Test plan

*Version 1.0*

Smart Traffic Light

Software Engineering Project

*Instructor: Dan Ofir*

*Group member:*

*Maxim Marmer*

*Netanel Davidov*

**Table of content**

1. Test Plan Identifier
2. References
3. Introduction
4. Test Items
5. Software Risk Issues
6. Features to be Tested
7. Features not to be Tested
8. Approach
9. Item Pass/Fail Criteria
10. Suspension Criteria and Resumption Requirements

**1. Test plan Identifier**

The purpose of testing is to find out incorrect results on the program and further prevent them. For this, it is necessary to find errors at the initial stages of the system and not continue to work until these errors are corrected. Correction in the early stages will require less effort than in the later stages of verification. For a correct software tests, it is necessary to divide the tests into separate categories, such as:

* Functionality tests - are tests of specific program operations, such as functions and algorithms and their results and the data operations such as DB storage operation, accessing methods and more.
* No-functional tests - testing excessive use of resources and preventing excessive time for operations, time management.
* Efficiency tests – test for specific functions.
* Input tests - tests of the correctness of the entered data in the program.

**2. References**

Junit – java library for testing<https://junit.org/junit5/>

Project code - <https://github.com/MarmerMax/Smart-Traffic-Light>

Sumo (Simulation of Urban MObility)- simulator for comparing results - <https://sumo.dlr.de/docs/Tutorials.html>

**3. Introduction**

Smart Traffic Light is a program for simulating traffic distribution using an intelligent traffic light that adapts to the current situation on the road. For the program to work correctly, it is necessary to check it at the development stage.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding. Our objective of testing STL is to design tests that systematically uncover different classes of errors and do so a minimum amount of time and effort.

**4. Test items**

* Traffic lights – Modes, color states, state durations, position, car queue, belonging to lanes, times, receiving and executing orders, monitoring, data.
* Crossroads – Modes, position, traffic lights, general info correctness (cars, lanes, directions and more), times, data, send conditions to control algorithms, receiving and executing orders, sending orders to traffic lights.
* Geographic network module
* Main algorithms – receiving condition data, processing, sending correct data structure, calculate correct monitoring and switching traffic lights color states or modes, calculate correct duration states times, parameters, update GUI, update database.
* GUI – correct user window, thread process, running, components, drawing, bounding, visibility.
* Random data
* Database interfaces
* GUI to Database
* GUI to algorithms

## 5. Testability

In theory, a software engineer develops a computer program, system, or product that takes into account "testability." Software testability is an indicator of good architecture and proper operation of the entire system in the future. The following checklist that provides a set of features that lead to the software under test.

* Operability. The better the testing works, the more effective the program may be in the future.
* Observablity. Everything that the user sees and has access to use it should be tested.
* Controllability. The better we can control the software, the more the testing can be utomated and optimised.
* Simplicity. The more correctly the program is written, the easier it will be to test it.
* Stability. The earlier errors are detected and corrected, then the program will be able to work more stable in the future.
* Decomposability. By controlling the scope of testing, we can more quickly isolate problem and perform smarter retesting.

**6. Features to be tested**

Functionality test

GUI

* All buttons have a hover effect
* Spinner fields have two input types: manual and arrows
* Each button has a specific action
* Closing a program requires confirmation
* Аhe font is clear and clearly visible
* The interface is easy to understand
* The user will have access to the result of the algorithm

No-functional tests

Algorithm

* Check that algorithm finish the process in finite time
* Check that algorithm not using excessive resource

Efficiency test

Algorithm

* checkInitialStateDuration - calculate the traffic duration of the initial state
* calculatePassedCars – calculate all passed cars of two crossroad in single time unit
* calculateCars – calculate passed cars in single time unit
* findBetterDuration - calculate a better duration than the actual duration
* isBetterDuration – check if duration is better than other duration

Car

* generateRandomCar – generate random car on road

Crossroad

* changeStates – change states of all traffic lights in crossroad. it is necessary to check that changes in the traffic signal occur in the correct order
* addTimeToEastWestRoute – add time to east-west direction. It is necessary to check that after adding time to one of the traffic light directions, the time in the other direction should change accordingly
* addTimeToNorthSouthRoute – add time to north-south direction. It is necessary to check that after adding time to one of the traffic light directions, the time in the other direction should change accordingly

Traffic light

* changeState – change state of specific traffic light

Input test

Algorithm

* Check input argument that all fields are exist

DirectionInfo

* Check input argument that all fields are exist and correct

Conditions

* Check input argument that all fields are exist and correct

Crossroad

* Check input argument that all fields are exist and correct

CrossroadInfo

* Check input argument that all fields are exist and correct

Road

* Check input argument that all fields are exist and correct

SystemSTL

* Check input argument that all fields are exist and correct

**7. Features not to be tested**

There are the features that are not subject to verification, since they are used with the help of external programs or libraries, so you should rely on their serviceability provided by the manufacturer.

