UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS 301 - Mini Project

Assignment 1

Software Requirements Specification and Technology Neutral Process Design

---

Team Delta

Mpho Sharon Baloyi (14133670)

Dirk de Klerk (28159102)

Killian Kieck (12252213)

Daniel Malangu (13315120)

Dzilafho Mulugisi (13071603)

Duncan Smallwood (13027205)

Dian Veldsman (12081095)

Contents

1 Introduction

This document aims to set out the functional and non-functional requirements of a system as specified by the Department of Computer Science. The system is required to allow the department to collaborate, share, and maintain research articles in an effective and efficient manner. The document will also serve the purpose of communicating the requirements and specifications as needed by the client.

2 Vision

The client for this project, Department of Computer Science, has called for the design of an application that will allow the department to keep track of all research publications written and published within the department. The main idea behind the project is to alleviate the stress and time required for collaborating on, maintaining, and completing research articles. We have therefore envisioned the following goals:

- Simplify the effort required in maintaining publications as well as pending works.
- Ability to add/remove/specify main author and co-authors.
- Keep track of impact scores of different publications
- Allow authors and staff members to collaborate on articles.
- The ability to add/remove/edit meta-data of said articles.
- Allow different levels of privilege on user accounts.
- Afford privileged users (HODs) the ability to access addtional information on authors, publications and/or a summary page.
- The need to integrate the software in android or web based applications.
- Afford only head authors the ability to remove co-authors as well as transfer authorship.

3 Background

We find ourselves within the Information Age which means we are bombarded with more information than we can handle. This implies the rise of new challenges. How does one deal with this information overload in a timely and effective manner? Technology is the primary cause of the problem but at the same time provides us with multiples solutions in the form of networking.

The very nature of research means that a team or individual will constantly have to revise and apply changes to their work as new information becomes available. Some research efforts are simply to large handle on an individual level as is the case with interdiciplinary and cohort studies. Thus effective team work is required. When team members find themselves in remote locations, this can lead to a serious problem. How does one succesfully collaborate effectively and efficiently?

The above problem is what is currently hindering research efforts within the Department of Computer Science and the world! It is for this reason that the client is interested in developing a research repository that will allow them to share, maintain, and collaborate on various research papers in an effective and timely manner. A system, that is easy to use, is thus suggested to enable remote collaboration on research materials allowing the user to save considerably on time and effort. Such a system would certainly improve the throughput of research efforts, thus allowing authors to focus on quality research rather than deadlines.

4 Software Architecture

4.1 Architecture Requirements

4.1.1 Architectural Scope

4.1.2 Access Channel Requirements

This section specifies and describes different access channels through which the services of the target system will be accessed by humans. The different functions of the system will be highlighted and how different access channels will provide access to these functions. The system must be able to provide the platforms with an interface through which they can access the functions this will be done through the use of a Application Programming Interface (API).

To access the services of the systems users have to be connected to a network preferably the internet instead of a LAN, this is to enable users to access the services of the system from anywhere around the world. The type of devices that will enable access to the system's services are devices capable of connecting to the internet.

To provide access to the services of the system, the client requested that the following platfroms be used:

- A Web Interface
- An Android Application

The above mentioned platforms will have ways of providing the functionality of the system through their methods that are defined in their API's. The access channels will have to provide the users with the ability to view certain data structures , search through the data structures, create new entities (users and publication meta-data), delete existing entities, update attributes of existing entities , to open, view, edit or import certain types files (excel,image files etc), user registration, log in and authentification.

Android Application.

In addition to the device being able to connect to the internet it has be able to run on this operating system. Android provide access to its features via its API, therefore the transfer of data between the system and the platform will be done through the API's. Different versions of the operating system can be used to access the system's services,preferably the recent version 4.4

Android offers a range of libraries and functions that enable its application to send,receive,view data in various formats as well as interfacing with other applications to provide more functionality.

Web interface

For clients who access the system's services via a web interface can use any of the avaliabe web browsers such as firefox, Internet Explorer, Google chrome, Safari, Opera

and others. Web browsers make use of web technologies to provide the functionalities that are provided by the system's to the users

Restful web services clients will interact with our system via its RESTful api that makes use of the http request to obtain data from other systems over the internet.

4.1.3 Quality Requirements

Quality not only refers to how many clients requirements the product fulfils. It also refers to how much effort is put into a product to satisfy the consumers needs and wants compared to other similar product. The aim however is to fulfil, at the least, every need over the wants of the client.

- Performance:
  - Program should be designed to be run as efficient as possible.
  - Execution time should be reduced by avoiding unnecessary comparisons and recursive function calls
  - Database queries should be optimised by avoiding needless joins and using stored procedures
- Reliability:
  - Information stored in the database should remain correct when being transported from the system front-end.
  - System should always be available for use unless the system is intentional shut down for maintenance
- Scalability:
  - System needs to be handle growth in terms of the number of users by:
    - Managing activities running concurrently
    - Storing information in lightweight format to save storage space
- Security:
  - The application should maintain permissions for access and modification of information for the different user groups.
  - Also proper authentication protocols should be implemented for every user group to prevent unauthorised users access at any user group.
- Flexibility:
  - There should be provisions for the addition of new user groups and the modification of existing ones.
  - It should be possible to add new groups of data without rebuilding the system
  - Provision for the modification of data types
- Maintainability:

    - Loose coupling should be encouraged, creating pieces that can be repaired without making huge modification to other parts of the system.

- Auditability:
    - Modifications on database transactions should be stored and logged
    - Information of when and where users access the system should be logged

- Integrability:
    - Program should follow the coding standards specified by the client

- Usability:
    - System should have options grouped in logical manner
    - Help hints should be available for more advanced procedures in the program
    - Where information should be entered in a specific format an example of it should be displayed to the user
    - Manual should have a detailed description of all the functionality in the program

4.1.4 Integration Requirements

Integration involves merging various subsystems into a single cohesive system. With regard to this project the subsystems may consist the application being developed as well as database management systems.

In order to successfully and safely integrate these subsystems they need to be directly and securely linked to one another. This can be easily achieve by creating a secure connection through the application to the database management system.

The system itself will need to be integrated on two seperate platforms, namely a website and mobile application. In the case of the website it may be desirable to display certain content dynamically as opposed to reloading a new page for each item.

Integration Channels

It is recommended that the following be used to successfully achieve the aforementioned integration:

- PHP: Hypertext Preprocessor

In order to achieve this dynamic display the following technologies will be used:

- AJAX
- JSON

Quality Requirements Integration

Systems consist of both functional, what a system has to do, and non-functional requirements, how a system needs to behave. In other words non-functional requirements refers to qualitative attributes of a system. The following quality requirements will need to be integrated into the system:

- **Low resource consumption(Mobile use):** the user should be kept in mind when developing the system, to ensure that no unnecessary resources are used.
- **Good performance:** The system must achieve what it is designed to do in as little time as possible.
- **Reliability:** The system needs to be stable and provide the user with access at all times.
- **Security:** The materials hosted on the system as well as the user accounts needs to be protected at all times from unauthorised users.
- **Safety:** Integrity of the data hosted on the system needs to be assured.
- **Scalability:** The system needs to accommodate growth.
- **Flexibility:** The system needs to be easily adaptable to change should the client require additional functionality for example.

4.1.5 Architectural Constraints

The Architecture constraints were indicated on 16.02.2016 in a Client requirements session and lists the following technologies that will be used in the project:

- HTML (Hypertext Markup Language)
- PHP
- AJAX (Asynchronous JavaScript and XML)
- Git (Version Control System)
- Andriod

4.2 Architectural patterns or styles

Patterns and Styles define the structure to be used when coding an application. In this ection we will be describing the different patterns and styles to be followed when developing the system.

- Layering
  - Client-Server (2-tier)
    - Server handles storage of data into the database and provides data on request. The Server is responsible for all back end functions of the software including the verification of users for security.
    - Client provides an interface between the system and its users. Responsible for the front-end of the application, the sending of queries and requesting services from the server.
- MVC
  - Model
    - Manages the data and logic of the software.
    - A DBMS (Database Management Software) will used to model the data and control the data flow.
  - View
    - Interchangeable platforms on which the data will be represented, android for mobile and web version for all other devices.
    - JSON for lightweight transfer of data and AJAX to communicate with the server and access the system.
    - Data can be represented and structured sufficiently in HTML.
  - Controller
    - Accepts the users input and converts it into commands the model or the view can understand. PHP processor will execute the commands between the view and model..
- Hierarchical
  - User groups are divided into different levels where the groups at top level having access to the most options in the system and the groups at the bottom level having the least amount of options.
  - Controller in the MVC will Implement this pattern by verifying the user input against the model.

4.3 Architectural tactics or strategies

Tactics in general, serve as the building blocks from which architectural patterns are ultimately constructed or developed. In the following section we will discuss, briefly, the various tactics that may be used to realise various quality requirements of a system.

4.3.1 Performance:

The performance quality attribute concerns itself with decreasing latency while increasing the throughput of a particular system.

Latency can be decreased if a system is responsive, including its ability to respond to a request, such as a database request for example, in rapid time.

The throughput of a system is increased if a system is able to respond to multiple requests at the same time without hindering its performance.

Performance can thus be improved by simple resource management. This can be achieved in a multitude of ways. Introducing concurrency to the system and thus spawning multiple threads to take care of user requests. Make use of a scheduler to provide each request or operation with a fair share of the resource.

4.3.2 Reliability:

This quality attribute concerns itself with whether a part or component, or system as a whole can perform its intended operations on a given environment for a specific amount of time without failure.

To improve a systems reliability the system needs to be fault tolerant. This implies that a system needs to be able to maintain a certain level of performance when a fault has occurred or an interface has not been used for its intended purpose.

Reliability is also improved through the recoverability. This implies that a system should be able to recover to a certain level of performance and recover data.

To achieve the above mentioned, the handling of exceptions and errors is general is of the utmost importance. Error and Exceptions handling will be a useful tactic for allow faults to occur as well as giving the user the option of recovering from a fault.

4.3.3 Security:

Security concerns itself with unauthorized access. More specifically it intends to prevent unauthorized access to a resource, information or a particular service. This includes both aspects of confidentiality and integrity. Thus the information on that system is private and cannot be changed by any unauthorised users.

To prevent attacks a few tactics may be used. Thus includes authenticating users during login, thus checking that a person is who they say they are. Provide authorisation is another tactic that ensures that authenticated users are only allowed to access information relevant to them. Limiting access to resources is another tactic

in which the number of access points to particular information is limited, and in turn limits opportunities for unauthorised access.

It is of course impossible to prevent all forms of attack so a strategy need to be in place that can detect attacks.

4.3.4 Usability:

Simply put, this quality attribute is all about ease of use. Usable systems imply that a system is easy to learn, to navigate about and to use in general. Usable systems should be able to guide users through the entire process and provide feedback on both success and incorrect operations.

To achieve this, tactics that keep the user in mind need to be employed. In certain situations a user may enter an operation by accident and would want to leave. It is not feasible to force them to first complete the operation before continuing thus a simple Cancel operation may be used.

Likewise the system needs to cater for user mistakes as well as afford them the opportunity to save progress should it be applicable during certain operations. This can be easily achieved through undo, load, and save operations.

