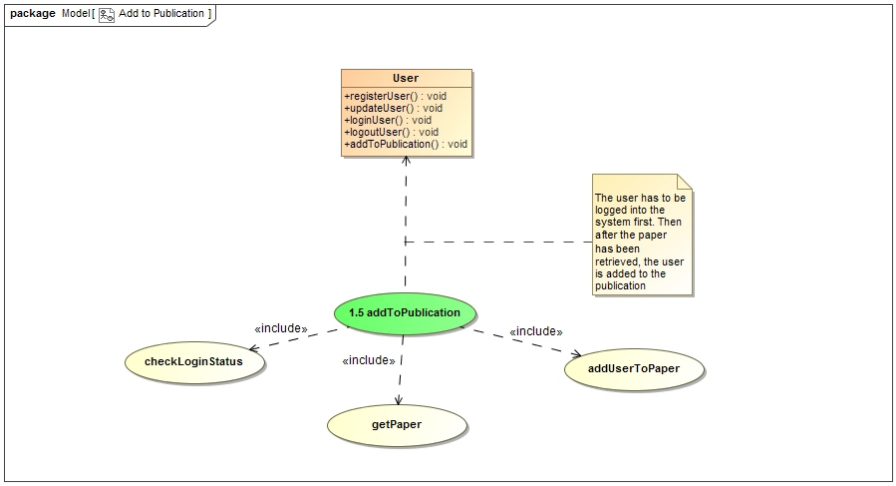
Figure 5: Use case diagram for the Conference, Journal and Thesis Subsystems


package Model[Register User]
User
+registerUser() : void
+updateUser() : void
+loginUser() : void
+logoutUser() : void
+addToPublication() : void
The user can only be registered by admin and the user can't have been registered before this. After registration, the user is added to the database and linked to a publication
1.1 registerUser
«include»
checkIfAdmin
«include»
checkIfUserExist sAlready
«include»
addUserToDatabase
«include»
linkToPublication

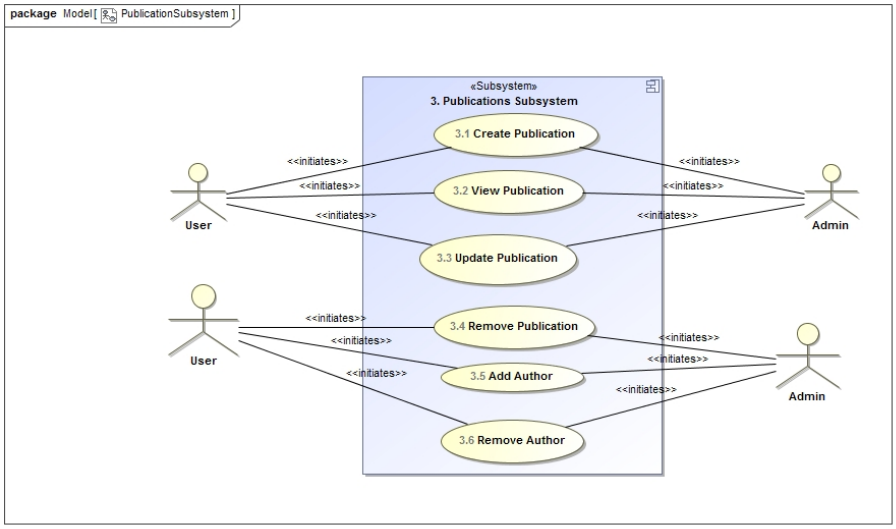

package Model[Update User]
User
+registerUser() : void
+updateUser() : void
+loginUser() : void
+logoutUser() : void
+addToPublication() : void
The user has to be registered to the system then the changes to the users profile are saved after the use case.
1.2 updateUser
«include»
getUser
«include»
saveChanges


package Model[Login User]
User
+registerUser() : void
+updateUser() : void
+loginUser() : void
+logoutUser() : void
+addToPublication() : void
The user has to be registered to the system first and if the login details are correct the user is logged into the system.
getUserPublications
«include»
1.3 loginUser
«include»
checkUserExists
«include»
«include»
startUserSession
validateLoginCredentials


package Model[Logout User]
User
+registerUser() : void
+updateUser() : void
+loginUser() : void
+logoutUser() : void
+addToPublication() : void
The user has to be logged in first then only will they be logged out of the system.
1.4 logoutUser
«include»
«include»
isUserLoggedIn
stopSession



