Requirement Analysis Document - Yellow Project

William Diedrichsen Marstrand, Dennis Thinh Tan Nguyen,
Jacob Mullit Miniche, Thor Valentin Olesen

September 24, 2015

1 Introduction

1.1 Purpose of the System

Today, the amount of data has grown to a level where it is starting to challenge researchers who need to extract the best available research on a specific question. As a result, researchers apply systematic studies on big data sets where they exploit meta data to classify subsets with useful data. This requires smart data processing tools that can use meta data to narrow down relevant research data. The purpose of such a systematic review is to sum up the best available research on a specific question. This can be achieved by combining the results of several studies. In this regard, the system in this project is comprised of two parts, client and server side. Our system scope is restricted to the server side and shall support the configuration of summarized research data relevant to a given research question.

1.2 Scope of the System

The server shall provide teams with tools to conduct secondary studies (SMS or SLR). It should support activities of planning and conducting a review distributed in a research team. The server shall make sure that all data is stored for use in setting up a study configuration requested by the client. The system should support .bibtex files as a minimum. Security matters (e.g. user authentication) are not taken into primary account due to the scope of the project. In order for the system to fulfill the previously described purpose, it has to support the tasks described in the "Proposed System" section **??**. Among these, the system shall support management of distributed research systems to work on a study. The reviewers should be able to export data sets and filter them with inclusion and exclusion criteria. Finally, the system should allow specific sets of data to be reviewed and screened by specific members of a research team.

1.3 Objectives and Success Criteria of the Project

The system should be easy to deploy and install. It should include an installation and user manual used to describe how to configure and prepare research papers for screening. It should be easy and quick to distribute relevant data and the overview should outcompete the one achieved in other third-party programs such as e.g. Excel. The system should define rules for which data goes to whom to achieve a succesful screening of paper and efficient data extraction. The yellow system has to provide a user interface from which the blue team can extract data based on user roles and rules. The system has succeeded if users in the blue team can query the yellow system for relevant studies and tasks based on a given study configuration. Specifically, the yellow system should collect research and aggregate stacks of research material based on a research question. The blue system should then efficiently be able to extract a subset of primary studies provided from the screening of the search hits in the yellow system. The configured data may then be used by reviewers in the blue system (visualize, sort, export and categorize). Finally, the system should be able to replicate an existing study.

1.4 Definitions, Acronyms, and Abbreviations

- Systematic Studies: methodology used to sum up the best available research on a specific research question or topic.
- Yellow system: server side also referred as "server"
- Blue system: client side also referred as "client"

1.5 References

Tell, Paolo, and Steven Jeuris. Autosys: Supporting Distributed Teams Performing Systematic Studies. 1st ed. Copenhagen: ITU, 2015. Print.

1.6 Overview

The rest of the document will describe the current systems available and the proposed system in our project along with the requirements collected from users, customers and stakeholders. The system requirements are used to generate system models such as potential scenarios and use cases.

2 Current System

The current is a Excel based application which facilitates searches on academic topic. The system does not contain the articles, but data regarding the articles content. The system support multiple users to contribute data and makes it possible to identify if another user is contributing data on the same article. The Data contributed is mostly keywords, which makes it easier to search and compare articles. ¡¡¡¡¡¡¡ HEAD To search for articles, a user must submit keywords, which the articles will be ranked after. Articles are also ranked after the amount of keywords they have in common with other articles.

The current system can work, but become increasingly hard to manage. Therefor a new system which solves the negatives while keeping all of the functionality the system already support is recommended. This can be done with a separation of the data and the user interface, by creating a dedicated database and a dedicated user interface. This will ensure that new functionality will not be limited by the database structure.

The negatives and pros of the current system are described below: ======= To search for articles, a user must submit keywords, which the articles will be ranked after. Articles are also ranked after the amount of keywords they have in common with other articles.

The current system can work, but become increasingly hard to manage. Therefor a new system which solves the negatives while keeping all of the functionality the system already support is recommended. This can be done with a separation of the data and the user interface. This will ensure that new functionality will not be limited by the database structure.