* JDBC – external library for connecting MySQL program
* MySQL – external database to store data
* JavaFX – external library for GUI creation
* Junit – external library for testing

**8. Approach**

This section describes the order and how the tests will be performed:

1. **Unit tests** - are tests at the level of small system units that verify the normal operation of the units. The test usually covers the smallest code unit, which is a procedure or function. So in the initial stage, we want to check that each 'unit' is working and working properly as planned by using JUNIT (a code document allows unit testing in the software), The following is the planning:

* Traffic light
* Check to see if the color changes to the next color in line with the traffic light received.
* Check to see if the traffic light operates according to a specific mode set for it (synchronous mode, autonomous mode).
* Check the traffic light timing, that is, once the traffic light is given a color change command after time X the traffic light does (we will set time X and see if the traffic light has changed exactly according to the time).
* Check the duration of a certain color appearance by the time the traffic light is given as a command and then whether the traffic light really stops the color appearance and moves to the next color.
* Check whether there are cars or not in the lane on which the traffic light is.
* Check how many cars there are in the queue of cars on the traffic light.
* Check if the function that transmits the information from the traffic light to the crossroad where it is located is working properly and the information is indeed the last information that was in the traffic light. The test will take the data of the traffic light given by the manual data calculated on a page and make a comparison.
* Crossroad
* Check if there are several traffic lights at the intersection according to the number of lanes defined (each traffic light should have one lane - one-value match).
* Check the amount of lanes and traffic lights they gave (there cannot be one lane with one traffic light or one lane with two traffic lights because it is not a valid intersection).
* Check lane directions - There can be two lanes, facing each other, and each other in the opposite direction because it is invalid.
* Check that the intersection receives valid information on traffic lights ie there cannot be two traffic lights whose lanes intersect and both with green light or check that no situation can occur with no intersection ie cars but all traffic lights are red.
* Check that the information received by the node from the algorithm is transmitted and distributed correctly to the traffic lights located at the node. The test will include four traffic lights that will each receive information from any node they are in and want to see if the information is being transmitted correctly.
* Check the information transfer from the node to its traffic lights, in the correct timing.
* Geographic network module

We will check if the software correctly places the objects on a geographic network, the dimensions of the network will be predetermined. The tests of this module are done by creating some road junctions, traffic lights, cars and roads with their location data (X, Y) in a 2D axis system.

* Check to see if the software does not allow for predefined location limits. Let's take a look and see if it exceeds max\_x\_bound and max\_y\_bound.
* Random data

The user will be able to start the application with random data. Although the data is random, the program needs to verify it by checking the fields items like real data. That the values ​​are in acceptable norms and do not contradict each other to run the algorithm.

* Database

Access to the database will be local, which means that each user will have access only to their data that is stored on his device. Hence we need to check the correctness of data

that insert and update, and extract exactly the information required for a particular function.

1. **Integration tests** – Their purpose is to test the effect that a defined part of the system has with other parts of the system and with parallel and launching systems. The integration tests are performed by connecting several individual modules and connecting them to a system or subsystem, or to an external system.

* Geographic network module –
* Check the locations of the components on the network at any given time according to the locations in the open source [SUMO](https://sumo.dlr.de/docs/Tutorials.html) simolator at that point in time and generally whether the location setting for all components is technically and visually correct.
* Main algorithms – In this section the tests will be done by comparing results that were pre-calculated on another page or computational program.
* Check whether the algorithm correctly received traffic times and information from the nodes - create nodes including traffic lights in a certain situation where the colors of traffic lights are correct and all information on the roads is correct and at a certain point in time we will see what the algorithm returns.
* Check the results of the algorithm and the waiting times by the open source [SUMO](https://sumo.dlr.de/docs/Tutorials.html) simulator where we will enter all the data of the algorithm and see what the output results of the simulation are if the average amount of time is good or not good or is it better to use automatic (fixed synchronous) or autonomous (variable) mode.
* Database GUI - a window was created for users to use data from the database. In this window, the user will see all his saved data.

