Inf2C Software Engineering 2017-18

Coursework 1

Capturing requirements for a tour guide app

1 Introduction

The aim of this coursework is to produce a requirements document for a tour guide app that could be installed on a tablet computer, smartphone or smartwatch. The app guides the user along the route of a selected tour, highlighting features of interest. The choice of kind of tour supported is left to each coursework group. For example, the tours could be on foot, by bicycle or by car. They could be around a city, in the country or within a building.

Later courseworks will involve the design and implementation of this software. These later courseworks will focus on the core functionality, not the user interface.

2 System Description

2.1 Background

A *tour* is a finite sequence of *waypoints*, positions on a map. A *leg* of the tour is a stage of the tour between two adjacent waypoints. Each waypoint optionally has an *annotation* which may be some combination of text, audio, pictures and video giving information about the waypoint. For example it may tell the user what things to look at and something of the history of these things. Also each leg has an optional annotation, perhaps giving information to interest the user or navigational information. For example, a leg annotation might remark that the leg follows a footpath or might give a street name.

2.2 Following a tour

The app helps the user navigate between waypoints by continually telling the user the direction of the next waypoint and the distance to it. To do this, we assume the device running the app provides some location sensing service. If the tour is outside, this may be GPS based. If indoors, this could be based on infrared or bluetooth beacons, perhaps in combination with input from accelerometer sensors.

For simplicity we also assume that users can travel more or less in a straight line on each leg; there are not obstacles that users have to navigate around.

As the tour proceeds, the app presents the user with the annotation for the current leg, if there is one.

Whenever the user is in the neighbourhood of a waypoint with an annotation, the app presents the user with the annotation.

On arrival at a waypoint that is not the last on the tour, the next waypoint being sought is automatically updated to the next waypoint on the tour.

2.3 Browsing tours

When a tour is not being followed, the app offers the user a choice of predefined tours. Each tour has an annotation giving overall information about it to help the user decide which they might want to go on.

In practice, predefined tours would likely be stored on a server maintained by the app provider. Perhaps the selection process might largely run on the server, with users browsing tour annotations held on the server and tour details only downloaded to the app itself when the tour is selected. Alternatively, perhaps the app downloads from the server a selection of tours for the region of interest to the user.

For simplicity, unless otherwise indicated, this and future courseworks need not consider this probable distributed nature of the whole tour guide system. Instead, let us assume that all predefined tours are stored in the app itself.

2.4 Selecting a tour

If and when the user selects a tour they would like to go on, the app presents instructions on how to get to the tour start. If the selected tour costs money, the tour fee is collected from the user at this point. Let us assume that the app itself is available free of charge and that tours can be browsed for free.

2.5 Creating tours

The app has a baseline tour capturing facility involving a tour author following the actual route of an intended tour. The author notes in the app whenever a waypoint position is reached and enters waypoint and leg annotations when the corresponding waypoint or leg is current.

We leave unspecified who can author a tour. Maybe the app provider creates tours, maybe distinct tour authors are employed, maybe all app users can contribute new tours.

With a real tour guide app it is likely that tour creation and editing would take place largely on some tour guide server using distinct server-based software. As we are not considering the nature of any tour guide server, let us ignore this complication.

2.6 Extensions

The above has just sketched out the basics. In real life many refinements would be possible and desirable. A few are sketched below. Unless otherwise specified, you should not cover these in the requirements document you produce for this coursework. Some of these or others may be included in later courseworks.

- The app could overlay tours on a map.
- Position information on each waypoint could include not only its 2D position on a map but also its height. For example, this would be useful if the app is used for hill-walking.
- A facility for users to add tour ratings or tour comments could be added.
- Each time a tour is followed, the time each waypoint is visited could be recorded. A user could then compare these times with those from previous occasions they followed the same tour. This would be useful if say the user was running the tour and keen to try improving their time. Possibly different users could compare their times.
- The time to completion of a tour could be estimated from knowing the average speed so far on a tour.

3 Your job

Your job for this coursework is to create a requirements document for the software that expands on the information presented above. Include in your document the information asked for in the following tasks.

3.1 Choose an app name

3.2 Describe the kind of tour supported

Choose a kind of tour you would like your app to be targeted for, and describe it in two or three sentences. Fixing this might help you more concretely think through stakeholders and requirements.

Refer the reader to these coursework instructions for further general information about the nature of the app.