5.3.2 Publication Subsystem Use Case



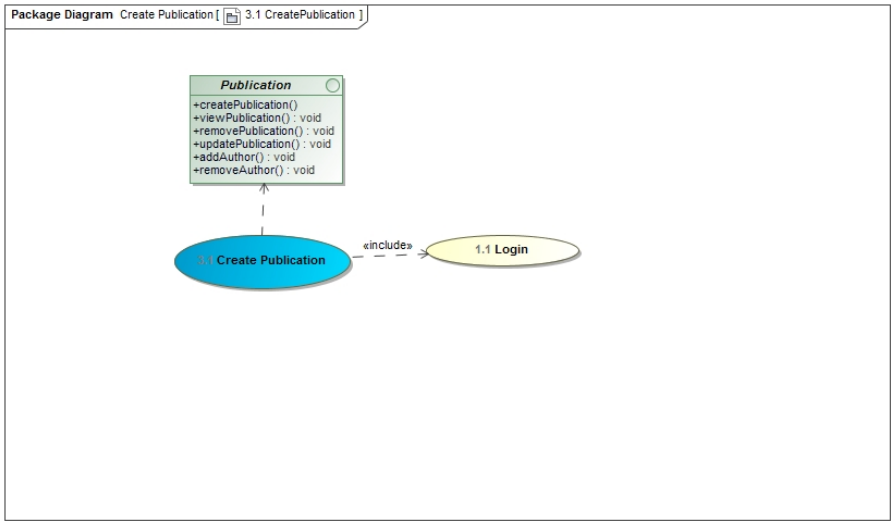


Figure 6: Create Publication

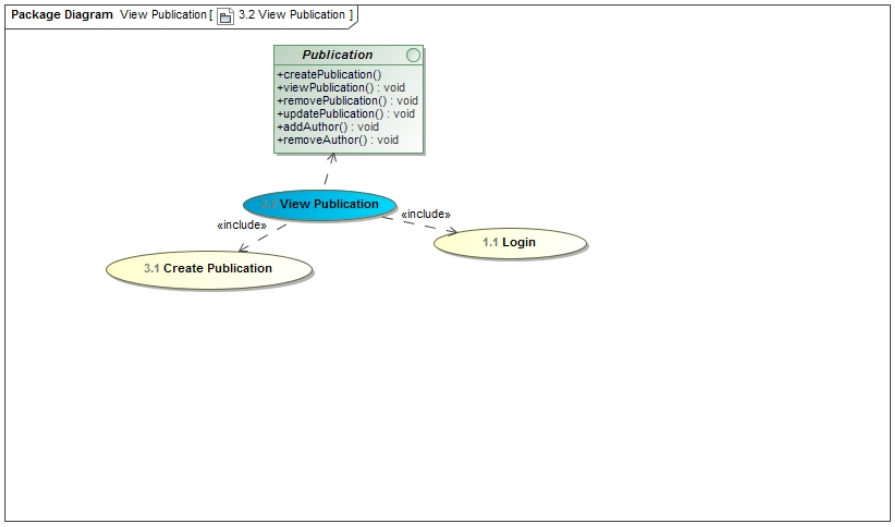


Figure 7: View Publication