The negatives and pros of the current system are described below: ¿¿¿¿¿¿¿ origin/master

2.1 Pros of current system

The systems pros are as following:

1. Excel licenses are cheap and is widely used by customer base
2. Its simple and easy to augment and implement changes.
3. If the excel files are Read Only, then data the is safe from hostile users.
4. It's easy to implement version control on .csv files.

2.2 Negatives of current system

And the systems Negatives are as following:

1. The data structures can be hard to manage, especially as the system grows in scope and content.
2. The system does not support multiple users editing the same files at the same time.
3. The system interface leaves a lot to be desired.
4. New features becomes hard to implement as the systems grow.

3 Proposed System

3.1 Overview

This section will describe the requirements of the system. In other words, it will clarify the purpose of the system with functional and non-functional requirements. The functional requirements will describe the services that the system should provide and how the system will behave. The non-functional requirements will describe the constraints of these services and are used to measure the quality of these.

3.2 Functional Requirements

The system must support the following functional requirements:

- The user must be able to define the phases of a study.
- The user must be able to create and manage classification criteria for different studies.
- The user must be able to create and manage inclusion and exclusion criteria.
- The user must be able to configure future phases in a study.
- The user must be able to store information about delivered tasks.
- The user must be able to store data about Users and Teams.
- The user must be able to filter research papers based on demographic information specified by the user.
- The user must be able to screen imported research papers. The result should be an export of papers consisting of primary studies (set of relevant papers).
- The user should be able to export studies as plain data sets in different formats, e.g. as comma-separated-values(cvs) format.
- The system should only accept tasks sent from the same user who received them in the first place.
- The system should support a role engine which identifies user rights. Possible roles to be supported could be a "viewer" and "validator" role.
- The system should be able to handle multiple client requests concurrently.
- The system should be able to store quality ratings of research papers based on coverage according to specific research criteria.
- The system should be able to extract data samples from specific studies for further validation by a validator role.
- The system should be able to generate a research protocol that describes the rationale, objectives, design and methodology of the data and configuration of a study.
- The system should restart automatically upon system failure.
- The system must be complemented with an installation manual and a user manual.

3.3 Nonfunctional Requirements

The system must support the following nonfunctional requirements:

3.3.1 Usability

- The Users must be able to configure a study through the use of at most 3 application windows.
- Users with no background in IT should be able to understand the entire set of tools, which the system provides in approximately 30 minutes.
- The user manual must provide knowledge about the interfaces provided by the system.

3.3.2 Reliability

- System restart upon system failure should be done within 30 seconds.
- The system should strive to always be available.
- The latest version of the system should always be available through a back up

3.3.3 Performance

- The system should identify the role of a user and load the appropriate studies within 15 seconds
- A user request for specific data should be acknowledged no more than 10 seconds after it is received
- The time before a response is send should be no more than 30 seconds after the request is received
- The system should support at least 200 users

3.3.4 Supportability

- The server is maintained by the supplier of the system
- The system must be fully functional independent of the other subpart of the complete system (the blue component)

3.3.5 Implementation

- The system should run flawlessly on any Windows operating system newer than Windows XP

3.3.6 Interface

- The system user interface for study configuration must at any time be replaceable by a newer version

3.3.7 Packaging

- The set up and installation of the server is done by the provider of the system independent of the client
- An employee with no prior knowledge about the system must be able to install and set up the server within a period of 15 min

3.3.8 Legal

- The system should be publishable as Open Source Software in accordance to the GNU General Public License v2.0
- The system should store data on a MySQL server provided by the IT University of Copenhagen following the rules issued by the institution

4 Scenarios

This section containts various scenarios regarding operations between the user and the system.

Scenario name	bobStartsNewStudyConfiguration
Participating actor instances	bob:`Researcher` System:`AutoSys System` Configuration UI:`StudyConfigurationUI`
Flow of events	1. Bob the `Researcher` has to start a new research and opens the client from `StudyConfigurationUI`. 2. Bob logs into the client and navigates to the "Study Configuration" page. 3. Bob specify two reviewer and one validator and defines a research question based on some inclusion- and exclusion criteria to specify what papers should be returned. 4. Bob confirms his study configuration and sends the request to the server by pressing "ok". 5. The `AutoSys System` returns an overview of the study configuration to the client.