From a visual design perspective usability can be improved by orders of magnitude by simple providing a visual hierarchy and logical layout of a GUI. The planning and use of appropriate colours should not be underestimated either.

4.3.5 Scalability:

This quality attribute is concerned with how predictable a system will perform as the workload or volume of requests increases. Thus a system needs to be able to handle increases in users and requests at any given time.

This can be achieved through optimization of operations that seem to be used quite frequently. For example querying a particular research article will occur more often than login and logout.

It is also important to note that scalability can only be achieved if performance goals are met, thus the tactics mentioned in performance are also relevant to scalability.

4.3.6 Flexibility:

Flexibility is concerned with the ease of which a change can be made to a particular system. This is also known as modifiability and has a direct influence on maintainability as well. Systems that are more flexible tend to be more maintainable. Flexible systems are also easier to port to multiple different platforms.

These systems in general tend to be modular and thus have propensity to adapt to certain changes.

The tactics for improving flexibility, quite simply put, is concerned with decreasing the effort need to make changes to a particular system. This can be achieved by limiting

what a change can do to a system, in other words what are the overall consequences of a change.

This can be achieved by grouping elements that belong together into a cohesive whole. This is because elements that work together are likely to change together if one is changed.

4.3.7 Maintainability:

This refers to the ease to which a software system can be maintained in terms of repair, adding new requirements and maximising other attributes.

Maintainability can be achieved through early planning. This involves anticipating what future changes may be made to the system. A rather important tactic is that of object orientated design. This allows program tasks to be more independent of each other and thus easier to maintain.

Maintainability can further be improved by making use of uniform naming conventions, Coding standards and styles.

Lastly, by providing proper documentation for the software, the maintainability can be drastically improved.

4.4 Use of reference architectures and frameworks

Reference architecture and software framework are often very closely related. Software framework is defined as a partial or complete implementation of reference architecture (Solms, 2015). Examples of software architecture include CORBA, MVC, Microsoft.Net or JBoss application server. A framework can thus be seen as a prepackaged implementation with the intent reduce the overall development time by allowing programmers to devote more of their time in development rather than dealing with low-level details of the system.

Reference architecture can be defined as a best-practices based template architecture which has been proves to address the typical challenges for a particular domain (Solms, 2015). Examples of reference architectures include JAVA-EE, Services-Oriented Architecture (SOA), Space-Based Architecture and AUTOSAR. One of the reference architectures we will be using for this project JAVA-EE. JAVA-EE or Java Platform, Enterprise Edition architecture is a very widely used architecture. It provides a reference architecture within which multi-tier applications are developed.

Another widely used is architecture is Services-Oriented Architecture (SOA). We will be using this in conjunction with JAVA-EE as our reference architectures. SOA aims to provide core integration architecture between various systems, to facilitate the orchestration of processes across the various systems used within the organization, the service provider and partner systems whose services are integrated into the business processes (Solms, 2015). It simply means that SOA provides an infrastructure for service reusability across all technologies and business units, as well as decoupling systems and technologies.

Collaboration between the two reference architectures is needed in order to satisfy all quality requirements for the project.

4.5 Technologies

Technologies simplify how the system operates and also add on to the overall look and appeal of the system when it comes to usability.

4.5.1 Programming languages

Programming languages offer the ability to communicate instructions to the machine and the database in this situation. They are used to control the behaviour of the systems and states of data in the system. For our system we will use the following languages.

- HTML5
  - This mark-up language is used to present and structure the content of the website. It acts as the physical access point to the system as this system is primarily web-based.
- JavaScript
  - We use this language to work with data and to employ functionalities like method executions.
- PHP
  - PHP serves as the access point for database management and manipulation.
- CSS
  - The Cascading Style Sheet provides customization abilities. It enables the styling of the web-page.

4.5.2 Frameworks

Software frameworks provide generic functionality which the user can choose whether to change or use in their default form.

- Bootstrap
  - The bootstrap framework simplifies the styling of the web page and creates a more friendly and pleasing environment for the system users.

4.5.3 Libraries

Libraries house a collection of resources for programmers that the programmers readily have at their disposal.

- jQuery
  - This is used to simplify the use of scripts (JavaScript) on the client side. It can be used to create animations and for event management.

4.5.4 Database Systems

These are computer programs which manage and run operations on the data contained within them.

- Apache Cassandra
  - This open-source distributed database management system handles the database system which hosts all the information about the documents.

4.5.5 Operating Systems

Having this as a web-based system enables us to support all operating systems. The core-functionality primarily depends on the web-browser so any OS which support Mozilla Firefox, Google Chrome and any similar browsers will run the system.

5 Functional Requirements and Application Design

5.1 Use Case Prioritisation

The Use Case Prioritisation is specified for each use case listed under the next section: "Use Case/Services Contracts".

5.2 Use Case/Services Contracts

5.2.1 Use Cases

- **Registration**

  Description: A user is able to register an account on the publishing website.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must make use of a unique user name.
  - A user can register an account if they are a staff member.
  - A user who is not a staff member must register through an administrator or Head of department.

  Post-Conditions:

  - The user information is stored in the database.
  - The user receives the login information.
  - The user is able to login the website.

- **Login**

  Description: A registered user is able to log into the website making use of the features available to them.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must have a registered account in order to login.
  - A user must login with the correct user name and password.

  Post-Conditions:

  - A user has access to certain parts of the website according to their privileges.
  - A user has access to certain functions of the website according to their privileges.

- **Logout**

  Description: User is able to logout of the web page.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must be logged into the web page in order to log out.

  Post-Conditions:

  - User no longer has privileges of being logged in.

- **Create publication**

  Description: A user can create a publication which holds information about the publishing, also authors who contributed.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must be logged into the web page in order to create a publication.

  Post-Conditions:

  - At least one user must be assigned to the publication (Creator or author).
  - Meta-data on the publishing must be stored in some form of database.).

- **Add author**

  Description: Add author - A user can add authors to their publication who have worked or helped with the publication.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must be logged into the web page in order to add an author.
  - A user must create a publishing before adding another author to the publishing.
  - The user being added must have a registered account with the website.

  Post-Conditions:

  - Added Author is listed on the publication.

- **Remove author**

  Description: A user can remove authors from their created publications.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must be logged into the web page in order to delete an author.
  - A user can only delete authors they have added.

  Post-Conditions:

  - Author deleted is removed from the list of authors on the publication.

- **Edit publication**

  Description: A user can change the details of their created publications.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must be logged into the web page in order to edit a publication.
  - A user can only edit their own publications.

  Post-Conditions:

  - Changes made to the publication meta-data is changed on the data on the web server.

- **View publication**

  Description: A user can view publications they are working on.

  Use Case Prioritisation: Critical

  Pre-Conditions:

  - A user must be logged into the web page in order to view a publication.
  - A user can only view publications they are directly involved in.

  Post-Conditions:

  - Information is correctly diplayed to the user.

- **View profile**

  Description: User can view their own and other user profiles.

  Use Case Prioritisation: Important

  Pre-Conditions:

  - A user must be logged into the web page in order to view a profile.

  Post-Conditions:

  - A user will view and have access to edit their profile.

- **Edit profile**

  Description: User can edit their own user profiles.

  Use Case Prioritisation: Important

  Pre-Conditions:

  - A user must be logged into the web page in order to edit their profile.
  - A user can only edit their own profile.

  Post-Conditions:

  - Any changes made to their profile will take effect.

- **Generate summary**

  Description: Certain users/members can generate information on the publications.

  Use Case Prioritisation: Nice-To-Have

  Pre-Conditions:

  - A user must be logged into the web page in order to generate a summary on certain publications.
  - A user can only generate a summary on publications they are involved in.
  - Head of department can generate summary on all publications..
  - Research leaders can generate summaries on fields of research.

  Post-Conditions:

  - Information displayed of selected publications must be accurate.