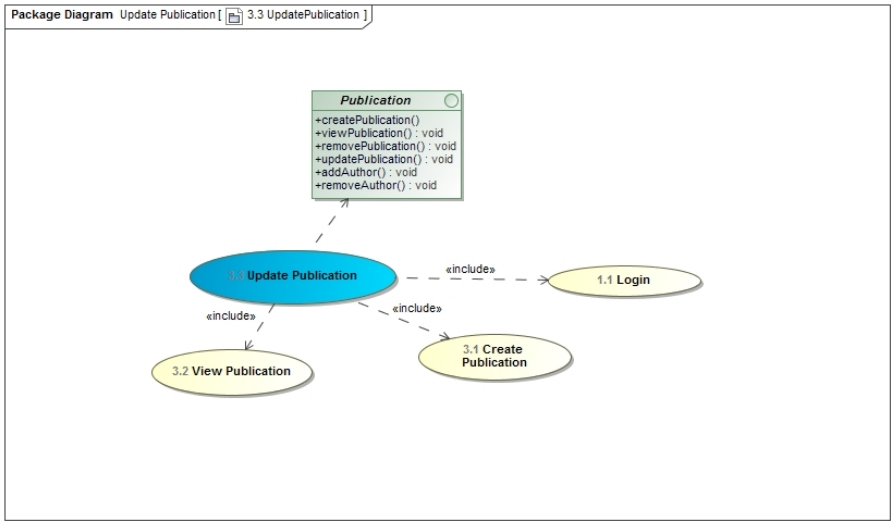


Figure 8: Update Publication

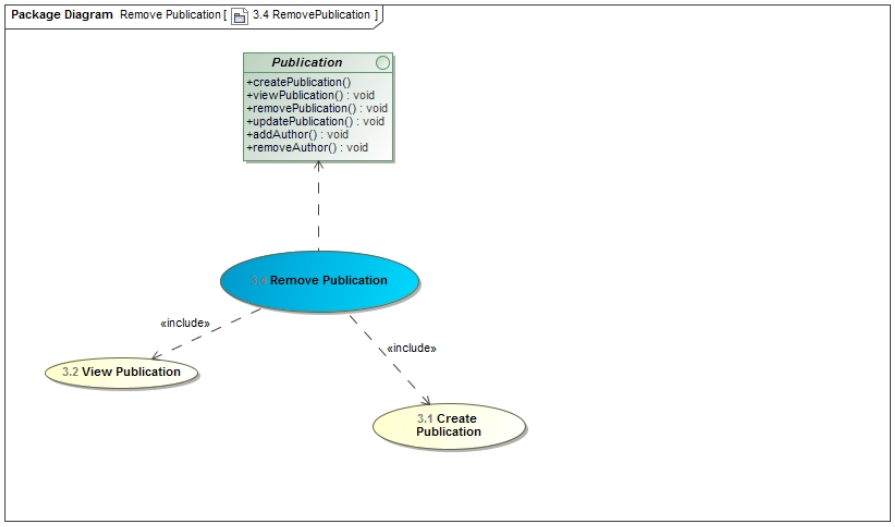


Figure 9: Remove Publication

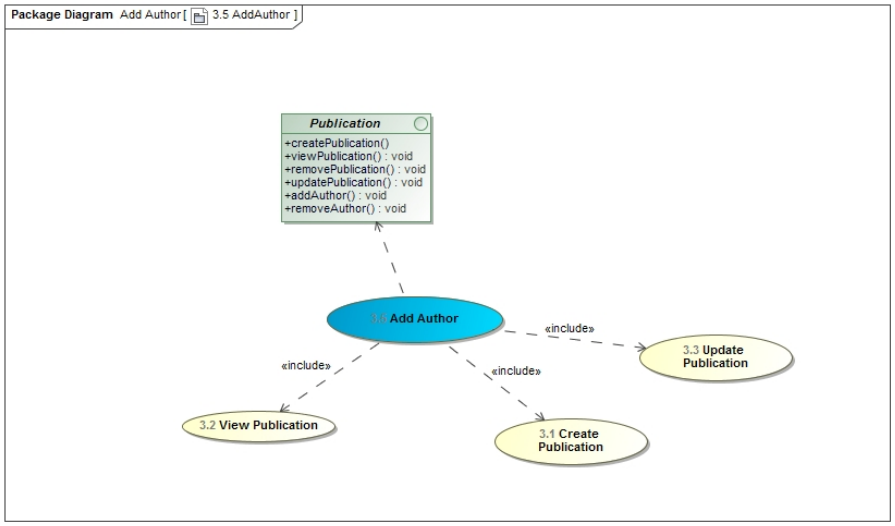


