Test Plan for Clemson Game Day Queue Warning System Architecture

Test Plan Number	TP_1.0
Test Plan Name	Master Plan for Clemson Gameday Queue Warning System Architecture (CGDQWS)
Version	1.0
Level	Master
Test Type	Functional, Performance and Security
Author	Hameeda Dildar Taj, Simeon Babatunde
Contact Info.	{htag, sbabatu}@g.clemson.edu

Introduction

A test plan is a document that defines the strategy that will be used to verify that the product or system is developed according to its specifications and requirements [2]. The test plan for the CGDQWS architecture below contains information of the testing scope, testing techniques to be used, resources required for testing and schedule of intended test activities. The test plan will focus on the major architectural components of the CGDQWS system.

Test Items

The following is a list of CGDQWS architecture components and functions that needed to be tested:

1. Vehicle OBE
2. Vehicle Data-bus
3. Remote Vehicle OBEs
4. Roadside Equipment
5. ITS Roadway Equipment
6. Traffic Management Center
7. Traffic Information Center
8. System communication functions
9. System algorithm update functions
10. Interoperability functions
11. System response strategies
12. Power control functions

Hardware/Software Risk Issues

The following are the identified hardware and software risks associated with the CGDQWS architecture:

1. Unavailability of replacement for major communication components like Vehicle OBE, Roadside equipment.
2. Government rules and regulations regarding the use of CGDQWS in vehicles.
3. Extremely complex communication feature between Traffic management center and Roadside Equipment, leading to a complex architecture.
4. Insufficient documentation on the operation of the components parts
5. Inability of the drivers to fully comprehend the principle of operation of the CGDQWS systems.

Components listed above may affect system functionality sometimes. Some of the changes, checks that should be done regularly to avoid these issues are:

A. Software Hardware Interface (Middleware)
B. Queue Detection Algorithm
C. User Interface
D. Fail-operational Algorithms
E. System Interface

Features to be tested:

1. Lange Change Strategy : H
2. Power Control: M
3. Thread Periods: H
4. Queue Detection: M
5. Queue Prediction : M
6. Weight Budgets for Components : M
7. Queued Status Broadcast Control : L
8. Power Budgets for Sensors : H

Roadway Equipment:

1. Check if the object provides environmental monitoring including sensors that measure road conditions,surface conditions,surface weather, and vehicle emissions.
2. Check if traffic detectors, environmental sensors, traffic signals,ramp metering systems all work correctly.
3. Check if the signals work properly in the work zone systems such as work crew safety system,driver warning system.

Camera Sensor:

1. Check if the camera is installed correctly in the vehicle.
2. Ensure that the camera captures images correctly.

Features not to be tested:

The following are the features not covered in this test plan:

1. Manual interface

2. Roadside Equipment Sensors
3. Power Control Measurement Devices

Manual Interface: As the manual interface is only needed to display information regarding the operation of the system, it is not required for testing.
Roadside Equipment Sensors: Camera installations, Sensors are not checked.
Power Control Measurement Devices: Power Consumption Levels are monitored but the measurement devices are not required to be tested.

Approach:

Component	Testing Technique	Test Input	Test Output
Vehicle OBE	Check if collision warning information and driver update information is sent to vehicle data-bus	NA	Boolean Value
Vehicle OBE	Check if vehicle control event and vehicle location and motion is sent to remote vehicle OBEs	Vehicle signage data	
Camera Sensor	Check if the camera a self-check for full functionality	NA	Boolean Value
Camera Sensor	Check if most recent set of images are returned	NA	Image Data
Driver	Check if communication between driver and vehicle happens correctly	Driver Updates	Driver Input
Roadside equipment	Check if data exchange happens accurately with Roadway equipment	Vehicle Signage Data	Traffic Situation Data
Vehicle Data-bus	Information must be sent to vehicle OBE	NA	Host vehicle status and driver input information

Testing of the CGDQWS requires unit level testing, module testing, integration testing and finally acceptance testing.

Unit testing: Unit level testing is done in order to check whether the requirements of the system being built meets the primary expectations of the queue warning system. The unit testing is generally done manually and the information is provided to the personnel to do it in an extensive manner. Basic features like the algorithmic logic, complexity along with both

semantic and syntactical errors are looked for. Algorithmic calculations are tested with diverse values with calls to procedures being closely monitored for errors. Test cases should be developed such that they confirm the requirements and specifications of the system.

Module testing: Every module is tested both independently and together with its related modules to see whether it produces the intended outputs. Inputs will be varied to check the correctness of outputs. The testing team and personnel must be aware of the expected outputs and incorrect outputs to verify real time outputs that may occur only for certain scenarios.

