Inf2-SEPP 2024-25

Coursework 1

Capturing requirements for a self-service portal

SAMPLE SOLUTION

This courseword is addressed as if in a group of 4.

Task 1: Ambiguities and addressing them

The solutions to ambiguities from this task are consistent with the rest of the sample answers, and the sample answers don't contain any additional assumptions that are not discussed first as ambiguities in this task. The lab demonstrators may have answered differently and/or you may have interpreted their answers differently and/or they may have answered 'I don't know', or you may have not conducted an interview with them and made different assumptions. These are all fine as long as you provide justifications.

Some ambiguities related to the rest of the solutions to this coursework (there are many more!) and possible solutions to them are as follows (please note that here we are only providing the ambiguity and solutions to them and not using the full format of the task, for simplicity):

- Ambiguity: Does logging out use the student record system?
  Assumption: No, logging out is only taking place on our system and our system will not use the external record system for this purpose.
- Ambiguity: Can guests assign a course to their inquiry to members of staff?
  Assumption: Yes, they can, the same as students.
- Ambiguity: Can guests browse the FAQ using a course tag?

Assumption: Yes, they can, the same as students.

- Ambiguity: Can admin and teaching staff browse the FAQ using a course tag?
  Assumption: No, they cannot. Only students and guests can do this.
- Ambiguity: How do admin staff remove an FAQ question-answer pair?
  Assumption: This functionality is offered to them while browsing the FAQ topic hierarchy, on any level in this hierarchy. They need to provide the number of the question that is to be removed (numbers should be added to quesions to identify them). The system then looks for the question and, if it finds it, it removes it and its answer. If the question was the last question in a topic, the system removes that entire topic, which would mean bringing any subopics up one level.
- Ambiguity: What information is required to add a course?
  Assumption: The course's name, course code (unique identifier), description, requiring computers [yes/no,], course organiser (name, email address), course secretary (name, email address), all course activities (lectures, and/or tutorials, and/or labs) each with relevant fields, some of which are common (start time, end time, start date, end date, location, frequency, number of tutorials students must take weekly- 0 or more, number of labs students must take weekly- 0 or more) and some separate (lectures have recording on/off, tutorials and labs have a capacity).
- Ambiguity: How do admin staff add courses?
  Assumption: One at a time. They need to provide the relevant information (see above). An admin member cannot add a course with a code that already exists in the system - the system gives an error. It also gives a warning in case any labs without computers included for a course requiring computers. After successful addition, an email is sent to the Course Organiser.
- Ambiguity: How do admin staff remove courses?
  Assumption: One at a time. They need to provide the course code. The course is then removed from the system and from student timetables. The course organiser and any students that had the course on their timetable are then emailed, notifying them of the removal.
- Are guests allowed to view courses, add them to their timetable, or add tutorials or labs to the timetable?
  Assumption: They are allowed to view courses, and the details of an individual course. They are not allowed to add courses, tutorials or labs to their timetable, these are only available to students.
- Can students view their timetable, or does this happen only when adding a course, tutorial or lab for it?
  Assumption: They can view their timetable. This will include the system showing any errors and warnings present in this timetable.
- Considering that students will be able to use the self-service portal to oganise their time, shouldn't they also be able to remove courses from their timetable?

Assumption: Yes, students will be able to also remove courses from their timetable, which removes all of their activities.

- How can students remove courses from their timetable?
  Assumption: By choosing the option and indicating the course code for the course to remove.
- Ambiguity: What information is displayed when viewing all courses?
  Assumption: The course names and codes.
- Ambiguity: Does a student viewing all courses only involve the courses that the student can take that year?
  Assumption: No, all courses in the school.
- Ambiguity: What does it mean viewing a course?
  Assumption: Seeing information about the course in text, including its timetable.
- Ambiguiy: Can admin staff view courses and details about an individual course?
  Assumption: Yes, they can.
- Ambiguity: Can teaching staff also view the list of courses in the system, and details about an individual course?
  Assumption: Yes.
- Ambiguity: Can course organisers redirect inquiries?
  Assumption: No, only admin staff can do this.

Task 2: Use case diagram

A use-case diagram is shown in Fig. 1.