Figure 10: Add Author

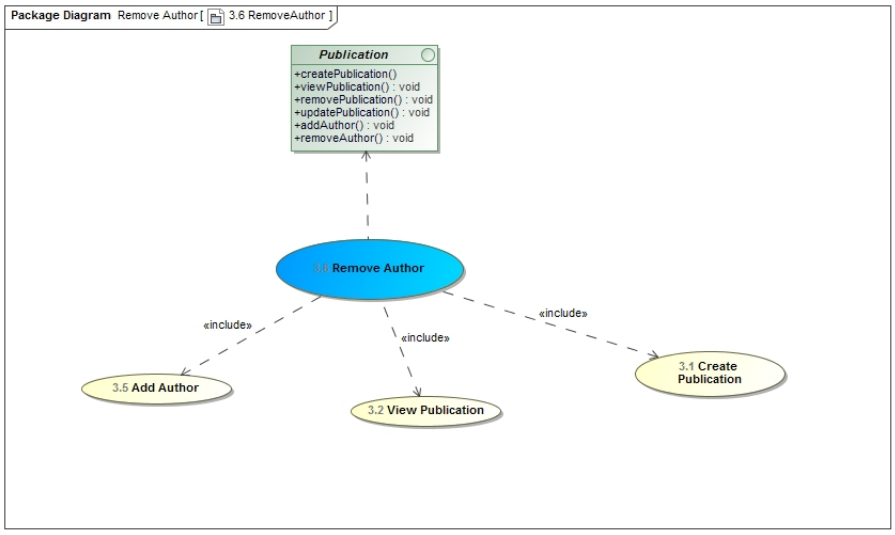


Figure 11: Remove Author

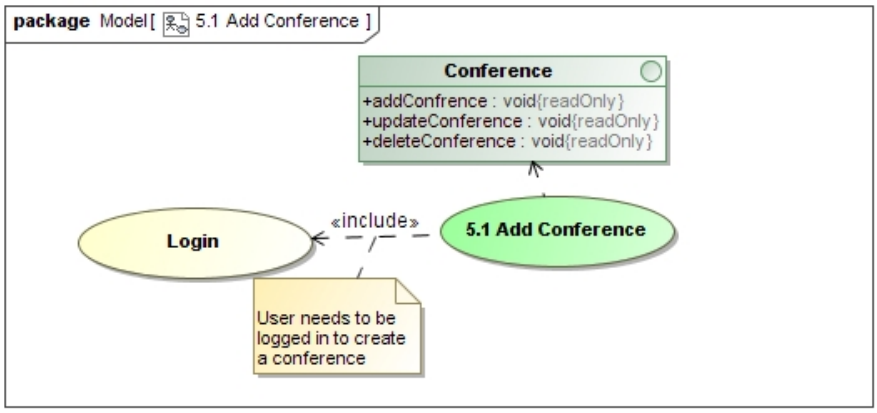
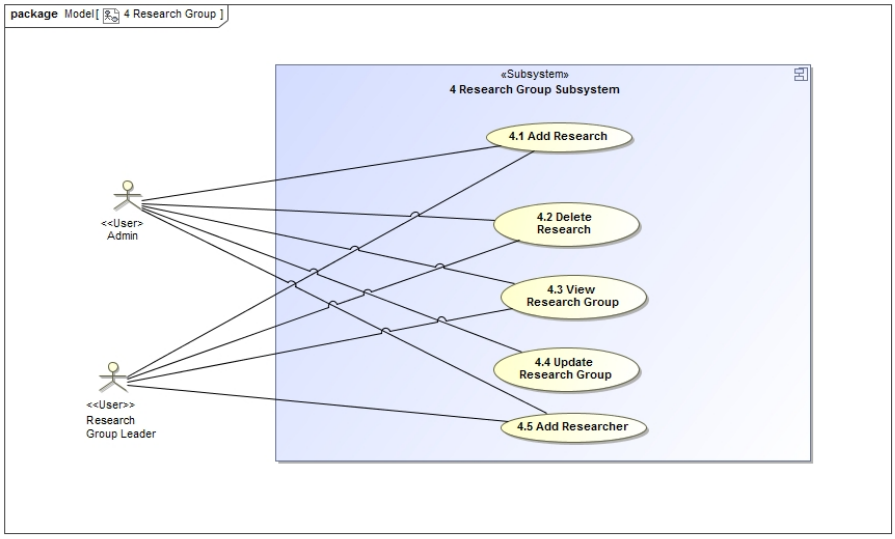