Table 1: Scenario when a user creates a new study configuration

Scenario name	ClientFilteringOperation
Participating actor instances	server:`Study Configuration Server` client:`Study Review UI`
Flow of events	1. `Study Configuration Server` is receiving a request from `Study Review UI`: Filtering keywords Design pattern, 2005, A gang of four 2. `Study Configuration Server` validates clients authentication. User credential is accepted. 3. `Study Configuration Server` measures all studies based on the given keywords. 4. `Study Configuration Server` finds 20 studies and a list is formed. Each study element in the list contains data about it. 5. `Study Configuration Server` replies to the clients request by sending the article list of found articles. 6. `Study Configuration Server` returns an overview of the `Study Configuration` to the `Study Review UI`,

Table 2: Scenario when a user sends a request with given filtering keywords trough the `Study Review UI`.

Scenario name	ClientGetsToManyRelevantPapers
Participating actor instances	server:`Study Configuration Server` client:`Study Review UI`
Flow of events	1. `Study Configuration Server` is receiving a request from `Study Review UI`: Search keywords 2001,2002,2003,2003,2004,2005,2005,2006,,2007 2. `Study Configuration Server` validates `User` authentication. `User` credential is accepted 3. `Study Configuration Server` measures all articles based on the keywords. 4. The list containing papears exceeds 10.000 papers, and the exception ToManyHitsException was thrown. 5. A response is sent to the `Study Review UI` that there in fact was too many papers returned.

Table 3: Scenario when a user has requested to many papers during one request.

Scenario name	ExcludingPapersAboutDesignPatterns
Participating actor instances	server:`Study Configuration Server` client:`Study Review UI`
Flow of events	1. `Study Configuration Server` is receiving a request from `Study Review UI`: Search keywords Design pattern, 2005, A gang of four, ExcludingAllNonAssignedArticles. 2. `Study Configuration Server` validates `User` authentication. `User` credential is accepted. 3. `Study Configuration Server` measures all articles based on the keywords. 4. 10 relevant articles was found but 5 was excluded because of the exclusion criteria. 5. A list with the remaining 5 articles is returned from `Study Configuration Server` to the `Study Review UI`

Table 4: Scenario when a user wants to exclude some papers.

Scenario name	ClientRequestWithInvalidUser
Participating actor instances	server:`Study Configuration Server` client:`Study Review UI`
Flow of events	1. `Study Configuration Server` is receiving a request from `Study Review UI` on task retrieval for a `User` named "bob" which is an invalid `User`. 2. `Study Configuration Server` validates `User` authentication. 3. The given `User` is not valid because it does not exist in the database. 4. A response is sent to the `Study Review UI` detailing why bob does not have access to the `AutoSys System`.

Table 5: Scenario when a invalid user is trying to get access to the server.

5 Use Case

Use case name	`ExportProtocol`
Participating actors	Initiated by `Researcher` Communicates with `Study Configuration Server` through `SystematicReviewClient`
Flow of events	1. The `Researcher` logs into the program using the `SystematicReviewClient` login UI. 2. The `Researcher` uses the `SystematicReviewClient` options for choosing a specific study. 3. The `SystematicReviewClient` presents different possibilities for getting data from the `Study Configuration Server`. 4. The `Researcher` chooses to export a research protocol from the study. 5. The `Study Configuration Server` receives the request and makes a validation of the `Researcher` user account(use case: `ValidateUser`). 6. The `Study Configuration Server` exports the requested research protocol to the `Researcher`. 7. The `SystematicReviewClient` visualizes the research protocol to the `Researcher`.
Entry condition	• The `Researcher` logs into the `SystematicReviewClient`.
Exit conditions	• The `SystematicReviewClient` has received an acknowledgment and the selected response from the `Study Configuration Server` and presented it for the `Researcher`, OR • The `SystematicReviewClient` has received an explanation indicating why the transaction could not be processed and presented it for the `Researcher`.
Quality requirements	• The `Researcher`'s request is acknowledged within 10 seconds. • The selected response to the `SystematicReviewClient` arrives no later than 40 seconds after the request was received by the `Study Configuration Server`.