The Inquirer actor from the old system was changed to a more general Inquirer/Viewer actor because users within the same role would be able to act as an inquirer but also view courses and course details. It is acceptable to also keep the Inquirer and for it to be a subactor to e.g. CourseViewer or similar. Here we considered as primary actors only those who will be able to accomplish the goals of the use cases, e.g. an inquirer/viewer could be allowed to try to log in but they would not be successful unless they were also a student or member of staff, so we considered only Student and MembersOfStaff as subactors of AuthenticatingUser. However, solutions where the Inquirer/Viewer (or similar names) can log in will also be accepted as it makes sense for them to have access to this functionality. For logging out, they shouldn't though.

Variations to this use case diagram which are motivated by answers received during the lab demonstrator interview, or assumptions made, are acceptable as long as these were clearly described in Task 2 (Ambiguities).

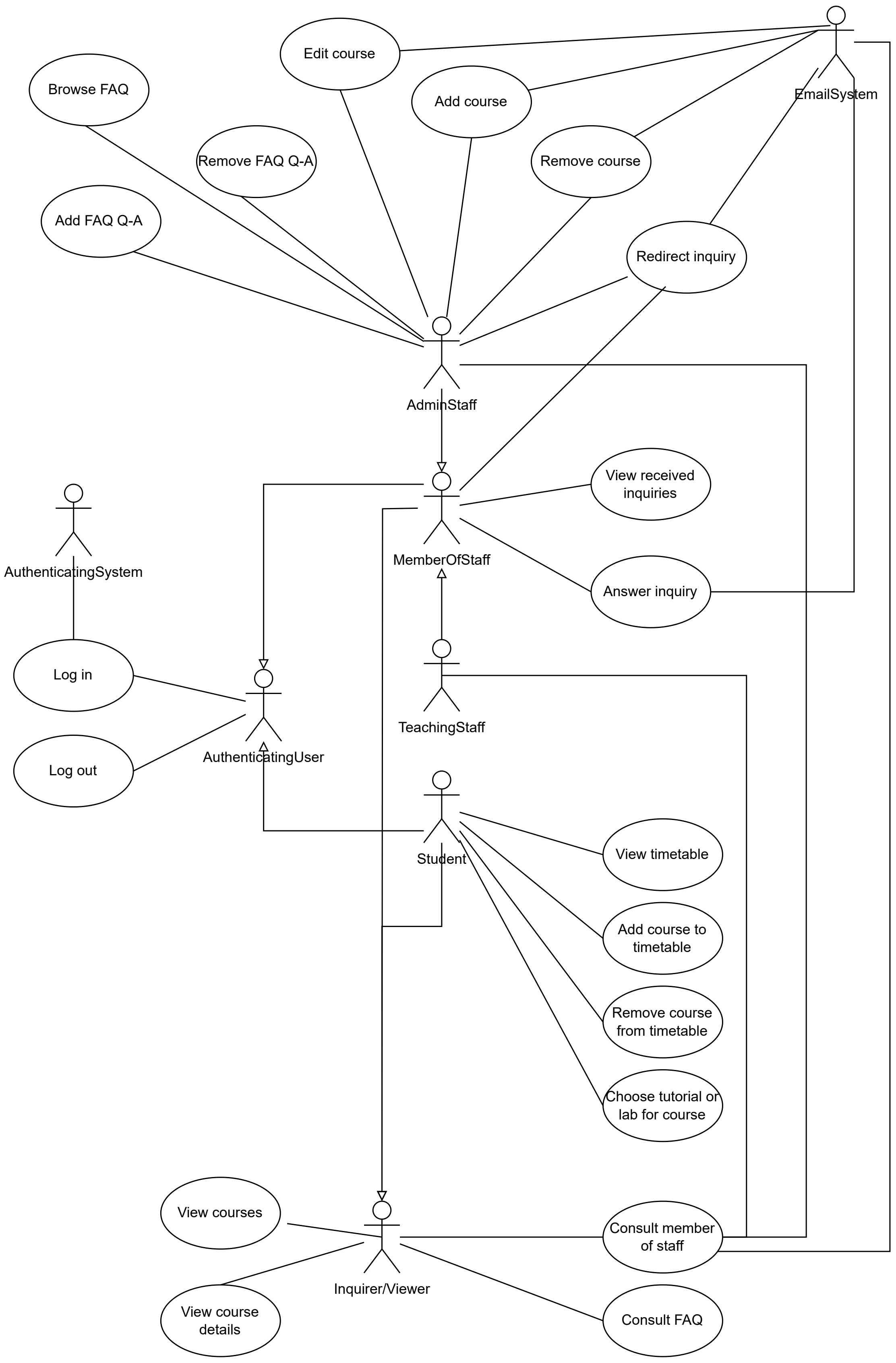

Figure 1: UML Use Case Diagram

Task 3: Use case descriptions

Below are descriptions of the new use cases and of the use cases that were modified as compared to their functionality in the old system:

4.1 Add course [New]

Primary actor: AdminStaff
Supporting actors: EmailSystem
Summary: A member of admin staff adds a new course to the system, including all relevant information of that course.
Precondition: Admin staff is logged in (reference 'Log in' use case).
Trigger: Admin staff requests to add a new course to the system.
Success Guarantee: The course is added to the system, and students can then add the course to their timetables.
Main Success Scenario:
1. System asks for relevant information: course's name, course code (unique identifier), description, requiring computers [yes/no,], course organiser (name, email address), course secretary (name, email address), all course activities (lectures, and/or tutorials, and/or labs) each with relevant fields, some of which are common (start time, end time, start date, end date, location, frequency, number of tutorials students must take weekly- 0 or more, number of labs students must take weekly- 0 or more) and some separate (lectures have recording on/off, tutorials and labs have a capacity).
2. Admin staff provides all the information in plain-text.
3. The system confirms that the course was added.