  Service Contracts

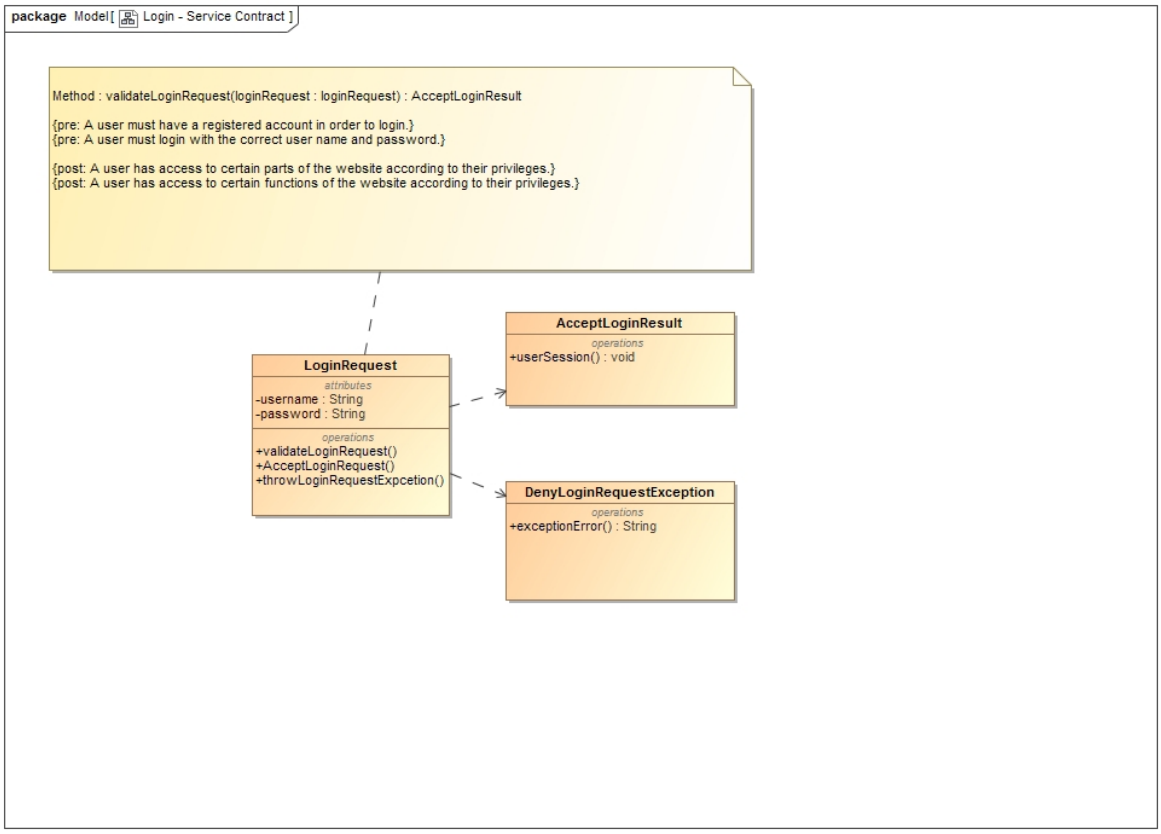


Figure 1: Functional Requirements: Post Subsystem

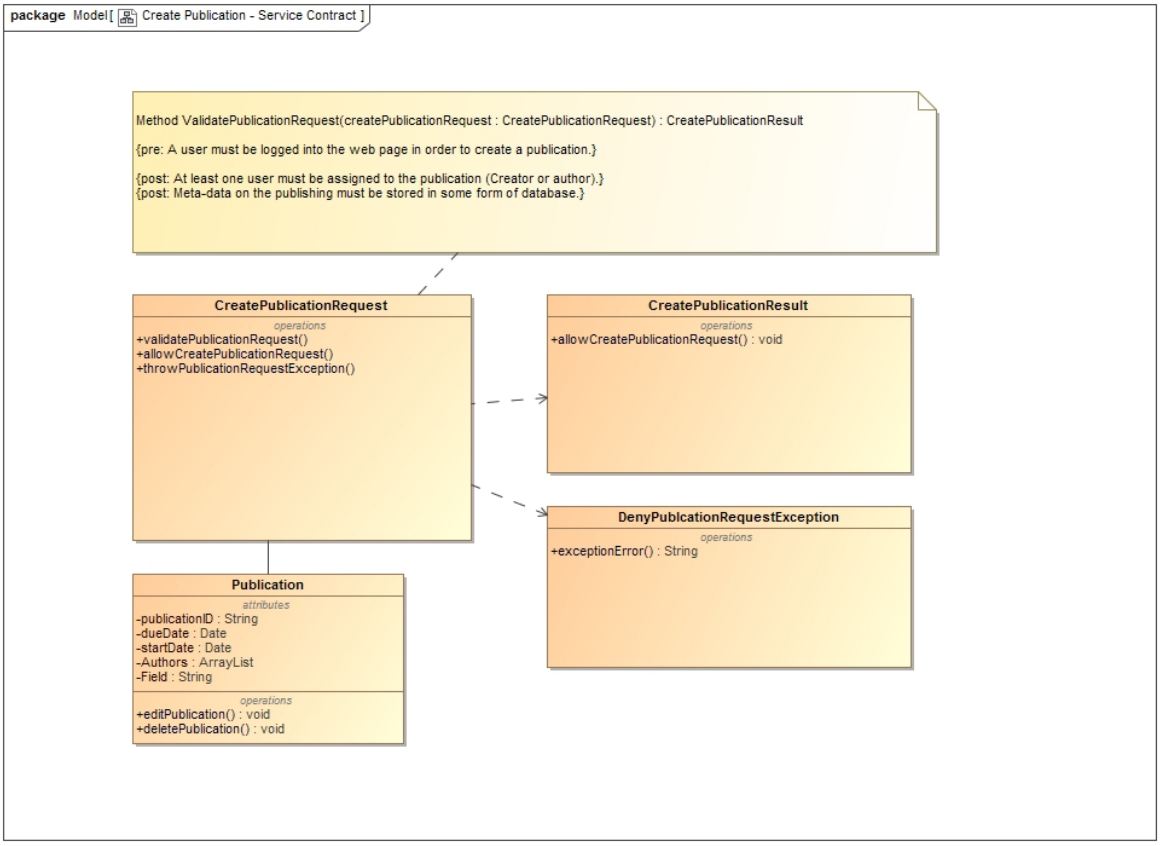
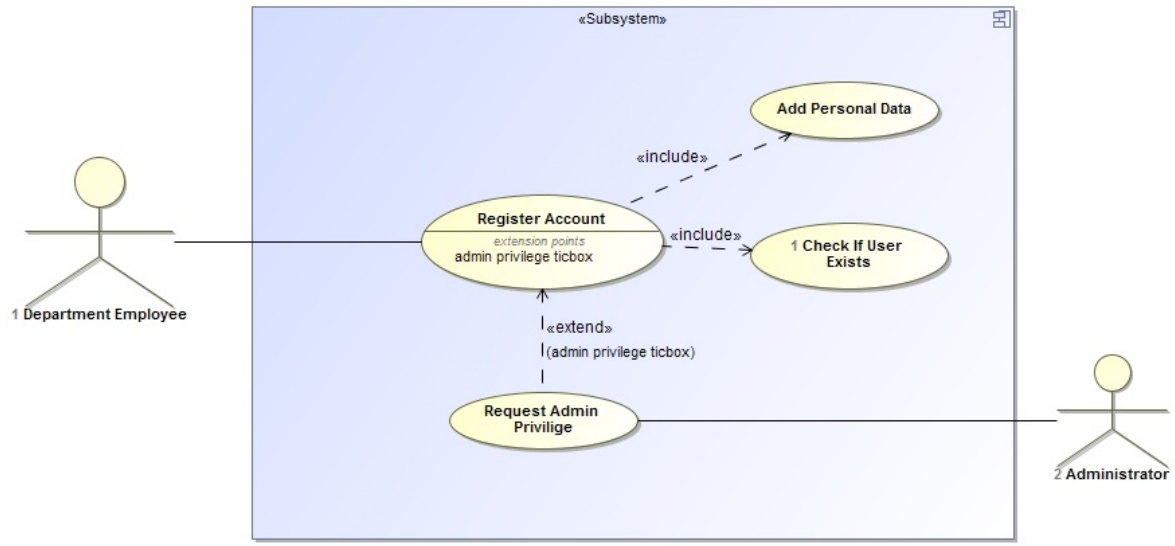


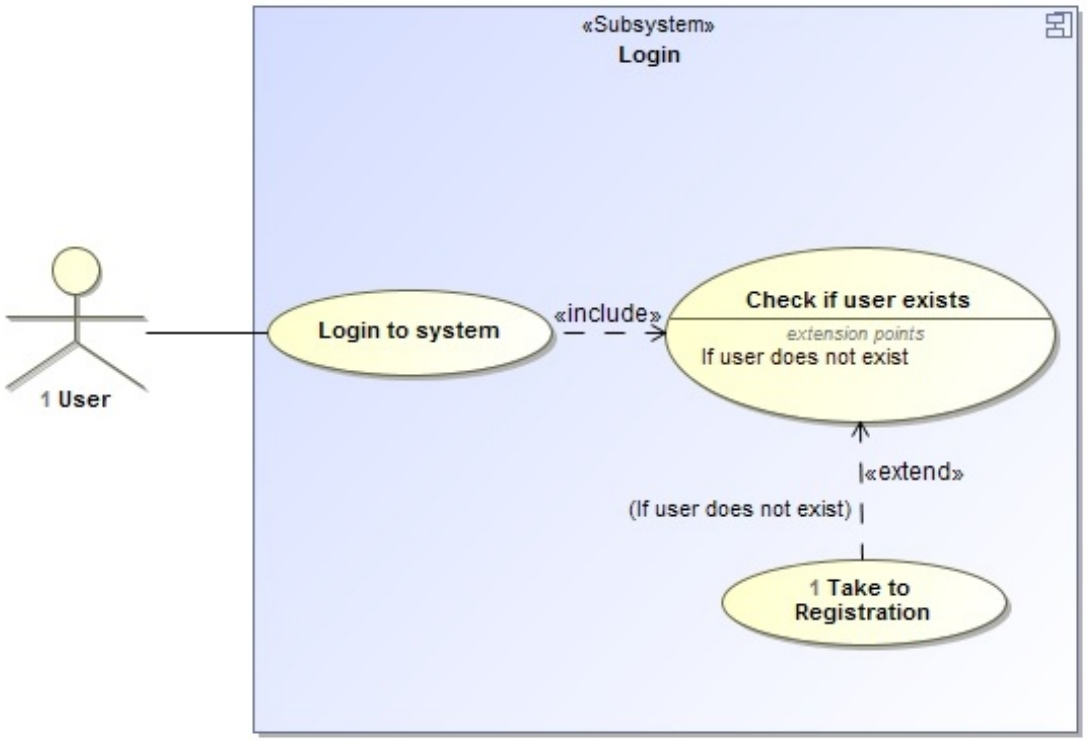
Figure 2: Functional Requirements: Post Subsystem

