Inf2C Software Engineering 2015-16

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 are patients, doctors, nurses, politicians, administrators (NB 5 of them, and Q only asks for 4 - some may coalesce doctors and nurses, which is OK given what the question tells them, but unlikely to be right in real life, so we have one group spare). 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 (doctorornurse, patient, administrator), 2 for use cases (enter consultation report, record changes, generate report), 2 for associations, 1 for details of notation e.g. capitalisation, 1 for comments.

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.

Use case name: User calls lift
Primary actor: User
Secondary 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: User requests destination floor
Primary actor: User
Secondary 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

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. 9th Oct 2015.