4. The system asks the email system to send a confirmation to the course organiser.
Extensions:
3a. The course code is empty, invalid or already used
.1 System gives error to admin staff. Use case terminates.
3b. Other compulsory information was not provided
.1 System gives error to admin staff. Use case terminates.
3c. Labs without computers included and the course requires computers:
.1 System gives warning to admin staff.
.2 Resume MSS at step 3.

4.2 Remove course [New]

Primary actor: AdminStaff
Supporting actors: EmailSystem
Summary: A member of admin staff removes a course from the system.
Precondition: Admin staff is logged in (reference 'Log in' use case).
Trigger: Admin staff requests to remove a course from the system.
Success Guarantee: The course is removed from the sysem and from all students'

timetables.
Main Success Scenario:
1. System asks for course code.
2. Admin staff provides the course code.
3. The system confirms that the course was removed.
4. The system asks the email system to send a confirmation to the course organiser.
5. The system asks the email system to inform students who had the course in their timetable that it was removed.
Extensions:
3a. The course code does not correspond to a course registered on the system
.1 System gives error to admin staff. Use case terminates.
5a. No students had the course in their timetable:
.1 Use case terminates.

4.3 Remove FAQ Q-A [New]

Primary actor: AdminStaff
Supporting actors: N/A
Summary: A member of admin staff removes an FAQ Q-A pair while browsing the FAQ. This leads hierarchy being reorganised in case the Q-A pair is the only one left in the original topic.
Precondition: Admin staff is logged in (reference ’Log in’ use case) and is browsing the FAQ (reference ’Browse FAQ’ use case).
Trigger: Admin staff requests to remove an FAQ Q-A pair.
Success Guarantee: The Q-A pair is removed and hierarchy reorganised as needed.
Main Success Scenario:
1. System asks for Q-A number.
2. Admin staff provides the Q-A number.
3. The system confirms that the Q-A was removed.
Extensions:
3a. Q-A invalid or does not exist:
.1 System gives error to admin staff. Use case terminates.
4. The Q-A was the last in the topic:
.1 System shows any subtopic(s) as new topic.

4.4 Remove course from timetable [New]

Primary actor: Student
Supporting actor: N/A
Summary: The student removes a course from their timetable, or gets an error
Precondition: Student is logged in (reference ’Log in’ use case)
Trigger: Student requests to remove a course from their timetable.

