Inf2C Software Engineering 2016-17

Tutorial 1 (Week 4)

Requirements Engineering

Study this tutorial sheet and make notes of your answers BEFORE the tutorial.

1 Introduction

In this tutorial you will look at some example systems and create for them use-cases, use-case diagrams and lists of additional requirements.

2 Warm-up exercise

This is a relatively-easy question from the Dec 2009 exam. The marks numbers for each part indicate the relative importance of the part in the exam marking scheme. Students in this exam (and other more recent Inf2C-SE exams too) had 60 minutes to answer 50 marks-worth of questions, so students were expected to be spending just over a minute for each mark.

(a) What is a *functional requirement* and what is a *non-functional requirement*? Explain, giving an example of each. **(4 marks)**

(b) After the ReformHealthcare party's surprise win in the 2010 UK General Election, a new type of health clinic is to be set up, using treatments whose effectiveness is highly controversial. Because the health outcomes associated with the new clinics will be of wide interest, a new computer system must quickly be developed to track patient treatments and outcomes.

All patient appointments must be entered into the system by the clinic administrators. During the appointment, the doctor or nurse will enter details of the patient's condition and any treatment prescribed. Patients are requested to report on any changes in their health, one week after each appointment. They can do this either by filling in a web form or by telephoning the clinic, in which case an administrator will enter the information.

The system must be able to produce reports of statistics such as the number of patients treated in a given period, their conditions, and their reported state of health a week afterwards.

Suggest four different stakeholders in the project, with a brief explanation of why each has a stake. **(8 marks)**

(c) Explain how use cases can be useful for managing requirements of systems with multiple stakeholders, and give one drawback of use cases for managing requirements. **(5 marks)**

(d) Draft a use case diagram for the system described in part (b). Comment briefly on any important ambiguities or problems you find in the system description. **(8 marks)**

3 Capturing requirements

This question gives you practice analysing some simple systems and identifying requirements and use cases for them.

The next subsection outlines two example systems. For each, write out one or more use cases, draw a use case diagram, and note down any important requirements missed out by the use-cases. Use cases often miss non-functional requirements and also functional requirement that specify what bad behaviours should never happen. For example, with the lift system, it is important that the lift doors never open when the lift is between floors. You might well need to make more assumptions about the described system. If you do, also note them down.

To write each use case, use the template given after the descriptions of the systems. This template is similar to that presented in the Use Cases lecture.

3.1 Example Systems

3.1.1 Lift

Consider a single lift system where on each floor by the lift doors there is a button for calling the lift and inside the lift there are buttons for selecting the desired destination floor. The lift controller system monitors the buttons and position of the lift, and controls the lift motor for raising and lowering the lift, the doors opening and closing, lights in each of the buttons, and lift position displays both in the lift and above the doors at each floor. The light in each button should normally be lit when the button is pressed and then turned off at some appropriate point later in time. Assume for simplicity there is just one user at a time.

3.1.2 Shopping List App

Its common for a person to quickly jot down a few items they require from the shops before going. In particular items required for a particular recipe. Imagine creating a shopping-list application for a smartphone. The user should be able to store recipes such that all the required ingredients are added to the current shopping list. They should in addition be able to remove some of those ingredients if they already have them. Another good feature would be substitutions, such as ingredients which could be used in place of the desired ingredient/item if none are available. Of course the user should be able to tick of items as they are placed into their basket or bought at particular shops. An implementation could make this the same operation as for removing items from a recipe that the user already has.

3.2 Use-Case Template

Use case name: Often a short verb phrase.

Primary actor: The actor with the goal the use case is trying to satisfy. Is usually but not always the initiator of the use case.

Supplementary actors: Others the system communicates with while carrying out the use case.

Summary: A one or two sentence description of the use case. Make clear the goal that the primary actor wishes to achieve. Try not to just reiterate the steps in the main success scenario.

Precondition: What the system should ensure is true before the system allows the use case to begin. Useful for telling the programmers what conditions they don't have to check for in their code.

Trigger: The event that gets the use case started.

Guarantee: What the system will ensure at the end of the use case in the event the use case execution is a success. Sometimes it can be useful to distinguish **success guarantees**, **failure guarantees** and **minimal guarantees** (guarantees for what holds at the end of all scenarios).

Main Success Scenario: The steps the primary actor and system would go through to achieve the goal. Focus on the most common and simplest course of events. If there are alternatives, record them in the Extensions field or in a separate use case.

See the lecture slides for an online shopping example. Note that each step is numbered and starts with the name of the agent (the actor or the System) that carries out the step. In describing steps, show the intent of the actor, not the low level details of e.g. user interface interactions.

Sometimes the first step is already described in the trigger. If so, don't bother repeating it.

Extensions: Variations on the Main Success Scenario. Start each with an identifier related to the step at which it starts and a statement of the condition for when the extension applies. Then list the alternate steps and if and when you return to main scenario. Ensure it is clear whether a variation results in the use-case goal being achieved.

Stakeholders & Interests: A stakeholder is someone or something that has an interest in the behaviour of the use case. The primary and supplementary actors are usually stakeholders, but also there can be *silent* actors. For example, a health and safety officer might be a silent actor for the lift system example. List the stakeholders here, and for each summarise briefly their interest in the use case. Even though these silent actors do not directly take part in the use case, it is important that their interests are considered when designing the use case.

Notes: Comments on the use case. E.g. mention associated non-functional requirements or open issues.

In using this template, don't feel obliged to fill out all the fields for every use-case, just use those you feel are appropriate and needed to explain the use case.

Paul Jackson. 4th Oct 2016.