Integration testing: The tested modules are integrated and tested further to verify functional, performance, and reliability requirements placed on the system design .Care must be taken while testing the hardware-software interface or middleware interfaces. The development team approves all the test cases. Personnel in the development team will use AADL in order to perform Integration testing. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing.

Acceptance testing: Acceptance testing is done to ensure that the system is acceptable in real-life scenario and to determine if the requirements of a specification or contract are met. Faults that cause errors which if discovered here must be handled immediately with highest priority. The acceptance tests may need to be performed multiple times, since all of the test cases may not be executed within a single test iteration. Gantt charts are used to track the progress of the testing. Test Matrix is periodically checked and updated based on:

1. Type of Defect
2. Cause
3. Severity
4. Time of Handling

Configuration Management Control:

All team members will use a uniform version management tool (GIT) to manage and update the code files and the development manager will be responsible to release the appropriate version for testing. Only the final version or configuration is tested and considered the final code. Regression testing is done for each of the subsequent deliveries, which could either be a bug-fix or an enhancement. Weekly meetings are proposed to discuss the outcome of the testing process and review of the specifications and coding methodologies applied by the testing team.

Staffing and Training needs

In order to ensure effective and comprehensive test coverage, about twenty testers that are well versed with ‘AADL’ using ‘alisa’ are required in order to complete the testing procedure within three months. Appropriate Training is required on using OSTATE2 to deal with verifying the system through the creation of .verify files. Using assure annex and Reqspec is required for effective system testing. Testers and members are to use a uniform version management tool GIT to manage and update code files.

Planning Risks and Contingencies

During the course of the project testing phase, the following are the risks that may arise:

1. Training of the personnel may be delayed in which case contingency would be to increase testing personnel.
2. Modification in requirements at a later stage of testing. Requirements must be properly specified, understood and documented in the specification phase.
3. Shortfall in personnel resources when testing is scheduled to begin.
4. Late handover of system.

Requirements specification is scheduled to end after first month and any further change could result in the following:

1. Unavailability of required hardware, software, data or tools for testing.
2. Delay in the delivery of the software, hardware or tools.
3. Delays in training on the application and/or tools.
4. Decrease in budget for testing. Loss of personnel is likely to occur
5. Test cases may vary depending upon new requirements. Develop alternate test scenarios.
6. Cost of testing is likely to increase. Proper budgeting and prediction of cost is required.

Schedule

Unit testing will begin fifteen days after the creation of the unit components that have to be used to build a module. Module level testing begins when unit testing is completed for the majority part and is set to take place 1 month after the unit handover for testing or 10 days in essence after the testing of units is completed. Integration testing phase begins only when all the modules are completely tested by the team and it scheduled for 2 months after the handover of the project from the development team to the testing team. Acceptance testing is scheduled into the 3rd month of testing. Final test report is handed in 4 months into the testing phase and cannot be delivered later than the fifth month. It should contain FPA's and other metrics based on test results.

Responsibilities

The overall workforce needed for the testing activities involve a development Manager along with a Testing team of fifteen individuals headed by a manager, they are collectively involved in testing the entire Queue Warn system in order to ensure its correct functioning.

Test Manager and Development Manager together are responsible for the majority of the plan.

This issue includes all areas of the plan. Below are their collective responsibilities:

1. Setting risks and defining their scope.
2. Defining the scope of testing within the context of each release/delivery.
3. Deciding on both, features that have to be tested and not tested.
4. Implementing and evolving appropriate measurements and metrics.

5. Planning, deploying and managing the testing effort for any given engagement/release
6. Resource Management during testing.
7. Providing for resolution of scheduling conflicts, especially, if testing is done on the production system.
8. Training Required for testing.
9. Critical decisions for items that are not covered in the test plans is handled by testmanager.

Environmental Needs

The various environmental conditions required in order to develop and test the system are mentioned below:

1. Hardware that is required are simulators, roadside equipment like CCTV cameras, etc, vehicle OBE and other objects.
2. Test data will be provided both manually by testers and specific values for instance cases. Also for certain modules that are inter-related, the data will be from outputs from other modules.
3. Power requirement: there should be backup power in case of shortage or failure.
4. Alisa version 2.0 with Java 1.8 is required
5. Restricted use of the system during testing.

Approvals

Unit testing: A developer on the team that is of level 1 can approve the test results of units. Module testing team programmers or developers of level 2 and level 3 are authorized to approve the test results of the modules being tested. Test Manager and Developer Manager are the only ones that can approve the results of the test on the integrated modules and is hence the highest level of testing before the acceptance testing. For the master test plan, the project manager of the testing team is authorized to give the go ahead signal. Acceptance testing is done by the managers of both the testing and developing team along with the project managers in order to make sure that all the business requirements are met.