Success Guarantee: The course is removed from the student's timetable.
Main Success Scenario:
1. System requires course code.
2. Student provides it.
3. System confirms removal from the timetable and provides the timetable with the course activities not in it.
Extensions:
3a. The course code does not correspond to a course registered on the system
.1 System gives error to student. Use case terminates.
3b. The course code does not correspond to a course that a student has in their timetable
.1 System gives error to student. Use case terminates.

4.5 View courses [New]

Primary actor: Guest
Supporting actors: N/A
Summary: The guest chooses the option to view the whole list of courses on the system. If there are no courses on the system, the system notifies the user of this fact and the use case terminates. Otherwise, the system provides all course names and codes.

4.6 View course details [New]

Primary actor: Guest
Supporting actors: N/A
Summary: The guest requests to view details of a course. They are required to provide the course code, and an error is issued if it does not match any courses. If it is correct, the system returns plain-text information about the course.

4.7 View timetable [New]

Primary actor: Student
Supporting actors: N/A
Summary: The student request to view their timetable. The system provides the student's timetable, but also any errors or warnings about clashes in activities or unchosen tutorials or labs.

Task 4: Non-functional requirements

Below are example non-functional requirements (NFRs)in the required format, which were suggested by the focus group with the university representatives. The ones numbered 3 and 4 are also measurable. Please note that the use of ’should’ instead of ’shall’ should be rare (if not sure of this NFR) or better completely avoided. Other reasonable NFRs for this kind of system are also accepted.

1. **Use case: ’Add course’, ’Remove course’**

   Non-functional requirement: The system shall only allow authorized admin staff to manage courses.
   Non-functional requirement category: Security

2. **Use case: ’Add course’, ’Remove course’, ’Remove FAQ Q-A’**

   Non-functional requirement: The system shall offer a response time of maximum 3 seconds when adding and removing information.
   Non-functional requirement category: Performance

3. **Use case: all apart from ’the ’view’ use cases which do not require user input**

   Non-functional requirement: Error messages provided by the system shall include what the problem was.
   Non-functional requirement category: Usability

4. **Use case: all**

   Non-functional requirement: It shall not take users more than 8 interactions with the system to accomplish any goal on the system.
   Non-functional requirement category: Usability

5. **Use case: all apart from ’view’ use cases which do not require user input**

   Non-functional requirement: The system shall be able to handle incorrect user inputs.
   Non-functional requirement category: Resilience

Task 5: The software development

For any reasons to the answers below, others are also accepted as long as they are correct.

1. a) Software product engineering b) Reasons for choosing software product engineering: the new system is meant as a universal solution to be sold to several customers (universities); it is developed by our own company (Acme Corp) with the approval of its former customer (University of Hindeburg) and seeing an opportunity in the market; our company decides where to take this development in terms of features, changes, life duration; the facilitated meeting highlights the two university reps as potential customers, and indeed in product engineering customers only emerge once release to market; the meeting is organised as market research, which can happen in product engineering. c) Reasons for which this assignment hasn't been doing software project engineering: no external customer initiating the development and contracting out company to develop the system (University of Hindeburg mentioned as not being part of the work); the software to be made into a universal solution for more customers, and not a custom solution for one; features are suggested by the two university reps, and informed by previous evaluations with users, but it is our company's decision whether to develop them (also for other customers); the university reps (or any other external customer) do not decide on changes and the life of the software (for other customers).
2. a) Plan-driven software development process. b) Reasons related to the characteristics of this type of process: requirements are being done thoroughly for the entire system before proceeding with design and implementation; we produce heavyweight documentation; we use UML formally.
3. Yes, agile would have been better in this context. Reasons: it is the recommended approach for product engineering because it can deliver quick releases, reactivity to changes, communication with potential customers; the two university reps, as well as others who are potential customers and consumers, may be willing to continuously collaborate to help decide on features for each iteration, and evaluate iterations; more likely to meet future customer expectations; quicker releases would be possible which can help establish a good position in the market and beat competition; possibility to innovate due to flexibility and adaptability of agile.

Borislav Ikonomov, Cristina Alexandru, Luke Tervit, 2025.