5.3 Required functionality

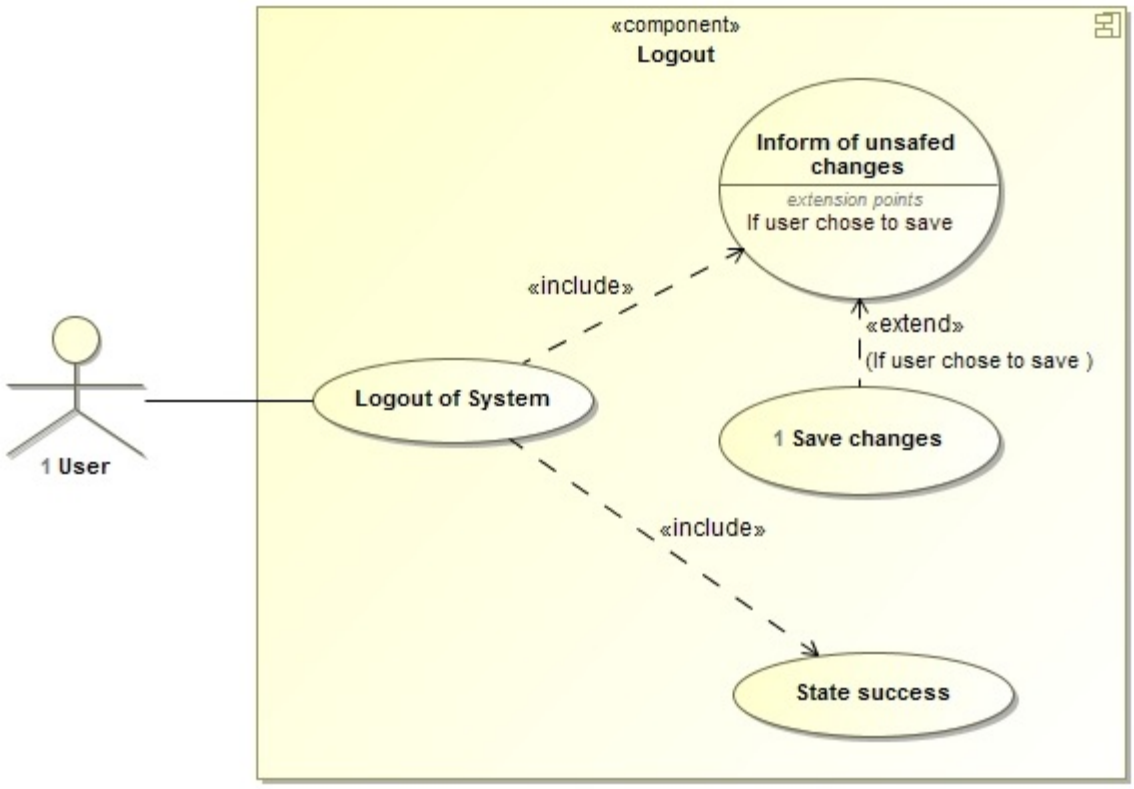
5.3.1 Registration Use Case Diagram



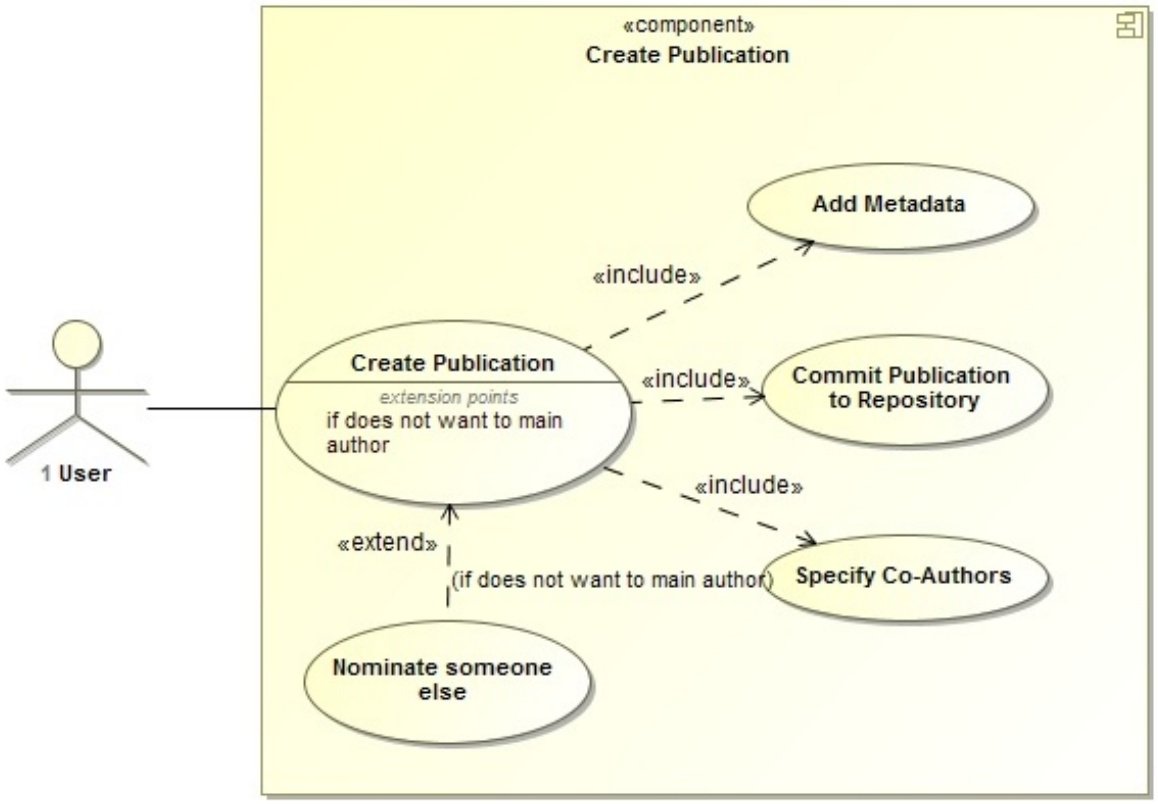
5.3.2 Login Use Case Diagram



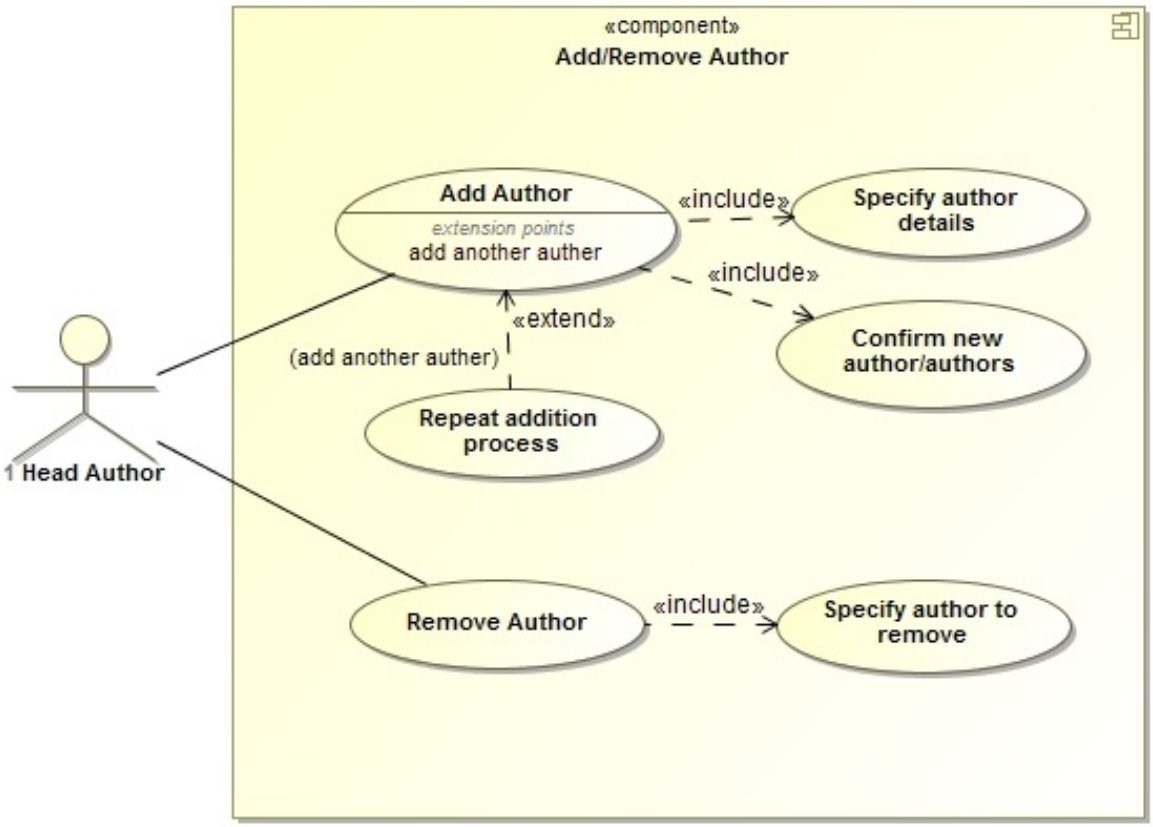
5.3.3 Logout Use Case Diagram



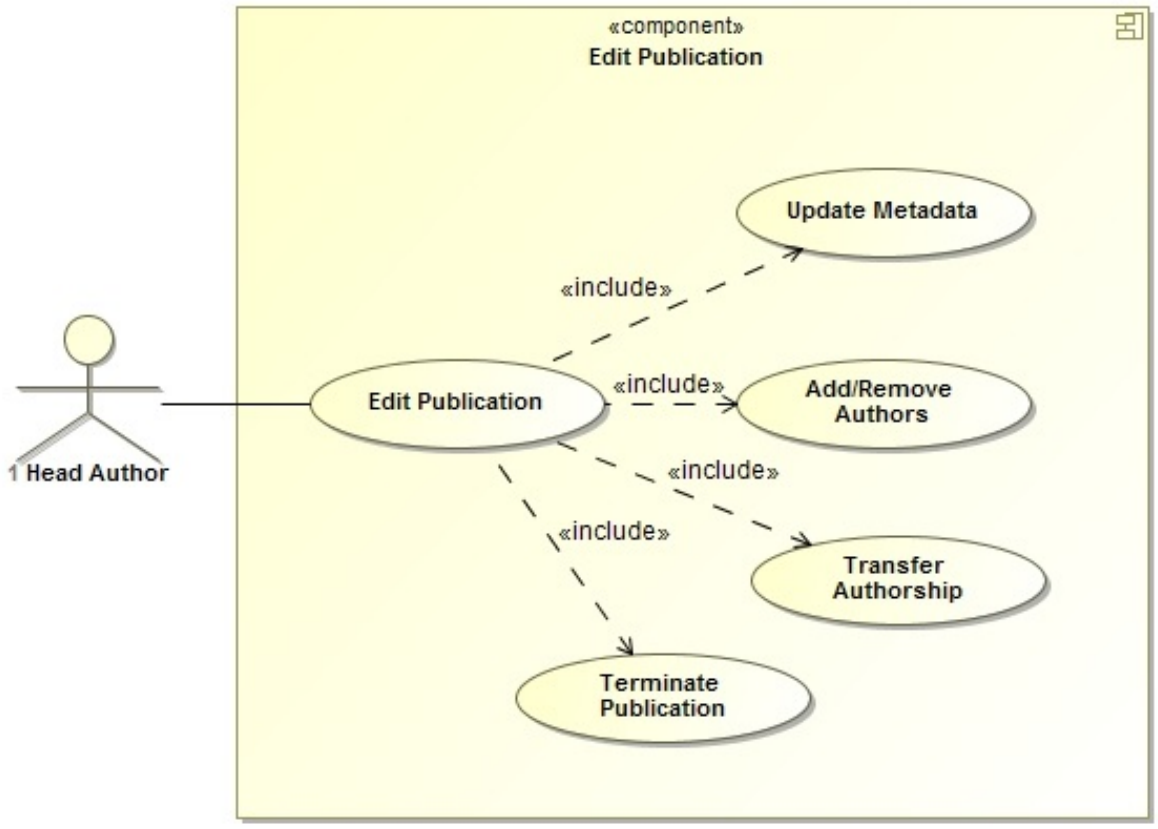
5.3.4 Create Publication Use Case Diagram



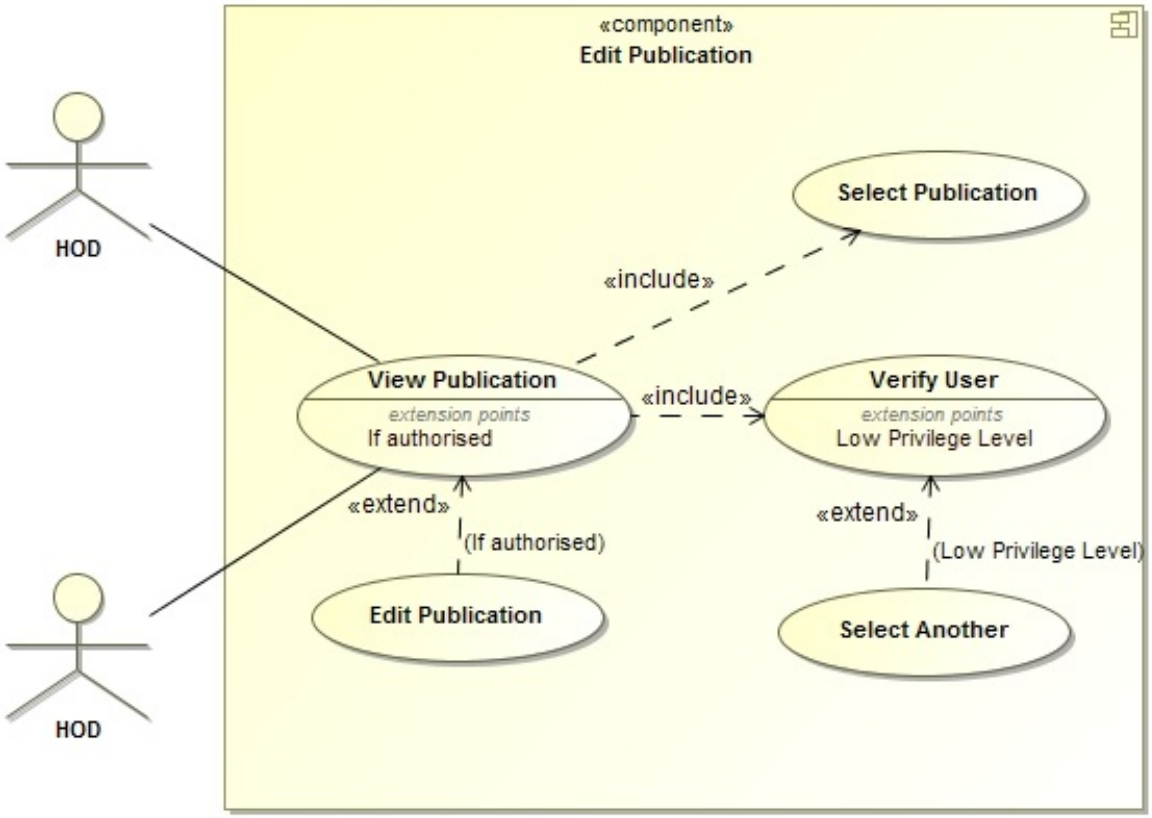
5.3.5 Add/Remove Author Use Case Diagram



