Inf2C Software Engineering 2017-18

Tutorial 1 (Week 4)

Notes on Answers

1 Introduction

2 Warm-up exercise

Marking scheme for the exam question was as follows. Note the acceptability of different correct answers, but also the need for correct use of UML.

(a) Bookwork. 1 each for explanation, 1 for example.

(b) Application of knowledge.

Stakeholders directly mentioned in the situation description, along with example brief explanations of their interests in the new system, are

- *Patients*: - wanting system that helps doctors and nurses deliver good health care to them
- *Doctors*: - wanting system that supports them in delivering good health care to patients and reporting outcomes to the country
- *Nurses*: - wanting system that supports them in delivering good health care to patients
- *Administrators*: - wanting system that helps them provide support to patients, doctors and nurses

Then, as the just-elected ReformHealthCare political party is important, we have

- *Politicians* - wanting to see health outcomes
- *Voters* - wanting to see health outcomes

Further stakeholders include

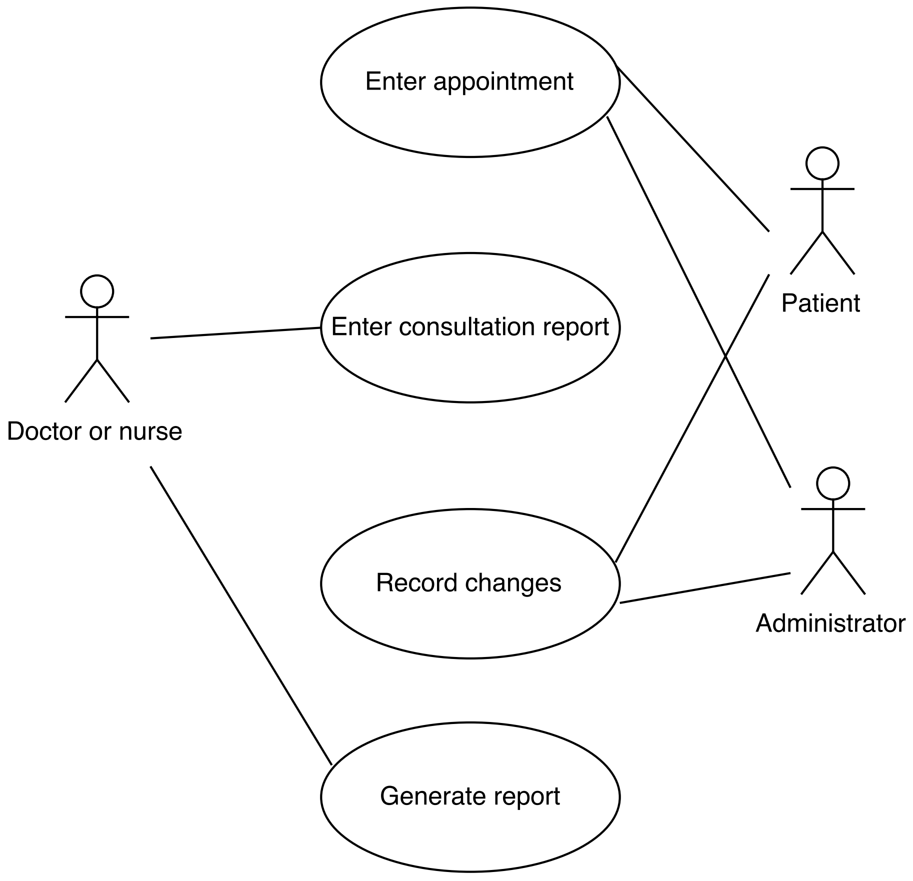
- *Tax payers* - wanting system to succeed and value for money

- *Software engineers* - wanting to deliver system on time within budget meeting requirements

Question only asks for 4 stakeholders. Hope that students get those explicitly mentioned or hinted at, i.e. those in first two groups. Some may coalesce doctors and nurses, which is OK given what the question tells them, but unlikely to be right in real life. Technically, those in last group are acceptable, though if students list those rather than those that are more explicit, they are missing obvious information provided to them. 1 mark for each identification, 1 mark for each explanation.

(c) Bookwork. 4 for "explain", at most 2 if "manage conflicts between requirements of different stakeholders" or similar not included. 1 mark for drawback (only captures requirements that show up as requirements on specific interactions.)

