Software Testing 2022

S1833951 li fang

January 26 2022

2.1 Construction of the test plan

R1: the drone should remain in the restricted area always

- This is a safety requirement, so we should devote a reasonably high level of resource to ensuring we meet the requirement.
- The principles of Chapter 3 indicate we should consider at least two different T&A approaches if the requirement has high priority.
- The visibility principles suggest we should be able to tell the progress or status against goal, so it will draw the picture of the output to check what is going wrong with the program. Also, all the point from route should be using standard JSON format for better understanding and consistency of standard. Each variable should be commented about its usage when it is assigned.
- The partition principle suggests we should decompose the requirement in to smaller and more controllable pieces. So, we had divided the restricted area to two parts, one reason is that the route-finding algorithm will not be able to avoid building during early developing stage, as we first only focus on avoiding the rectangular area of delivery.
- The requirement is meet by checking each path in the route does not intersect with any of the line of restricted area, the intersect function does check this by input two point like (x,y) and a list of area q, and check if the path between the two point intersect with any of the area by checking the line of the area is not intersecting with the path.

Input:

- r1 r2 r3 r4: the four points of the restricted rectangular area
- sp: start point of the path
- ep: end point of the path

Output:

- Intersect: Boolean value, this is true if path is not crossing with any of the boundary of restricted area

Specification:

- Intersect is true if path from A to B is not intersect with path from r1 to r2, r1 to r3, r3 to r4, r2 to r4

There are validation issue with this requirement, and synthetic data will be needed to test the system. To achieve this the following tasks will be need to schedule in to testing:

- Generating synthetic data which path does contain path crossing the restricted area
- Have Unit test on the synthetic data and observe the feedback
- Have separate system test for rectangular space and building area, as theses part would be developed differently according to time

R2: the drone should always find the next nearest point when planning the visit

- This is a performance requirement, as we should devote medium level of resources to ensure we meet the requirement normally. But this is one of the crucial feature of this project, we have to use reasonably high resources make sure everything work as expected.
- The visibility principles suggest we should be able to tell the progress or status against goal, so the program will be giving out file allowing us to plot the route generated, and we can check if the drone is always find the nearest route. All the variable should be in geojson format for consistency.
- The partition principle suggests we should decompose the requirement in to smaller and more controllable pieces. So I have divide the requirement to two function, one for calculating the distance between point, one for selecting the nearest point among all the other point.

Input:

- sp: start point
- lp: a list of point not been visited

Output:

- dis: distance between two points
- np: next nearest point that should be visited

Specifications:

- np is determined by comparing all the distance from the start to the list of point.

There is validation issue with this requirement, and synthetic data will be needed to test the system. To achieve this the following tasks will be need to schedule in to testing:

- Generating synthetic data with different point with different distance to the target point
- Have system test on the general function find the nearest point
- Have system test on the synthetic data and observe the response
- Have Unit test for the dis function.

Process and risk

Process: generating synthetic data can be done whenever got time, as the generated route can be used for both r1 and r2. No earlier task is needed, and this could be done manually. However, we can have a function created that create such route on purposes.

The risk would be if we manually generate the data that will be time consuming and hardly cover all the possible circumstances, and using a function generating the route would not be random enough to reflect the true performance of the system.

Scaffolding:

The scaffolding is for the generated synthetic data to use, as we set up a individual test and check if the function does response properly to the synthetic data. So I have two scaffolding:

For r1: generate route that does cross restricted area or going out the test area. See if the function does identify it properly

For r2: generate point around the point test, see if the function does recognize the nearest point as expected

There are plenty of other scaffolding during developing phase, mainly for checking if the output is valid.

2.2 Evaluation of the quality of the test plan

R1 and R2:

For the system test been used for r2

strengths:

- Test the entire system make sure it meets the requirements.
- May founds defect during integration phase, provide a comprehensive test for the software.

weakness:

- It may be difficult to ensure that all possible scenarios and edge cases are covered, for example as if the drone stop at one corner of the rectangular area.
- It may not be feasible to test the entire system in all possible combinations and configurations. Such as having route containing path entering the same building multiple times.
- This can also be time consuming as it requires a full set up before the test.

For unit test for r1 and r2

Strengths:

- It can be faster and less resource-intensive than system testing, since it only requires a limited test environment.
- It can be automatically generated and integrated with the software development process, which can save time and resources in the long run.
- It can increase the overall test coverage of the system, by testing small, atomic units of code that may not be covered by system testing.

Weakness:

- It may not be able to detect defects that only occur when the system is integrated and run as a whole. Such as route containing duplicate path.
- The automatically generated test does not cost much performance but could be redundant.

2.3 & 2.4 Instrumentation of the code & Evaluation of the instrumentation

I have not instrumented the code here because the ILP is not test driven, and the requirement's test is simple and structure easily since we are mainly using unit test here and other been test using scaffolding inside the code during developing phase, the whole requirement is around few important variables as route number, total distance, or true false Boolean value, there is no need to restructure the code or produce any extra variables that make test easier or more efficient. The variable exist is good enough for the job, following the standard procedure should be enough.