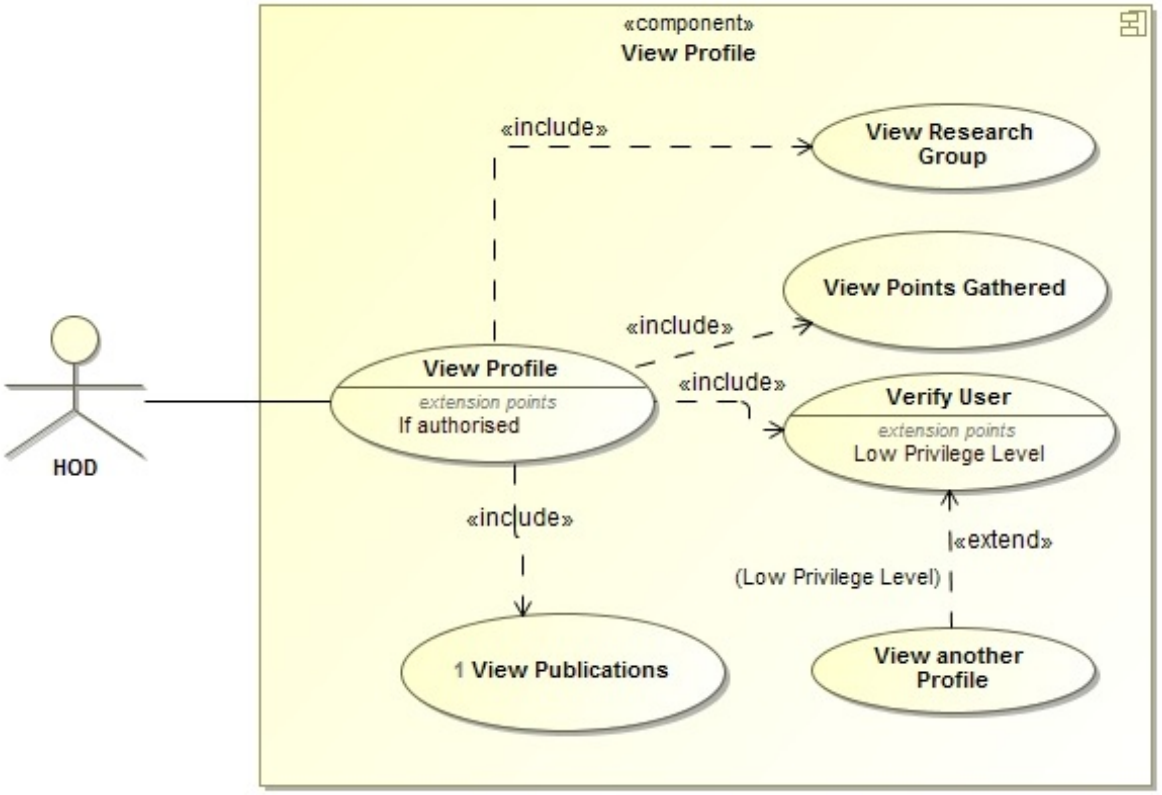
5.3.6 Edit Publication Use Case Diagram



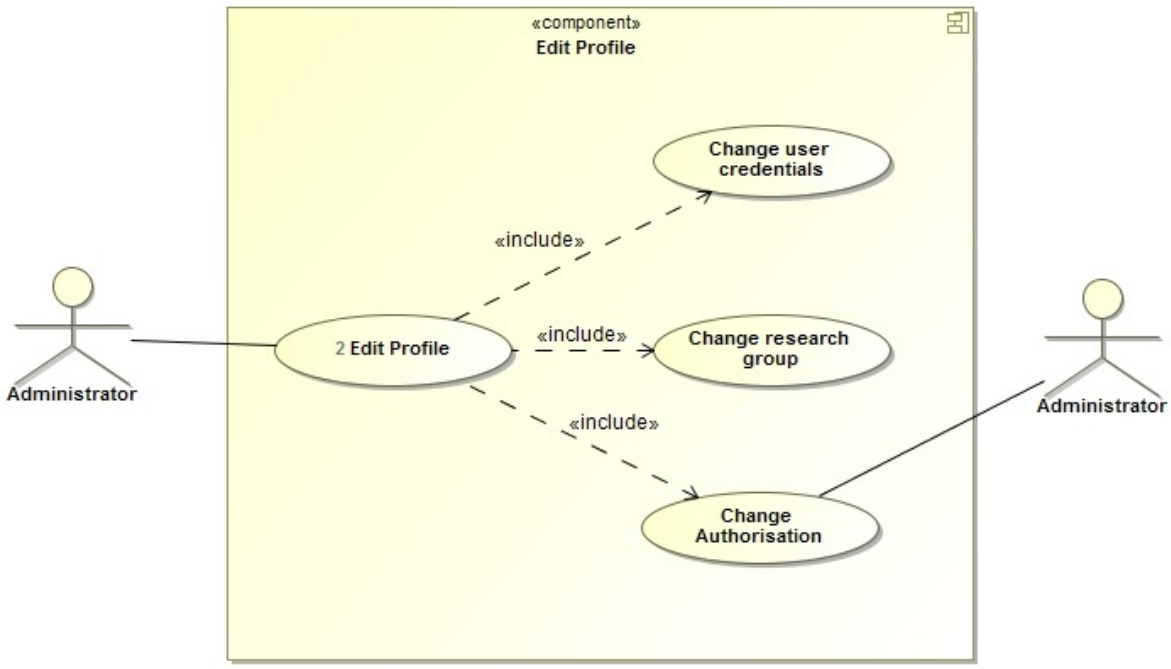
5.3.7 View Publication Use Case Diagram



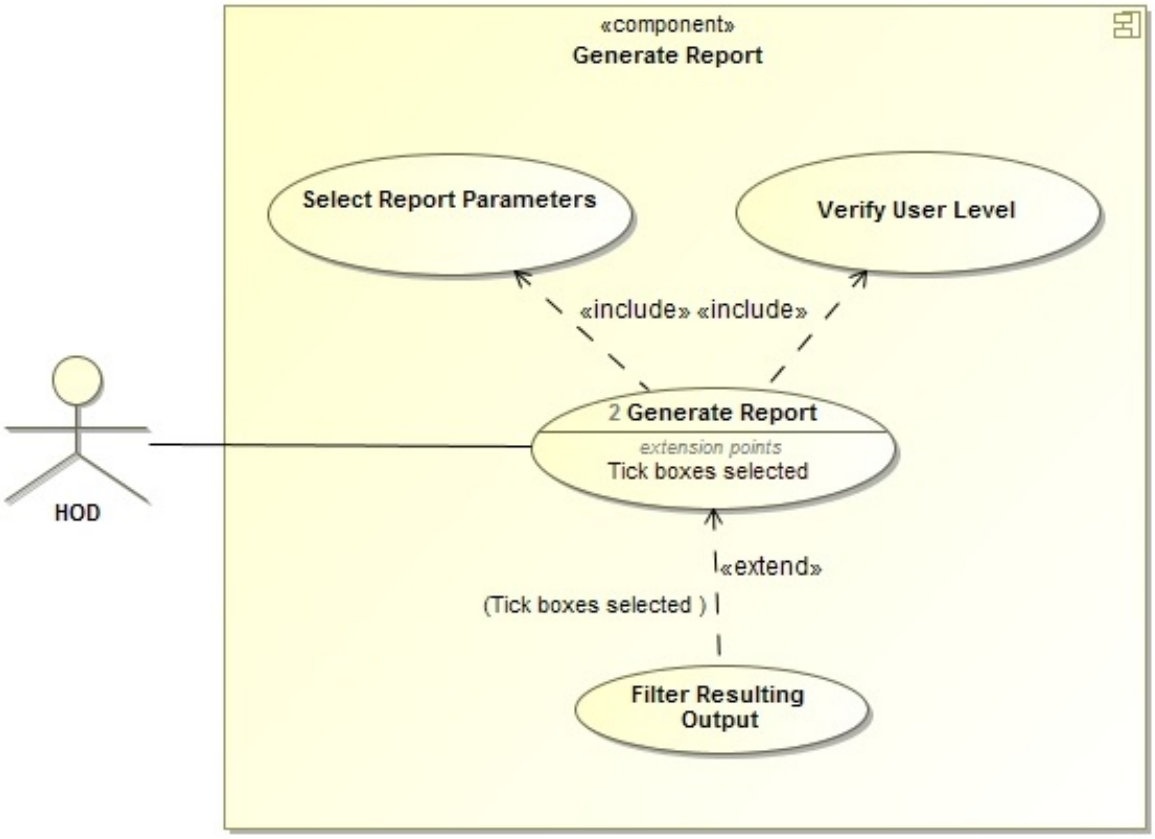
5.3.8 View Profile Use Case Diagram



5.3.9 Edit Profile Use Case Diagram

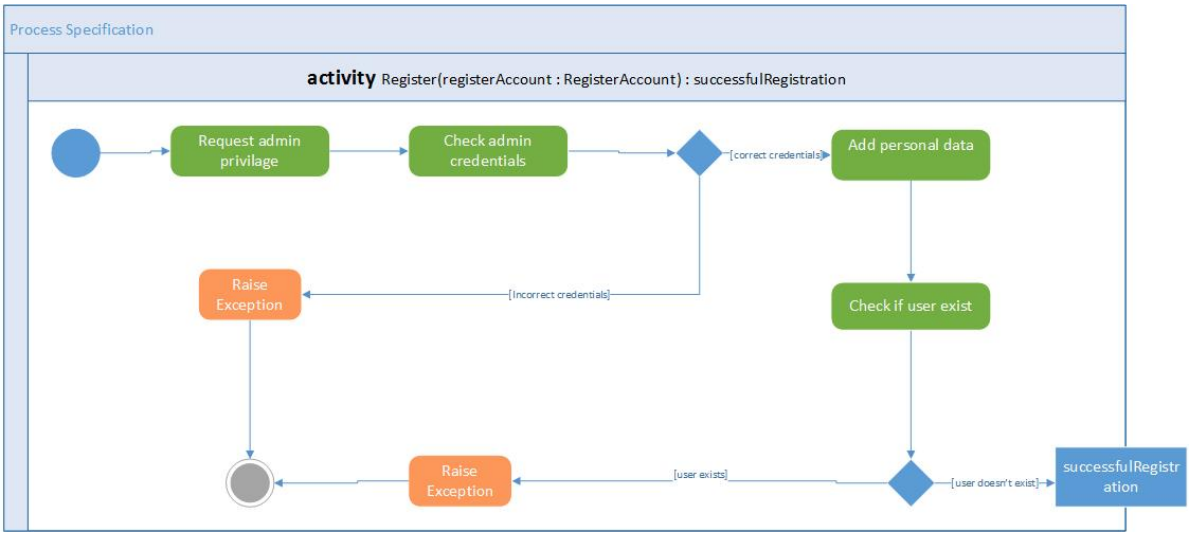


5.3.10 Generate Report Use Case Diagram