Table 6: Use case: ExportProtocol

Use case name	ManageStudy
Participating actors	Initiated by Researcher Communicates with SystematicReviewClient
Flow of events	1. The Researcher opens SystematicReviewClient, navigates to the Study Configuration Server page. 2. The Researcher invokes the ManageRole use case to define Role for each User that are appart of the StudyConfiguration. 3. The Researcher invokes the ManageClassificationCriteria use case to define classification criterias for the StudyConfiguration. 4. The Researcher invokes the ManageInclusionAndExclusionCriteria use case to define what to exclude or include to the StudyConfiguration. 5. The Researcher invokes the ManageTask use case to specify which Task a User should have. 6. The Researcher reviews all changes made and submits the StudyConfiguration to the Study Configuration Server. 7. The Study Configuration Server invokes the ValidateUser use case to validiate access rights for the User of the Request 8. The Study Configuration Server can either accept or reject the User. If User is rejected an ErrorMessage regarding this will be returned to the StudyConfigurationUI. If the User is accepted the Request will proceed. 9. The Study Configuration Server invoke the StoreUserData use case to update the Role for each User which was specified in the StudyConfigurationUI. The Study Configuration Server will also invoke the StoreStudyData, StoreTaskData use cases to save the study and the tasks for each User. 10. The Study Configuration Server returns an overview of the StudyConfiguration to the StudyConfigurationUI, which the Researcher can use to review if everything has been set up right.
Entry condition	• The Researcher has a new study to create.
Exit conditions	• The Researcher has received a StudyConfiguration overview ,OR • The Researcher has received an explanation indicating why the StudyConfiguration could not be created.
Quality requirements	• The Researcher receives the StudyConfiguration within 10 seconds. • The Researcher receives an ErrorMessage within 5 seconds if the Request is rejected.

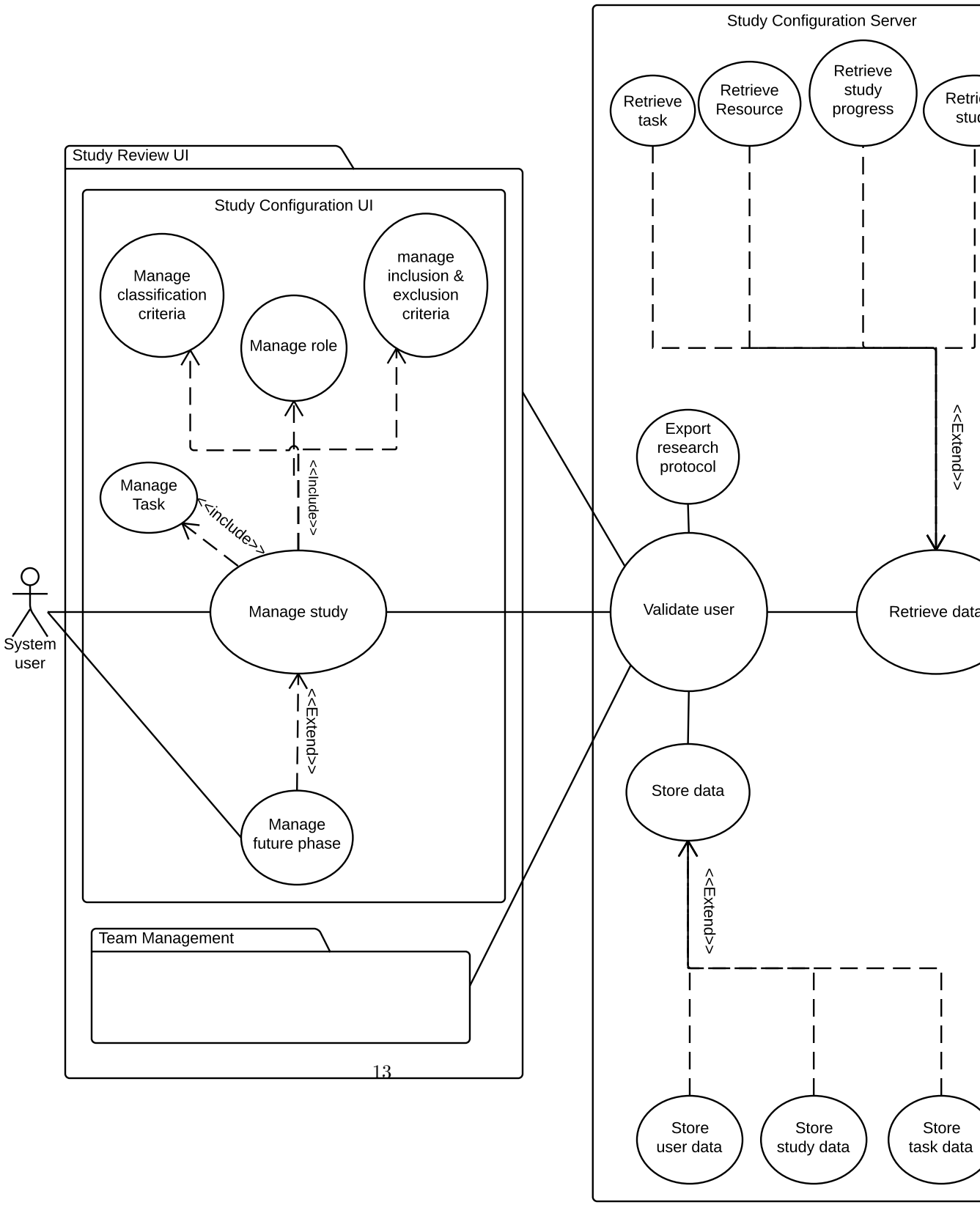
Table 7: Use case when a Researcher wants to create a new study through the StudyConfigurationUI