(d) Problem solving. Any reasonable solution acceptable. Marks as follows: 2 marks for actors (Doctor or nurse, Patient, Administrator), 2 for use cases (Enter appointment, Enter consultation report, Record changes, Generate report), 2 for associations, 1 for details of notation e.g. capitalisation, 1 for comments. An example is:



Here Patients are shown as interacting with the *Enter appointment* use case, as it is reasonable to assume they are closely involved in these use cases, and, as with reporting changes, maybe the system could allow them to make appointments directly, without the involvement of administrators.

The specification doesn't make clear who asks for the reports to be generated and who they are intended for. For example, given the outcomes are of wide interest, we might assume some reports are desired by politicians, some government statistics office, and ultimately the public. The diagram assumes the doctors in the first place ask for reports, as presumably these reports might help them monitor and improve the controversial treatments.

3 Capturing requirements

3.1 Lift

Here we introduce the lift as an actor: The lift is indeed an agent external to the system whose cooperation is needed in order to fulfill the use case goals.

We imagine some extensions when there can be mechanical failures. These highlight the need for some alarm system, not mentioned in the question system description.

We assume there is some mechanical arrangement by which the doors at each floor open together with the doors on the lift itself when the lift is at the floor. For simplicity we assume the lift position displays track the position of the lift. We skip mentioning updating these displays in the use cases.

The second use case here includes a *Stakeholder & Interests* field to illustrate its use. A similar entry could be used for the first use case.

Use case name: Call lift
Primary actor: User
Supporting actors: Lift
Summary: User requests lift to come to their floor
Precondition: Lift is stopped at some floor with doors open
Trigger: User presses call button at some floor
Guarantee: Lift is at floor of call, doors are open
Main Success Scenario:
1. System turns on call button light and closes doors
2. System commands lift to move to floor of pressed button
3. Lift arrives at floor of pressed button
4. System turns off light in call button
5. System commands doors to open

Extensions:
1a. Lift is already at floor of call
  .1 System finishes MSS immediately
3a. Lift motor fails before arrival at destination floor
  .1 System sounds alarm
5a. Doors stick closed
  .1 System sounds alarm
Notes:

- It would be easy to tweak this use-case to relax the *doors open* precondition..

- A non-functional requirement associated with this use-case is that the lift should arrive at the floor of the call within some specified reasonable time.

Use case name: Request destination floor
Primary actor: User
Supporting actors: Lift
Precondition: Lift is stopped at some floor. User is in lift.
Trigger: User presses one of destination floor buttons inside lift
Guarantee: Lift is at destination floor selected, doors are open
Main Success Scenario:
1. System turns on destination button light and, if doors are open, closes them
2. System commands lift to move to destination floor
3. Lift arrives at destination floor
4. System turns off light in destination floor button
5. System commands doors to open

Extensions:
1a. Lift is already at destination floor
  .1 System finishes MSS immediately, ensuring doors are open

Stakeholders & Interests:
User – Reaching their desired floor
Health and Safety Officer – Users are not put in danger (e.g. by having doors open between floors). Assistance is requested if the lift gets stuck.
Building Manager – Throughput requirements of lift system are satisfied
Notes:

- Extensions could be added for mechanical failure conditions as before
- A non-functional requirement associated with this use-case is that the lift should arrive at the destination floor within some specified reasonable time.

3.2 Shopping List App

Use case name: Add recipe to recipe library
Primary actor: User
Precondition:
Trigger: User adds recipe
Guarantee: New recipe in recipe library
Main Success Scenario:
1. User enters name of recipe and ingredients

Extensions:
Notes: Does there need to be some library of ingredients? If instead user invents ingredient names every time, it seems there would be an opportunity for multiple versions of each ingredient. This would confuse the shopping list generation process.

Use case name: **Determine shopping list**
Primary actor: User
Summary: Determine a shopping list for some chosen set of recipes
Trigger: User starts a new shopping list
Main Success Scenario:
1. System displays recipe library
2. User selects recipes to shop for, and maybe quantities for each recipe
3. System displays shopping list

Use case name: **Remove ingredient**
Primary actor: User
Summary: Remove ingredient from shopping list
Precondition: System displays shopping list
Trigger: User selects ingredient for removal
Guarantee: System displays shopping list without item selected for removal
Notes: Should the behaviour be the same both for removing ingredient because user already has it and because ingredient picked up in shop? Would it make sense to have two distinct Actors, say Cook and Shopper?

Paul Jackson. 2nd Oct 2017.