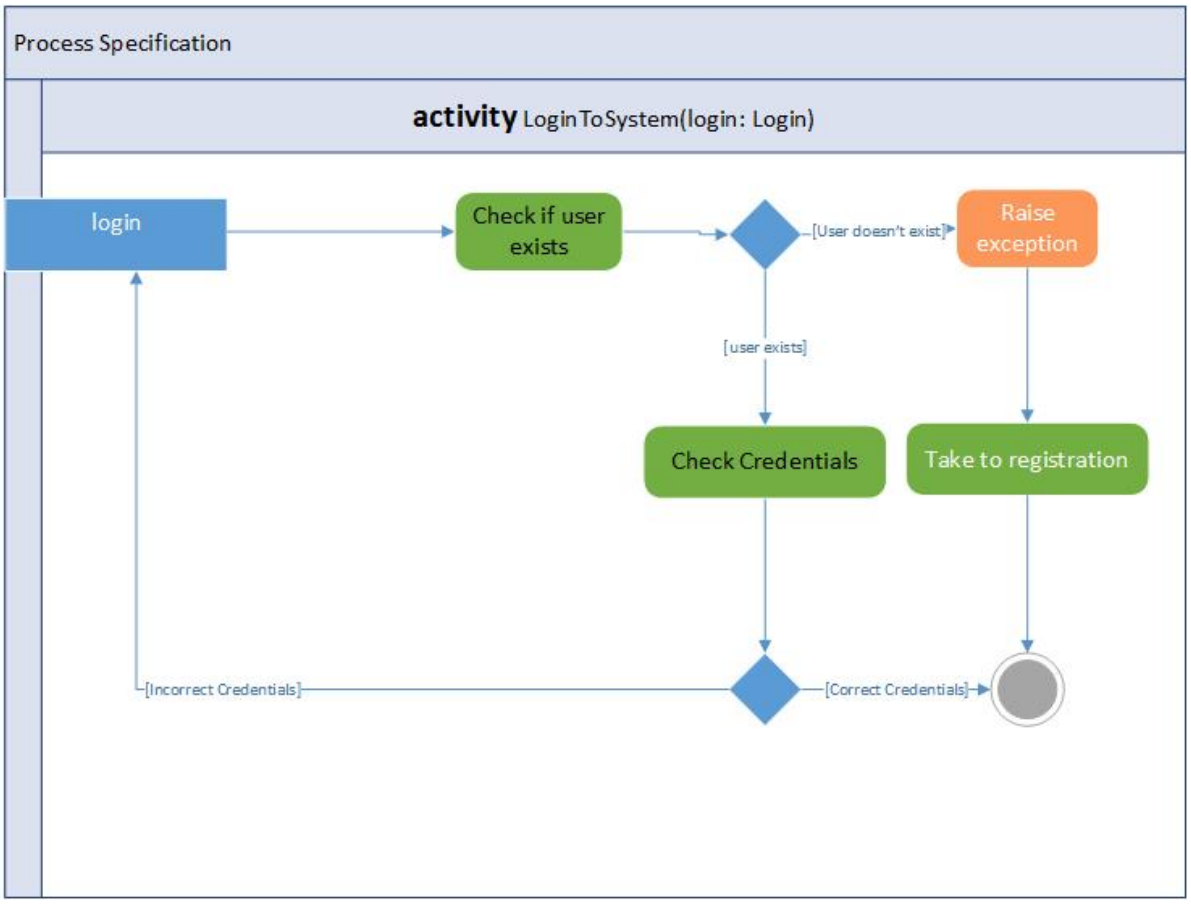


5.4 Process specifications

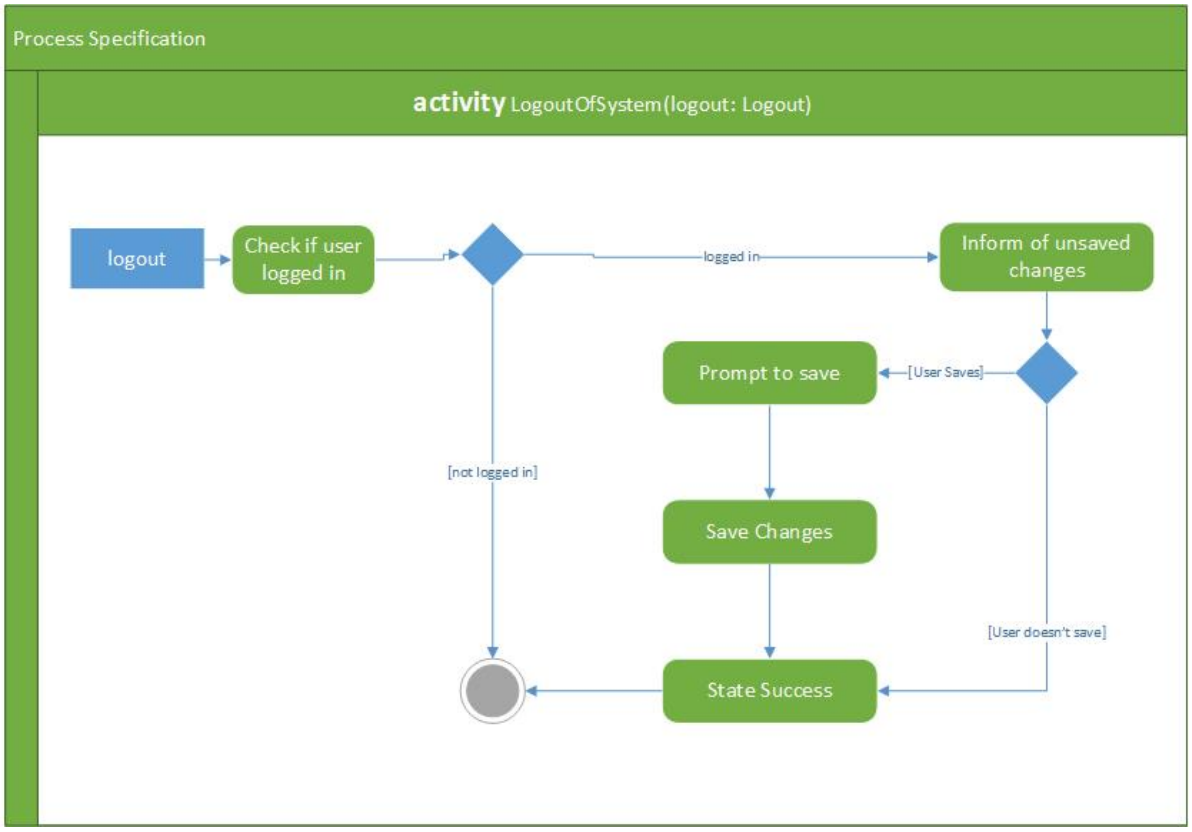
5.4.1 Registration Activity Diagram



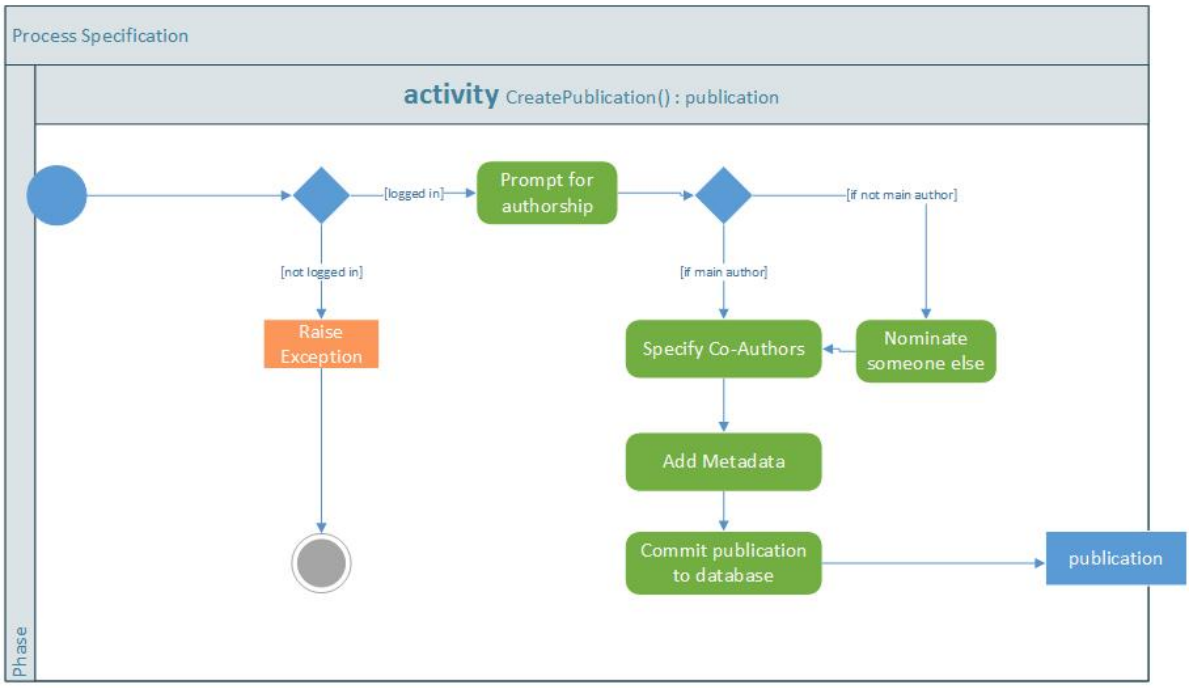
5.4.2 Login Activity Diagram



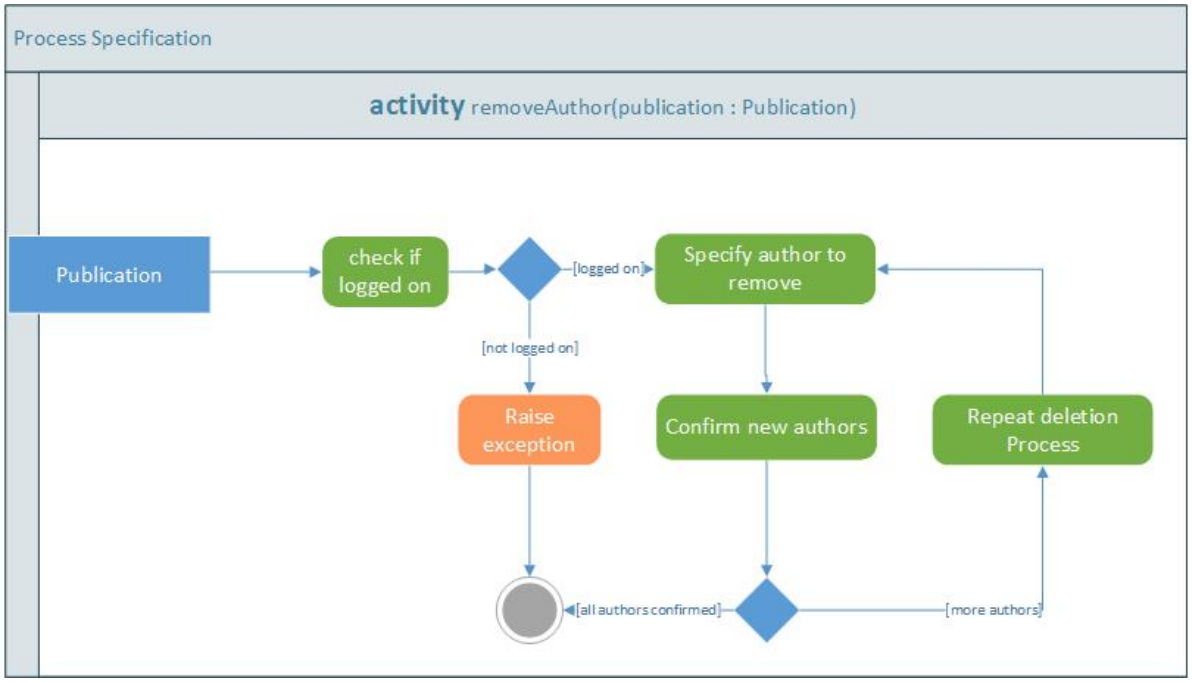
5.4.3 Logout Activity Diagram



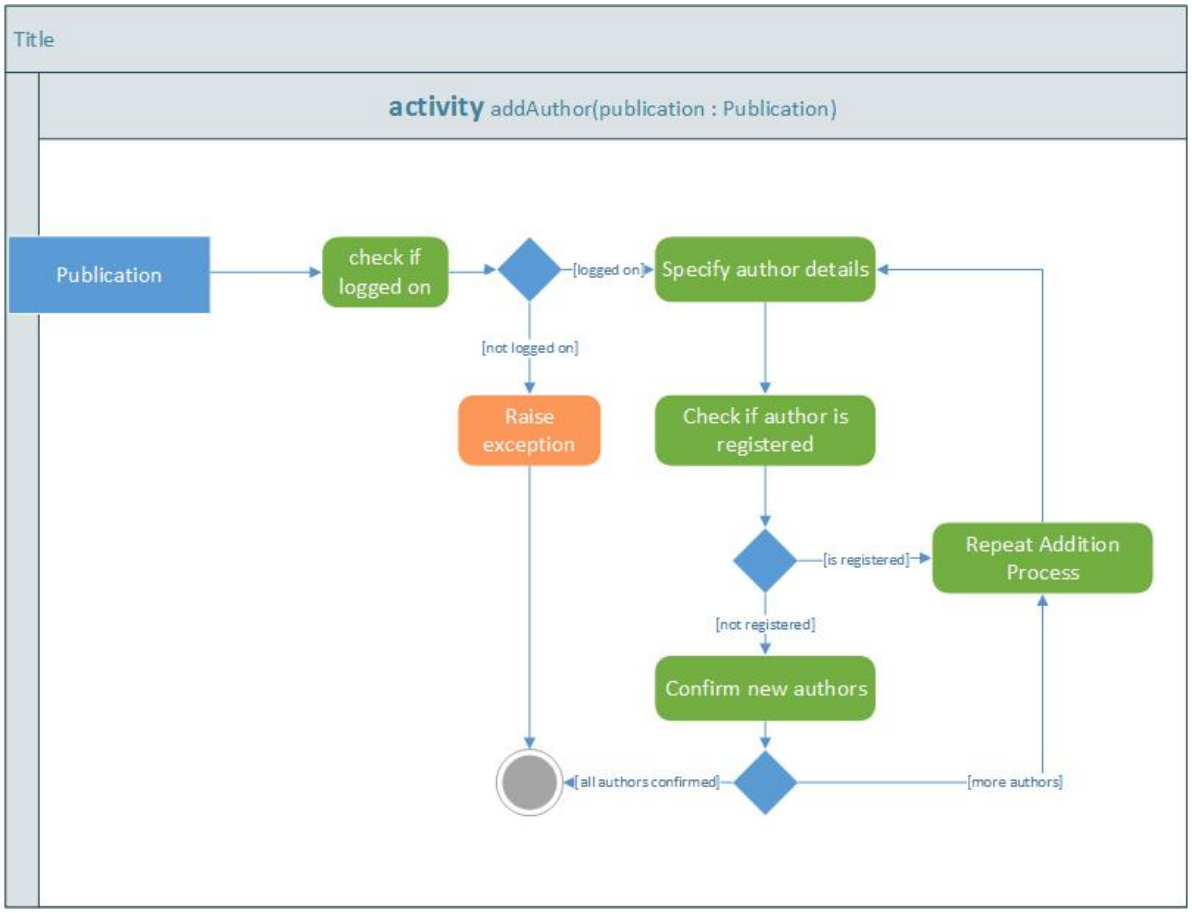
5.4.4 Create Publication Activity Diagram