Figure 12: Add Conference

5.3.3 Research Group Subsystem Use Case



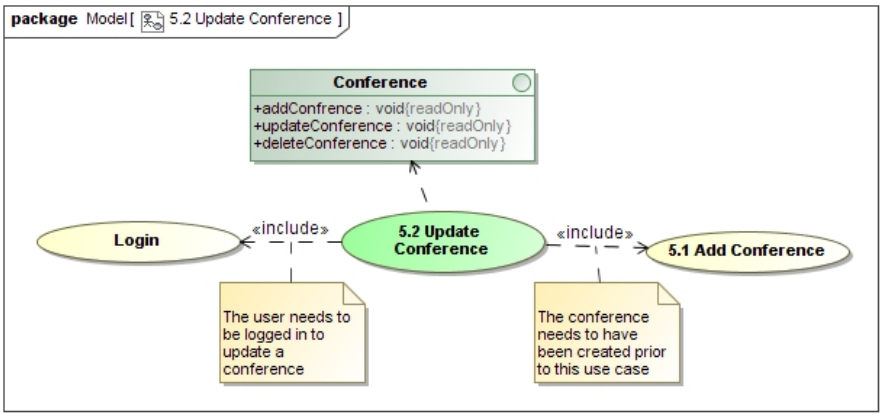


Figure 13: Update Conference

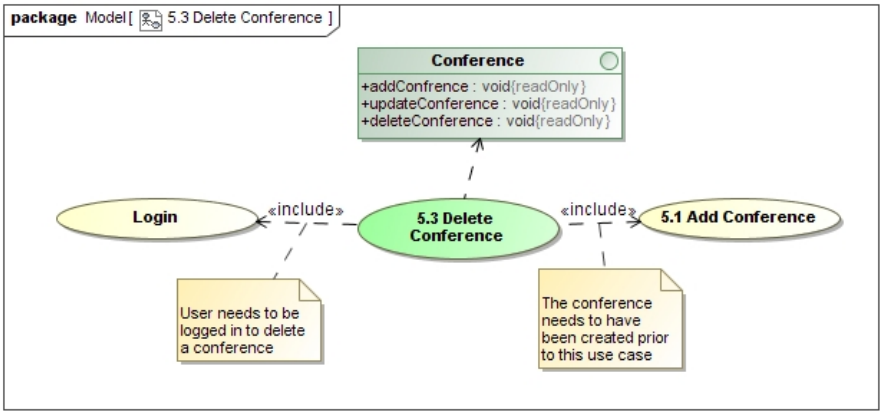


Figure 14: Delete Conference

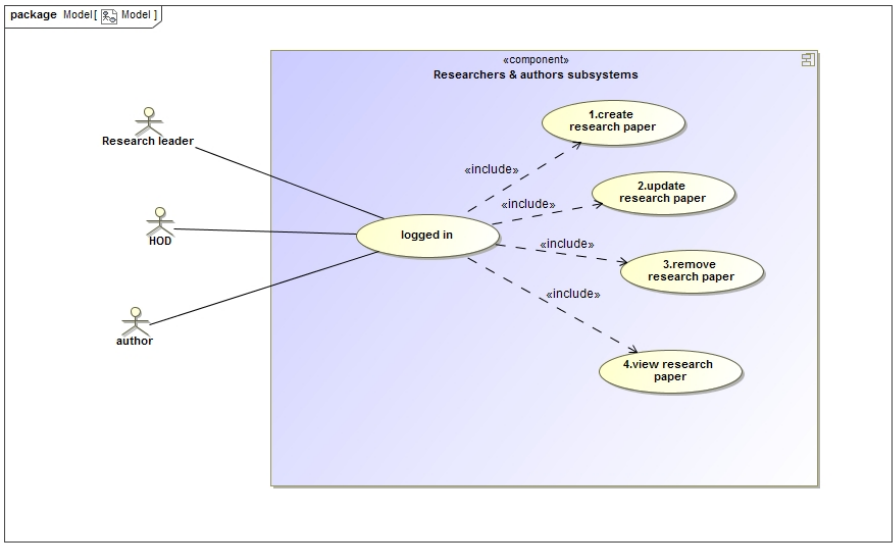


Figure 15: Research and authors use case diagram

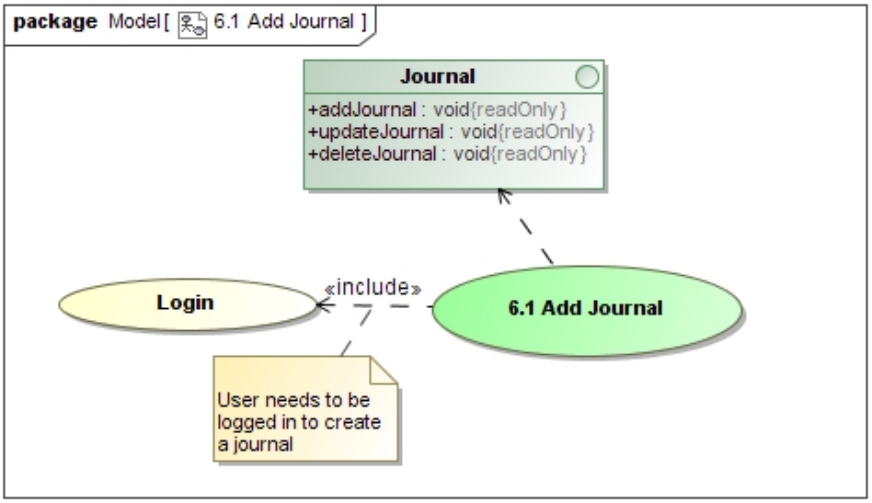


Figure 16: Add Journal

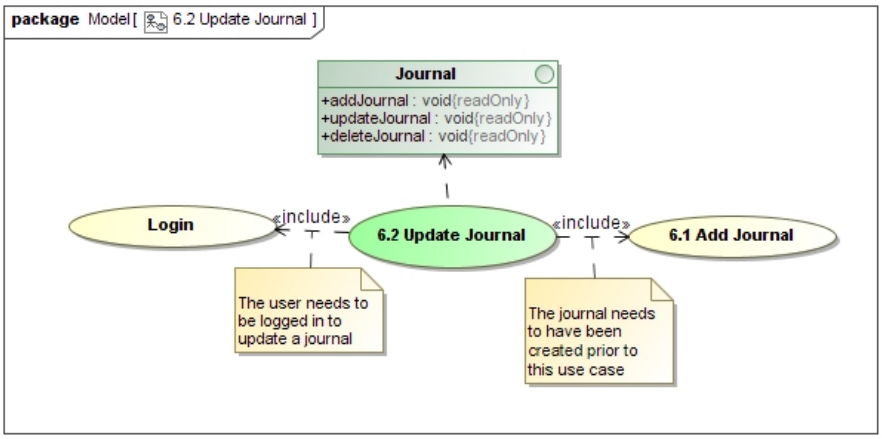


Figure 17: Update Journal

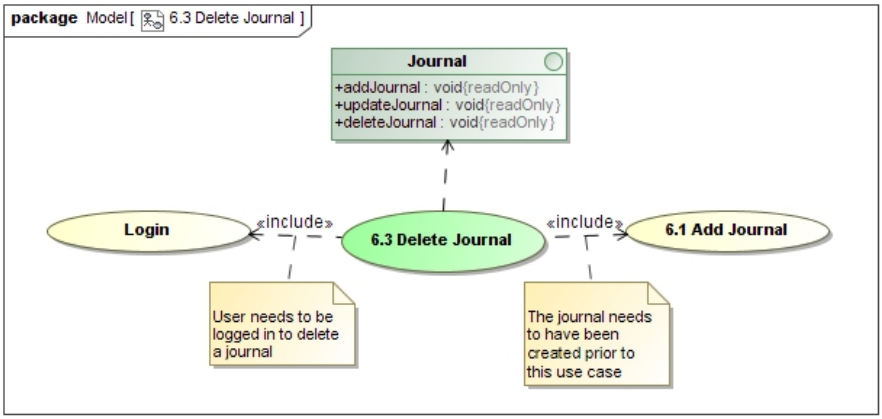


Figure 18: Delete Journal

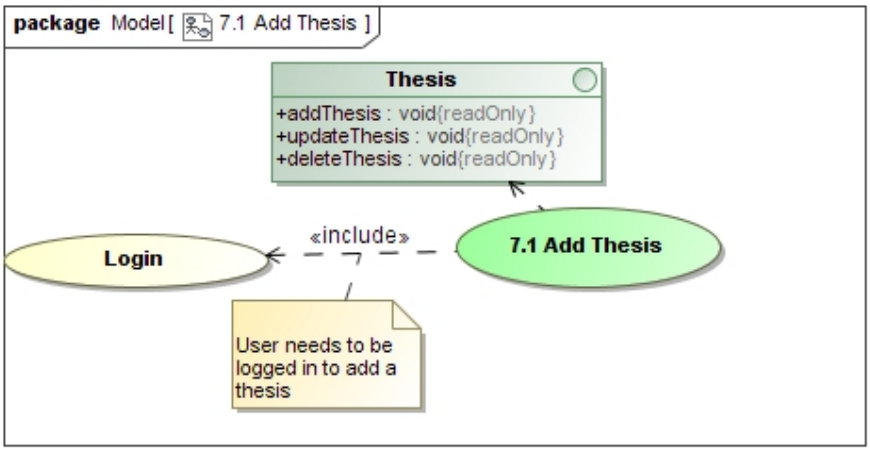


Figure 19: Add Thesis

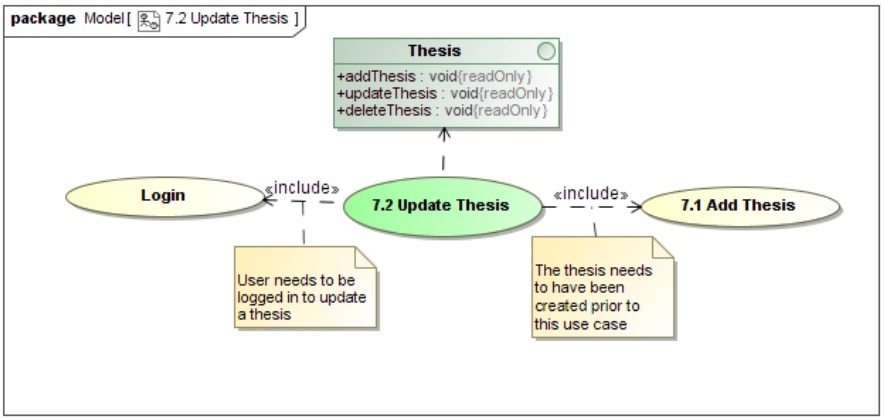


Figure 20: Update Thesis

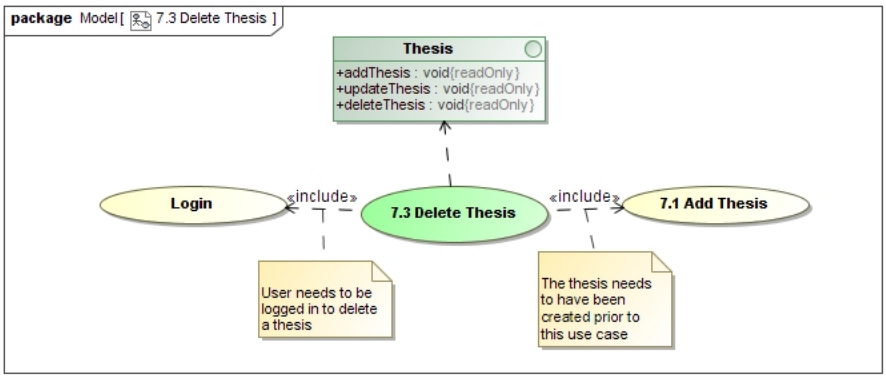


Figure 21: Delete Thesis

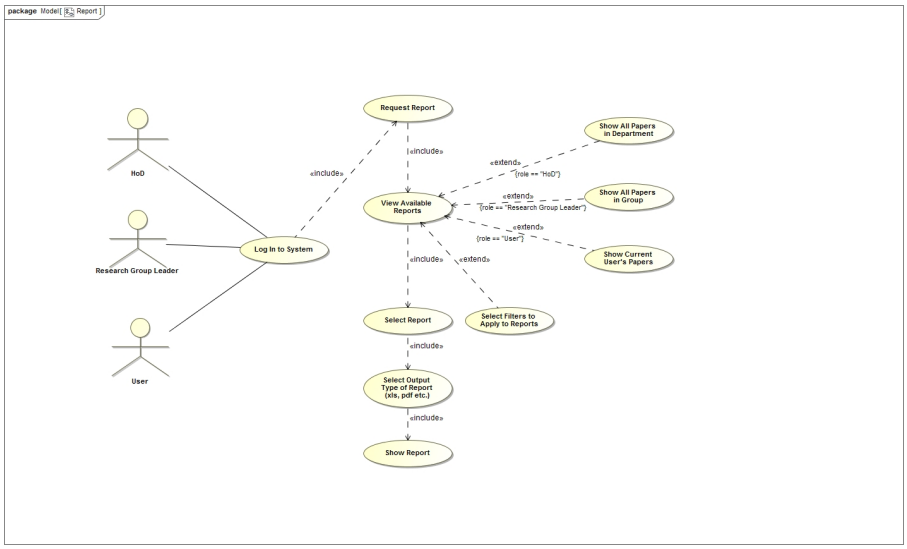
5.3.4 Conference Subsystem Use Case

5.3.5 Research and authors Subsystem Use Case

5.3.6 Journal Subsystem Use Case

5.3.7 Thesis Subsystem Use Case

