# Testing methodologies

In any application that we develop, it is essential for us to be able to test it to ensure we are able to ensure it meets the qualities that we have set to achieve with our application. These qualities act as goals, and are a good measure to see up to what degree we have been able to develop the application.

There are a number of qualities we would have to test on for our application, and each of these qualities have different ways to assess them. Among the qualities that we would like to achieve would be:

* Correctness

Correctness is defined as the conformance of the application to the original specifications that have been set out in the application. We can only say a software is correct if it satisfies the functional requirements set out in the software requirement specifications. The process of verification is to ascertain if the requirements are met.

There are a number of ways for us to work towards correctness, one of it being the use of proven methodologies and processes, and even the use of standard algorithms and libraries in our application development process. However this doesn’t allow us to be able to verify the correctness of the application.

For us to verify, we would need to take the following steps:

* Formal methods
  + We would try to prove the logic of the code mathematically to ensure the logic of the code is correct and is consisted to the logic set out in our software requirement specification
* Inspection
  + Static inspection of the code can be done to see that the code makes sense. It can help to a certain extent to ensure the code is doing the right thing, like calling the right functions or using the correct data types
* Testing with dynamic tools like CREST
  + Because it is difficult for us to manually generate various test cases, it would be possible to use some dynamic testing tools like CREST (since we are using C++ code), to test the code. It can help to provide better code coverage, with automated testing capabilities
* Blackbox testing
  + We can use blackbox testing in order to ensure the output we obtain is consistent to the output we expect to get out of the system. This could be a good starting point for us to begin our correctness testing.
* Whitebox testing
  + Whitebox testing on the other hand will allow us to be able to actually inspect the inner workings of the application. It would be best for all the code that we have written to undergo whitebox testing as it allows us to ensure that the program is truly giving out the correct values. It would be the second step in testing after blackbox testing. While the usage of dynamic testing tools and code inspection may take some time as we have not really done it before, we could use whitebox testing in code blocks which are of appropriate length for us to test by this method.
* Reliability

Reliability is a relative concept, that is, it is the measure of dependability of our software. There is no fixed threshold that defines an unreliable software, hence we would need to depend on creating one ourselves.

For our application, reliability testing can be done after we are sure the program is correct. The best way to do reliability testing would be to measure the number of times the application crashes over a certain number of attempts to run the application. For our application, it would be reasonable for us to set a benchmark of 2 failures per 10 attempts to run the application.

As it is possible for reliable software to be incorrect, it would be necessary for us to ensure correctness is achieved first. In most, if not all cases, if we achieve correctness, we indirectly achieve reliability as well.

We could also potentially introduce a beta release of our software to be used by a select number of people in order to simulate random user usage. As different users will be running the applications through different steps (like some users may start the simulation before starting the infographics, and vice versa), it gives better, more varied input to ensure our program is truly reliable.

* Robustness

Robustness is the ability for our application to be able to cope in unspecified situations, which is the application is able to handle even unexpected input.

The best way to test this is to supply our application with input that is close to the correct bounds of the function, to see if we are able to trigger the error detection functions and to see if the program is able to handle the incorrect input. For example in the case of vehicle speed, we would try to enter in a negative value or an extremely large value, beyond the range of speed values that are accepted by our system to see if it is able to handle this incorrect values properly.

* Performance

Performance is mainly the efficient use of resources. As every computer has a finite set of resources, we would need to ensure that performance of the application is well optimized. Especially in our case where we are making use of threads, we would need to ensure that threads are well managed to ensure we don’t have any runaway threads wasting the computer’s resources.

We should set a benchmark for the performance of our application, so that we can compare our application’s performance. For now, it would be reasonable for us to set that our application should not take up more than 20% of CPU usage on a computer, with a maximum usage of 512 MB of RAM at any one time.

* Usability

Usability would probably be one of the most important qualities we would need to achieve in order to consider our application to be a success. Usability is essentially the level of user friendliness of our application. The best way to test this is by running surveys on a select group of people to use a beta version of our user interface to collect feedback. We could also test to see how long would it take for a user to learn how to use our application and be able to use it fully. In our application, we could do this test by getting a group of users who have sufficient general computer knowledge to use our system after reading the user manual 2 times. This limit can be altered later on once we are closer to the completion of the first beta of the application, as we would need a beta release to conduct this form of usability testing.

* Portability

Another relatively important quality we have to consider is also the portability quality. One of the main things with Windows machines is that it varies in terms of hard ware from machine to machine, mainly because Windows is able to work on a diverse range of computers. In addition, as we are supporting Windows 8, Windows 8.1 and newer versions of Windows, we would have to ensure we run our application on all these versions of Windows on the same hardware first. Once we have succeeded running it on multiple versions of Windows, we could then look at running the software on various hardware. At least, we should ensure that the software runs without any issues on the lab computers made available to us.

By achieving the abovementioned qualities, confirmed via the suggested testing methods as above, we would be able to guarantee that the product produced at the end would be of high standards and would be highly marketable.