3.3 Identify stakeholders

Identify the stakeholders of the system and the value or benefit each seeks from it. Focus on stakeholders particular to this tour guide application. There is no need here to cover stakeholders common to most software development projects – software architects, designers, coders and testers, for example.

3.4 Describe system state

Describe in broad terms the nature of the state of the system. For example, include mention of the different modes of operation and key different states the system might be in within each mode. Use an enumerated list, and, as needed introduce multiple levels of lists.

Later on, this categorisation of the system state can help when describing the preconditions and guarantees of use cases.

This description forms part of the description of functional features of the system.

3.5 Describe use-cases

The provided description can suggest different numbers of use cases, depending on the size and level of abstraction of each use case.

For this coursework consider the three use cases

1. **Follow Tour**, including the selection of a particular tour.
2. **Browse Tours**
3. **Author Tour**

Keep the use cases high level: they are about the main interactions between actors external to the system and the system itself, they should not be concerned with all the details of user interface interactions: you are not doing design at this stage.

For the *Follow Tour* use case, produce a description using a template similar to that described on the *Tutorial 1* question sheet used for the Week 4 tutorials. Feel free to add extra fields if you feel it would help, and also omit fields when they are unnecessary. Focus mostly on the Main Success Scenario, noting at most one or two extension scenarios. Are there any failure scenarios?

The description of how the mobile device display and audio are updated cuts across much of the behaviour this use case characterises. Feel free to break this information out into a note included at the end of the use case, so the MSS steps then focus primarily on how each changes the central system state. This note can be structured as an enumerated list, as there are several categories of information presented; waypoint annotations and the direction of the next waypoint, for example.

For the other two use-cases, a simpler format with just the primary actor and a free-text description is fine.

3.6 Use case diagram

Draw a UML use-case diagram summarising the use cases and the actor or actors each is associated with.

3.7 Describe non-functional requirements

Describe non-functional requirements using two or more levels of enumerated lists. For this task, consider a realistic system without the various simplifications suggested earlier and perhaps with some extensions such as those sketched above. Also consider the requirements of the whole system, encompassing not only the app software but also the mobile hardware the app might run on and perhaps briefly too the server-side hardware and software. Such requirements for the whole system could likely imply more specific requirements on the app software.

General categories of non-functional requirements include security, usability, performance and reliability. Consider the notion of performance broadly, including not only time-related performance, but also power consumption, location sensing accuracy and mobile roaming costs, for example.

3.8 Ambiguities and subtleties

The provided system description above omits many details. A number of these might be resolved at a later design stage. But some are ambiguities that need to be settled and agreed to by the various stakeholders.

Make a note of ambiguities you have noticed, when appropriate making suggestions of obvious ways you think these might be fixed.

You may find it most convenient to include these suggestions in other tasks, but do summarise these suggestions here.

4 Asking Questions

Please ask questions on the class discussion forum if you are unclear about any aspect of the system description or about what exactly you need to do. For this coursework tag your questions using the *cw1* folder. As questions and answers build up on the forum, remember to check over the existing questions first: maybe your question has already been answered!

5 Good Scholarly Practice

Please remember the University requirement as regards all assessed work. Details about this can be found at:

`http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

6 Marking scheme

Marks will be allocated as follows.

- 15% Stakeholders and their interests (3.3)
- 45% Functional requirements (3.4, 3.5)

- 15% Use case diagram (3.6)
- 25% Non-functional requirements (3.7)

A separate mark will not be allocated for comments about ambiguites (task 3.8), but their consideration will influence the other marks.

7 Submission

Please submit two files

1. A PDF (not a Word or Open Office document) of your requirements document. The document should be named **report.pdf** and should include **a title page with names and UUNs of the team members**.
2. a text file named **team.txt** with only the UUNs of the team members (one UUN on each line) as shown,

   ```
   s1234567
   s7891234
   ```

How to Submit

Only one member of each coursework group should make a submission. If both members accidentally submit, please alert the course organiser so confusion during marking is avoided.

Ensure you are logged onto a DICE computer and are at a shell prompt in a terminal window. Place your *report.pdf* requirements document and your *team.txt* file in the same directory, ensure this directory is set as your current directory (i.e. `cd` to it), and submit your work using the command:

```
submit inf2c-se cw1 report.pdf team.txt
```

This coursework is due at

16:00 on **Thursday 19th October**

The coursework is worth 20% of the total coursework mark and 8% of the overall course mark.

Paul Jackson, 9th October 2017.