- For the correct data storage in the database, it is necessary to check the user request.

- To load data from the database, you must check the correctness of the stored data.

3. **System tests and final tests -**

* Our system should give a simulation that mimics a realistic situation of traffic light system at certain intersections, the system can work in two modes:
* Fixed Time Mode - The green and red light cycle time is fixed (constant) each time and the user can preset it.
* Autonomous Mode - The cycle times for red and green light activity vary according to certain parameters in the situation.
* Tests for simulation in a Fixed Time Mode:

We will compare the results of the simulation according to the results we would expect in reality.

For example, look at a particular crossroads in an arbitrary city where there are 4 traffic lights (one in each direction) that works as follows –

* The traffic lights heading north and south - from 6 am to 12 pm The red light show time is 60 seconds, the red to green switch time is 3 seconds, the green light show time is 18 seconds, the red to green switch time is 5 seconds.

From 12 pm to 6 am the red light show time is 30 seconds, the red to green switch time is 3 seconds, the green light show time is 18 seconds, the yellow time switch to green is 5 seconds.

* The traffic lights heading east and west - from 6 am to 12 pm The red light show time is 30 seconds, the red to green switch time is 3 seconds, the green light show time is 48 seconds, the red to green switch time is 5 seconds.

From 12 pm to 6 am the red light show time is 30 seconds, the red to green switch time is 3 seconds, the green light show time is 18 seconds, the yellow time switch to green is 5 seconds.

Put the additional parameters needed into our system (either manually or from the database) and run the simulation in a Fixed Time Mode (according to the above times), the result obtained should be completely identical to the result in reality, ie the traffic light color change times and the number of cars passing at the same time period should be similar. We will therefore know whether the system meets its requirements.

* Tests for simulation in a Autonomous Mode:

The simulation results according to the requirements should optimize the light show times at traffic lights and thus reduce the waiting times of the cars at the traffic light. The algorithm needs to decide how and how long to allocate each light instance at intersection traffic lights depending on the amount of cars at the intersection, current speed and additional parameters that the user will enter.

Take exactly the same example described in "Fixed Time Simulation Tests" but we'll run it in autonomous mode.

* The measures that will assess the quality and correctness of the simulation are:
* Weighted average time of red light on traffic light –

To measure how good our simulations results are, we want to put as many cars in traffic lights as possible and avoid traffic jams following traffic lights. Therefore, the simulation should calculate how long on average each car that was in the simulation waited to cross the intersection.

* How many cars go through a cycle of green light at all –

To measure how good our simulations are, we want to get as many cars as possible in each traffic light green show, so we will check the average speed and time, how many cars pass the traffic light during a certain time cycle, we want to achieve the optimal result.

* Efficiency versus non-autonomous traffic light –

To measure how well the results of the autonomous mode simulation are compared to the normal situation the intersection traffic lights are currently operating on, we take the average wait time that a car takes to pass the node that we receive in the normal mode versus the wait time it takes for the car to pass the simulation node in the autonomous mode We will accept whether the use of the autonomous mode is more effective in this case or not.

Through these accuracy and efficiency tests, we can tell whether our simulation algorithms are valid, efficient or necessary.

**9. Item Pass/Fail Criteria**

At this stage of program development, specific tests were designed for the overall operation of the system. Tests should check the operation of the entire system so that all operations are carefully checked and errors are caught and eliminated.

The following are the indicators of the tests that were conducted:

* Unit test – all tests were passed successfully. (100%)
* Functional tests – at the moment the application is not finished and it is not possible to check functionality of all the tests.
* No functional - the program works properly at the development stage, however, at this point, the application algorithm is not yet complete and the resource check cannot be reliable.
* Input tests – most of the input data is working properly, we also have errors in some cases. This will be eliminated in the future.

**10. Suspension criteria and resumption requirements**

The STL system in the future should solve the problem of traffic on the road in real time, so the speed of the system is very important. It is extremely clear that in order to solve the problem of traffic jams on the road, the algorithm should work quickly and always provide an option for the development of events equal in time or better, but not worse than the real one. Therefore, the tests should also be fast and not interfere with the algorithm to work.

It is necessary to restart the process if the test exceeds the operating time of the system itself.