5.3.8 Report Subsystem Use Case



5.4 Process Specifications

5.4.1 Conference Subsystem

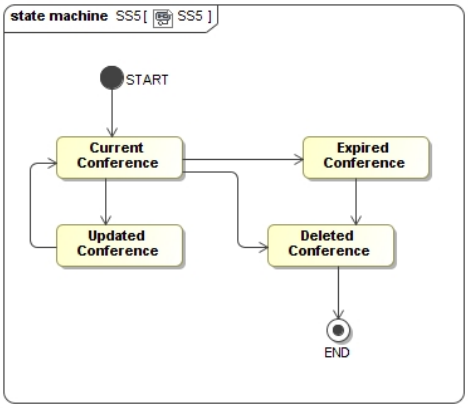


Figure 22: Conference State Diagram

5.4.2 Journal Subsystem

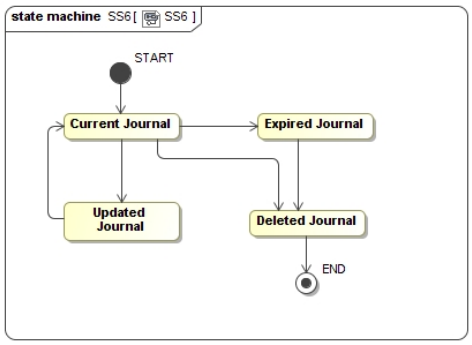


Figure 23: Journal State Diagram

5.4.3 Thesis Subsystem

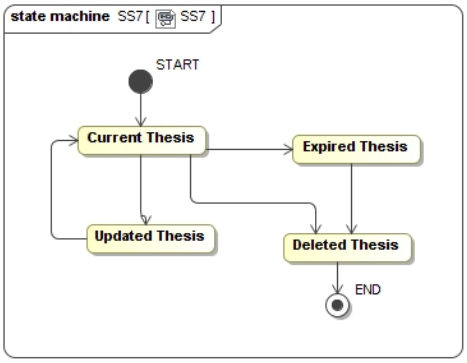


Figure 24: Thesis State Diagram

5.5 Domain Model

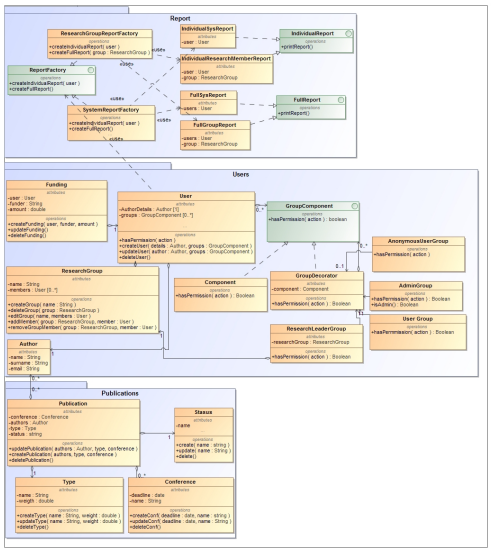


Figure 25: Domain Model

6 Open Issues

- Specifics with regard to the DoE and UP weighted contributions
- Further elaboration on how funding occurs.

7 Architectural Tactics or Strategies

- Scalability - It is estimated that there will be approximatly one hundred users. The system should be able to handle each user being active at the same time even though it is unlikly.

7.1 Reliability