Use case name	`RetrieveStudyInformation`
Participating actors	Initiated by `SystematicReviewClient` Communicates with `Study Configuration Server`
Flow of events	1. He/she activates the Manage Study module 2. The `SystematicReviewClient` chooses to upload task data about a completed task. 3. The `SystematicReviewClient` specifies the study information he/she wishes to retrieve item The `SystematicReviewClient` confirms he/she wants to retrieve details regarding the Study 4. The message is sent to the server where the senders credentials are checked in the database. If he has access, then precede otherwise a statement I sent to the `SystematicReviewClient` detailing why he was rejected access. • There exist an entry which matches the search criteria, then continue • There is no entry which matches the search criteria and an error message will be sent to the `SystematicReviewClient`.
Entry condition	• The `SystematicReviewClient` is logged in and connected to the internet.
Exit conditions	• The `SystematicReviewClient` has received an data entry detailing a Study or • The `SystematicReviewClient` has received an error message.
Quality requirements	• From the `SystematicReviewClient` has confirmed he wants to retrieve the Study data, he should wait no more than 30 seconds (Given he/she is connected to a stable internet with a reasonable transfer time)

Table 8: Use case: RetrieveStudyInformation

Use case name	StoreTaskData
Participating actors	Initiated by Researcher Communicates with Study Configuration Server through SystematicReviewClient
Flow of events	1. The Researcher logs into the program using the SystematicReviewClient login UI. 2. The Researcher uses the SystematicReviewClient options for choosing a specific study. 3. The SystematicReviewClient presents different possibilities for interacting with the Study Configuration Server. One of which is to deliver a completed task. 4. The Researcher chooses to upload task data about a completed task. 5. The Study Configuration Server receives the request and makes a validation of the Researcher user account(use case: ValidateUser). The validation goes through because the Researcher uploading the completed task is the same as the one who received it in the first place. 6. The Study Configuration Server stores the data from the completed task in the database. Then it sends back a respond to the SystematicReviewClient letting it know, that the request was received and acknowledged. 7. The SystematicReviewClient receives the respond from the Study Configuration Server and responds to the Researcher.
Entry condition	• The Researcher logs into the SystematicReviewClient.
Exit conditions	• The SystematicReviewClient has received an acknowledgment and the selected response from the Study Configuration Server and presented it for the Researcher, OR • The SystematicReviewClient has received an explanation indicating why the transaction could not be processed and presented it for the Researcher.
Quality requirements	• The Researcher's request is acknowledged within 10 seconds. • The selected response to the SystematicReviewClient arrives no later than 40 seconds after the request was received by the Study Configuration Server.