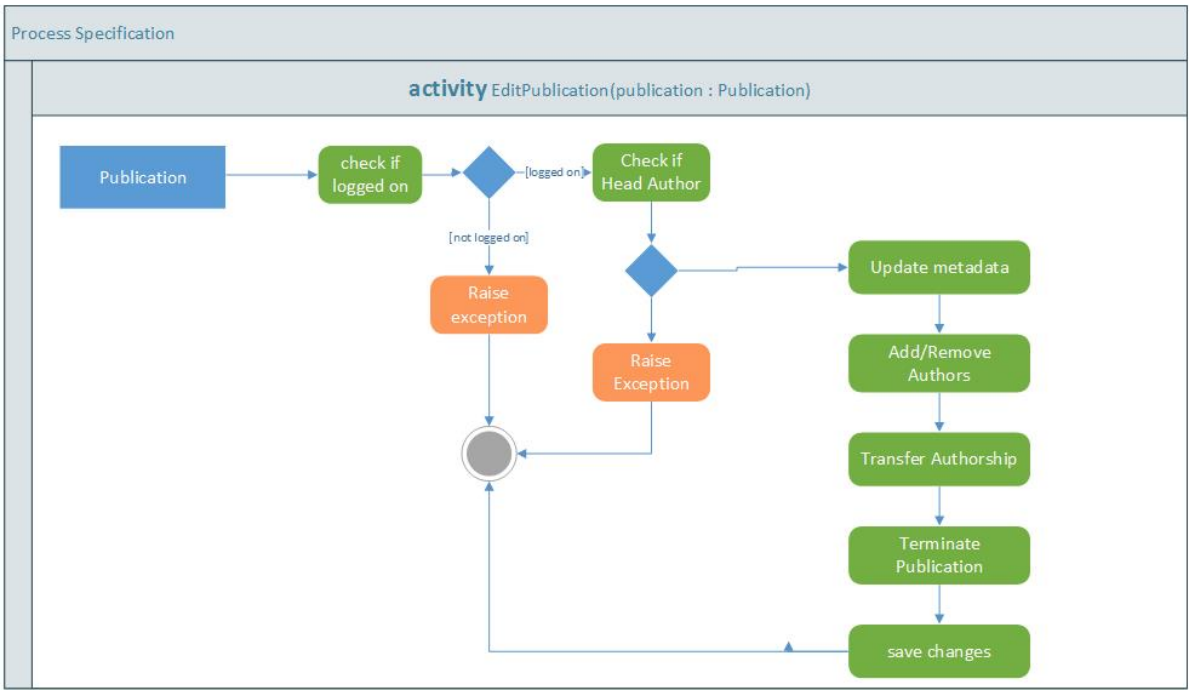
5.4.5 Remove Author Activity Diagram



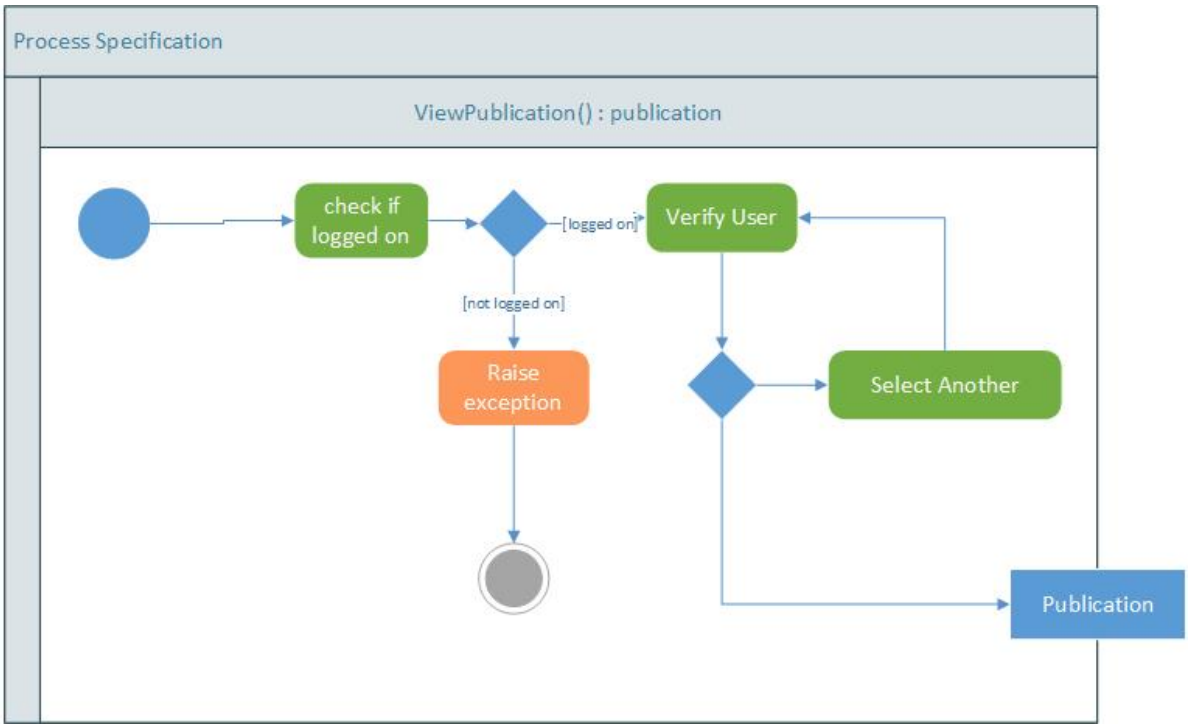
5.4.6 Add Author Activity Diagram



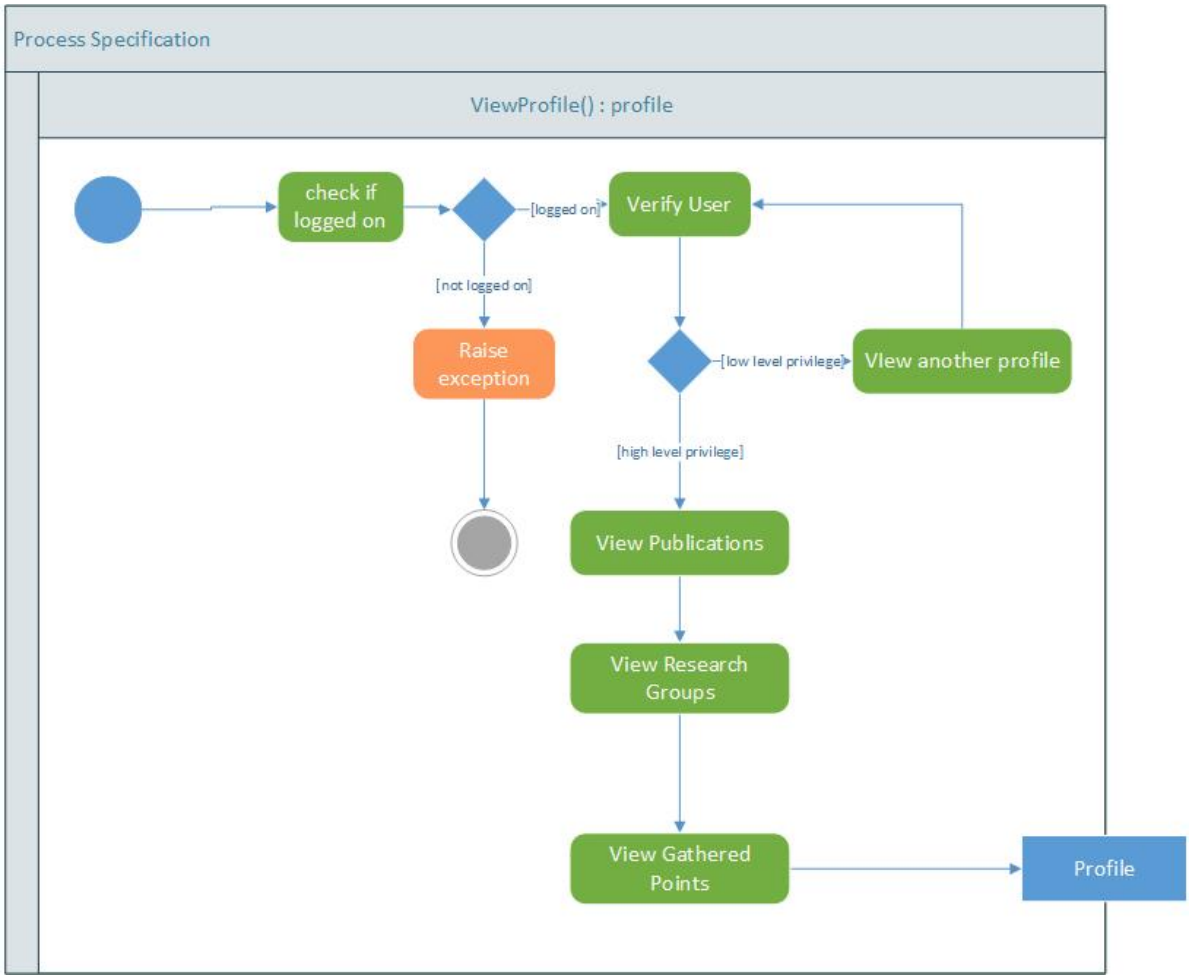
5.4.7 Edit Publication Activity Diagram



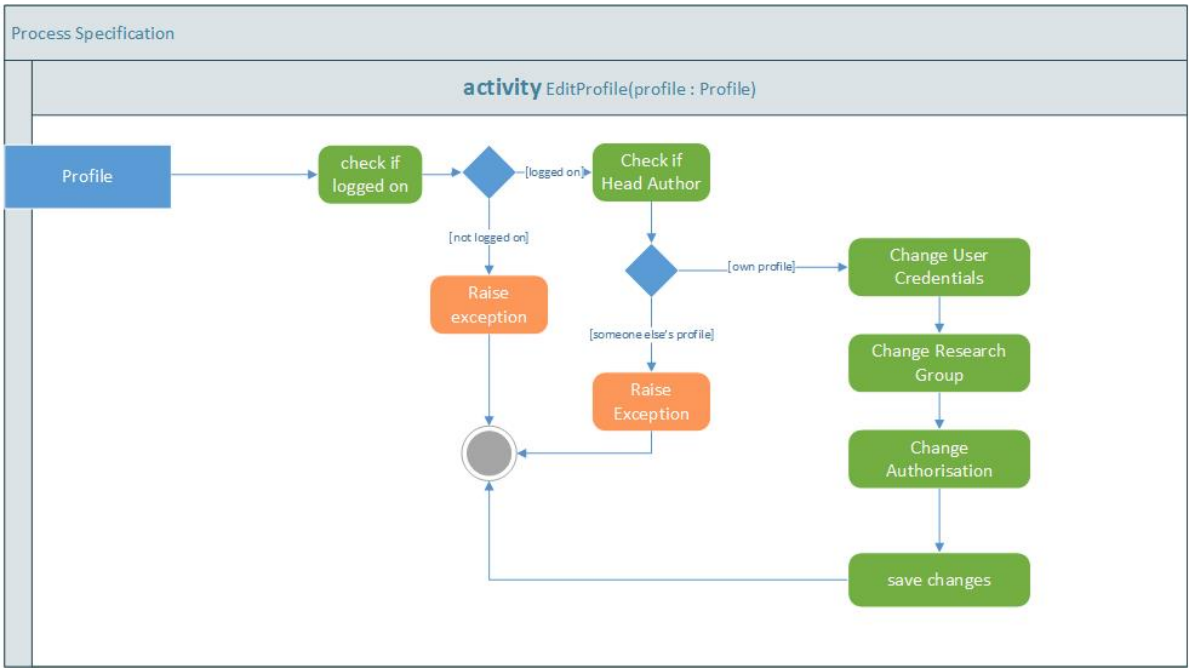
5.4.8 View Publication Activity Diagram



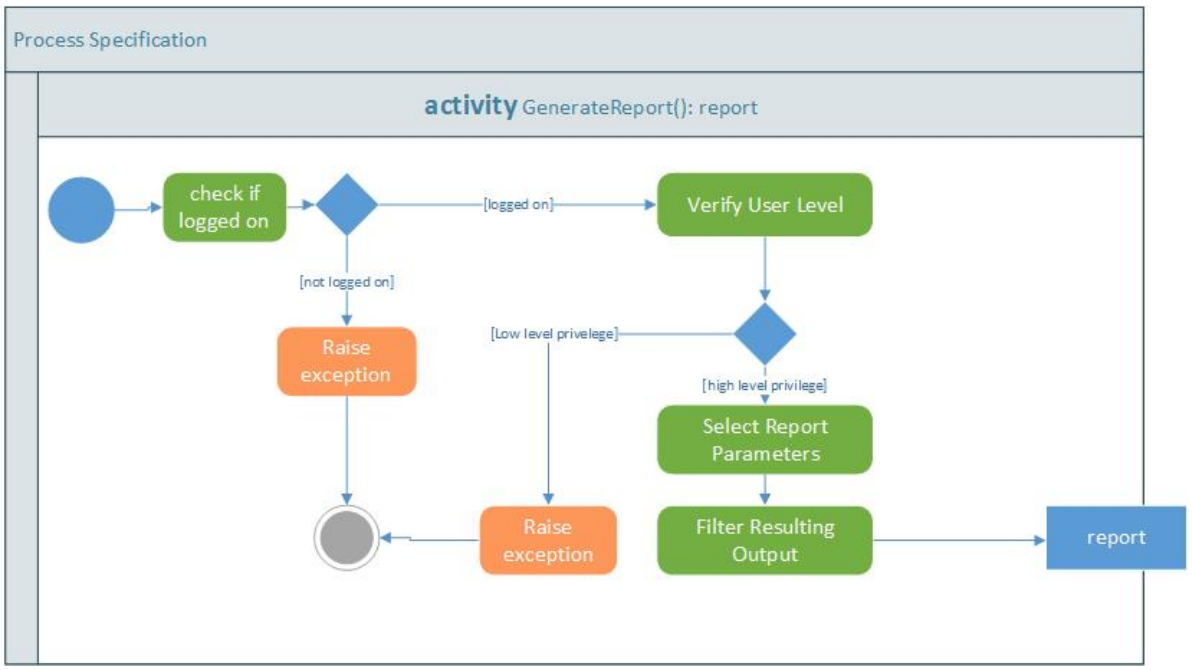
5.4.9 View Profile Activity Diagram



