Software Requirements Specification

In this document I will briefly describe the purpose and intended use of the software I will be developing. I will then set out the software's features and requirements.

Purpose

The software I will be developing is a drone lunch delivery system which the School of Informatics has tasked me with creating. The system will allow students to save time on their lunch breaks by simply ordering from a selection of local restaurants and waiting while a drone picks up and delivers their order. It is hoped that the system will save time in students' busy schedules, and allow students who are new to the city to get food easily.

Overview

In order to deliver this functionality various steps must take place. The School of Informatics (SoI) will facilitate lunch orders in the morning and add these to an Apache Derby database, the system must read and store the orders from this database. Also kept up to date by the SoI is a webserver containing restaurants that are taking part in the scheme, their location and their menus. The webserver also contains the location of a series of no-fly zones - zones where other faculties in the University have prohibited our drone from flying - and the location of landmarks in Edinburgh to be used to help navigation. The system must read and store this information from the website. Once the system has retrieved all this information it must then calculate a flightpath that delivers orders within a series of constraints and aiming to maximise profit.

In order to create this software, I have divided it into smaller chunks to make testing and development easier. The smaller areas of functionality are Menus; responsible for retrieving and storing the restaurant and menu details from the webserver, Orders; responsible for retrieving and storing the orders for a given day from the database, Movement; itself split into 'Navigation' - how the drone stays within confinement area and avoids no go zones while visiting restaurants and 'Manoeuvre' - how the drone makes moves, calculates angle, distance etc., Record; how the system accurately records and stores data on its orders and deliveries, and finally App; the central function tying the whole system together.

Functional Requirements

The level of the requirement is given in brackets after the requirement.

Manoeuvre:

- System can calculate Euclidian distance between two Long, Lat coordinates (unit)
- System can calculate angle between two coordinates (unit)
- Drone can fly in a straight line in steps of 0.00015 degrees (unit)
- Drone can only fly in a direction that is a multiple of 10 from 0 to 350 (unit)
- Given a starting location and a direction, the system can calculate the next position after a move (integration)
- Drone has two move types, fly or hover (unit)

- Hovering takes up a move but does not change position and has angle of –999 (unit)

Navigation:

- Drone makes at most 1500 moves on any one day (System)
- System can test whether a coordinate is 'close to' (within 0.00015 degrees) another coordinate (unit)
- Drone should be 'close to' any restaurant it visits/drop off point/Appleton, within 0.00015 degrees (Integration)
- Drone must hover for one move when collecting a lunch order from a shop or delivering a lunch order to a customer (integration)
- Drone is launched from (−3.186874,55.944494) and returns close to this position at end of deliveries (integration)
- The system can obtain landmark coordinates (unit)
- The system can obtain feature coordinates (unit)
- Drone must stay within the confinement area (integration)
    - System can test whether a coordinate is within the confinement area (unit)
- System can test whether a given flight between two coordinates is possible without entering no-fly zones (integration)
- The system can obtain a location from three words using WhatThreeWords (unit)
- Given two coordinates the system can generate a series of moves moving the drone from one to another, avoiding no-fly zones and staying within the confinement area. (integration)
- The system can calculate how many moves would be required to navigate between two points including avoiding no-fly zones (integration)
- The system can calculate how many moves would be required to complete an entire order (integration)

Menus:

- The system must connect to webserver, read, and store the required data (unit)
- The system must be able to calculate the cost of delivering a certain order (unit)
- The system can calculate the delivery cost for a given list of items (unit)

Orders:

- The system can obtain the orders for a given day (unit)
- The system can obtain the details of a specific order given its orderID (unit)

Record:

- Each lunch delivery must be recorded in a table in specified format (unit)
- Each move made by the drone must be recorded in a table in specified format (integration)
- The system should produce a GeoJson file containing the flightpath of the drone in specified format. (integration)

App:

- Given a day and ports for the webserver and database; the app can obtain relevant data, calculate a flightpath to deliver orders for the day, while observing relevant constraints and storing data documenting the deliveries. (system/acceptance)
- Code should be readable (system)
    - Variables should have meaningful names
    - All methods should have Javadoc comments
- Code should be clear (system)

Non-functional requirements

- System must run in reasonable time to ensure happy customers (<60s) (system)
- System should be as cost efficient as possible, making the most money available from the orders for a given day (maximise sampled average percentage monetary value) (system)
- System should have high reliability (high MTBF) (system)
- Control of the drone is not susceptible to external control (cannot be hacked/attacked) (Security) (System)
- System can handle invalid inputs (system)

Test approaches

In this section I will discuss my plan for which testing approaches I will use. I will break down my testing project into; the nature of the specification, my experience, budget/quality constraints, and scaffolding costs. I will then use this to choose a selection of approaches which I will apply in due course.

Nature and form of the specification

Looking at the specification, most of the input domains are constrained in that there is a clear acceptable range of values. As with the principles discussed in 10.4 of the textbook, this points towards a partitioning method with constraints such as the category-partition approach. An example of the requirements that could be tackled with this approach are “Drone can only fly in a direction that is a multiple of 10 from 0 to 350 (unit)” (as this contains multiple restrictions on the input domain). This approach is beneficial as it would help us to effectively test these requirements by producing a test suite that has good coverage without too many test cases.

Due to time constraints some of the less important requirements should be tested using random testing, this should still provide reasonable coverage while saving me a lot of time allowing me to implement more time expensive methods such as model-based testing.

After the initial testing of modules aimed at meeting specific requirements, it is important to test the system more integrated. This leads me towards structural testing – testing the system based on its structure and ensuring adequate coverage over the system. For this (in a trade off between coverage and budget) I aim to use branch testing to achieve adequate coverage of the system.

Looking at the system structure we can see that it lends itself to a model-based approach at the system level. There is a clear structure with transitions between system states for example we would have a state for obtaining order details, one for obtaining restaurant details, one for the state were in

when we are controlling the flight of the drone with options for whether a direct route to destination is possible and one for when it is not, and so on. This model-based approach would help us to test the system based on software and structure as opposed to just on the specification.

Experience of test designer

I personally have experience mainly in category-partition and catalogue testing, I also have experience in finite state machines through other courses though not for testing. Therefore, I should lean towards these methods. Especially in model-based testing as learning and implementing another method of model-based testing could be extremely expensive in time and resources.

Budget and quality constraints

Overall, the quality of the system is important both in terms of safety (the drone doesn't fly through no-fly zones and stays in confinement area) and in terms of making sure the system is viable (effective and efficient). However, in terms of budget I have many other subjects and a project to do therefore for the sake of the course I aim to use an extensive range of methods to test the software but not to test the software completely extensively i.e., to pick a range of requirements and test them thoroughly with different techniques, however, not to test all requirements.

Scaffolding costs

For this project, each test case will have to be realised in the form of scaffolding code written by hand. This will be a time expensive process however most of the requirements will only require a small amount of scaffolding, so it is a viable approach. Test execution will be done automatically and therefore extensive testing is possible.

Summary

Taking all of this into account it seems the best testing approach given the specification and my experience is a functional approach initially, testing individual units using partition/category-based testing and random testing where appropriate. Then later in development implementing structural testing to ensure effective coverage of the software. In order to complement these techniques, I propose using a model-based approach, constructing a finite state machine and using transition coverage to effectively test the software.

I do, however, plan to give an example of various types of testing (even if they are not explicitly mentioned) in order to demonstrate my understanding of the material in this course.

Appropriateness of given approach

Generally, this approach is appropriate for the given specification, however there are limitations. I am opting for methods such as random testing in the interests of time, however these methods can often miss errors which other more time expensive methods would capture. A good aspect of the plan however is the range of methods, hopefully this will result in a robust system as the various methods, at various levels (system, integration, unit) should ensure that the final quality of the product and that most errors are detected by the end.