Table 9: Use case: StoreTaskData


Study Configuration Server
Retrieve task
Retrieve Resource
Retrieve study progress
Retri stud
Study Review UI
Study Configuration UI
Manage classification criteria
Manage role
manage inclusion & exclusion criteria
Manage Task
<<include>>
<<include>>
Export research protocol
<<Extend>>
System user
Manage study
Validate user
Retrieve data
<<Extend>>
Manage future phase
Store data
<<Extend>>
Team Management
Store user data
Store study data
Store task data

6 Glossary

- Systematic Review Client: Is the client part of the system which broadly provides the User Interface for requesting data about papers, reviewing papers, and validating papers.
- Study Configuration Server: Is the server (including the database) where all data is stored, and user requests regarding papers are handled.
- Study Configuration UI: Is the User Interface supporting only work tasks regarding the configuration of studies e.g. study tasks, study phases etc.

7 Template

8 Scenario Template

Table 10 shows an example scenario taken from the OOSE book on page 127. It shows how to format a scenario using LaTeX. We recommend using the package 'fullpage' so that more space is available within the flow of events. For the formatting of actors, analysis objects, and use cases we recommend defining new commands, as shown in this template, so you can easily reference and change them from a central location. Also note how '\ref{}' is used to reference the table from within this paragraph.

Scenario name	warehouseOnFire
Participating actor instances	bob, alice:`FieldOfficer` john:`Dispatcher`
Flow of events	1. Bob, driving down main street in his patrol car, notices smoke coming out of a warehouse. His partner, Alice, activates the "Report Emergency" function from her FRIEND laptop. 2. Alice enters the address of the building, a brief description of its location (i.e., northwest corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene, given that the area appears to be relatively busy. She confirms her input and waits for an acknowledgment. 3. John, the `Dispatcher`, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the `Incident` site and sends their estimated arrival time (ETA) to Alice. 4. Alice receives the acknowledgment and the ETA.

Table 10: Example of a scenario. (p127 in OOSE)

9 Use Case Template

Table 11 demonstrates the more complicated formatting required to create use case descriptions. The example is taken from the OOSE book on page 129. Notice that system responses are indented to easily distinguish them from actor actions. This template should get you going, but we might provide you with one requiring less boiler plate code later on.

Use case name	`ReportEmergency`
Participating actors	Initiated by `FieldOfficer` Communicates with `Dispatcher`
Flow of events	1. The `FieldOfficer` activates the "Report Emergency" function of her terminal. 2. FRIEND responds by presenting a form to the `FieldOfficer`. 3. The `FieldOfficer` completes the form by selecting the emergency level, type, location, and brief description of the situation. The `FieldOfficer` also describes possible responses to the emergency situation. Once the form is completed, the `FieldOfficer` submits the form. 4. FRIEND receives the form and notifies the `Dispatcher`. 5. The `Dispatcher` reviews the submitted information and creates an `Incident` in the database by invoking the `OpenIncident` use case. The `Dispatcher` selects a response and acknowledges the report. 6. FRIEND displays the acknowledgment and the selected response to the `FieldOfficer`.
Entry condition	• The `FieldOfficer` is logged into FRIEND.
Exit conditions	• The `FieldOfficer` has received an acknowledgment and the selected response from the `Dispatcher`, OR • The `FieldOfficer` has received an explanation indicating why the transaction could not be processed.
Quality requirements	• The `FieldOfficer`'s report is acknowledged within 30 seconds. • The selected response arrives no later than 30 seconds after it is sent by the `Dispatcher`.

Table 11: Example of a use case. (p129 in OOSE)