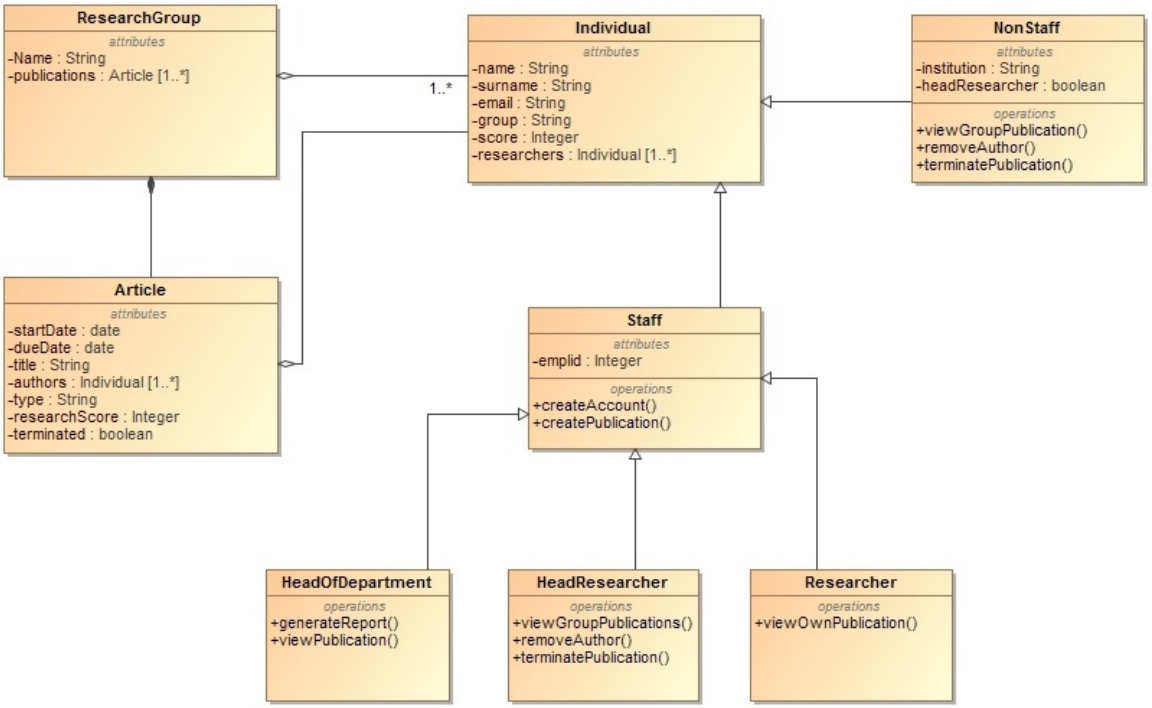
5.4.10 Edit Profile Activity Diagram



5.4.11 Generate Report Activity Diagram



5.5 Domain Model



6 Open Issues

As clarification, the client does not want to store any of the research articles on the system. They only want a link to where the research is being worked on.

The client has expressed an interest in a notification functionality, but has said that one is not nessassary.