Any data that is added to the system must be checked before it is stored. Any incorrect data (e.g. letters in a date) must be fixed before the form can be submitted. This ensures that the stored information is consistent.

A strict format must be applied to any reporting and any generated report must follow the format. If needed data is missing, there should be stand-in messages (such as "No data available").

7.2 Scalability

In order to reduce the load of multiple users, use cases that are common (such as some reports) should be optimized.

If multiple users have access to the same resources, the addition of data should be done on a first-come-first-serve basis, and the newest information should override older data.

When the amount of users logged in exceeds the load capabilities of the server, the server should be able to scale up in order to handel the excess requests.

7.3 Cost

Reports could be pre-processed and cached so that the same report is not generated multiple times, saving on resources. The system should be able to produce subsets of the report at a relativly low cost (e.g. a report for all researchers under a specific research group), and re-use those subsets to make the larger reports.

7.4 Security

Strict user control must be enforced, ensuring that each step of any use case makes sure that the user has the correct authorisation. By using hashing, passwords can be stored safely in the database, with a lower risk to user information leaks. **No plaintext passwords should be stored**.

7.5 Audibility

By using multiple log files, the different kinds of actions can be individually logged (i.e. individual logs for Errors, Logins, Data changes etc.), and there should be log files for individual days (a new log for each day) so that the information is organised. Over time, old log files can be merged together for archiving purposes.