Clemson Gameday Queue Warning System Modeling using V&V Techniques and Tools

Hameeda Dildar Taj

Simeon Babatunde

Document Control

Version	Date	Author(s)	Reviewer(s)
V1_120617	12-06-2017	Hameeda Dildar Taj Simeon Babatunde	Prof. John D. McGregor

Table of Contents

Abbreviations

Term	Meaning
CGDQWS	Clemson Gameday Queue Warning System
AADL	Architecture Analysis and Design Language
AGREE	Assume Guarantee REasoning Environment
CVRIA	Connected Vehicle Reference Implementation Architecture
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle

Executive Summary

In a bid to assuage the impact of high traffic experienced on Clemson Gamedays, we modeled a custom queue warning system for use on game days using software Verification and Validation Tools and Techniques. The rationale behind this project is based off of the need to minimize the amount of man-hour directed towards ensuring a free flowing traffic on game days through automation and Intelligent Transportation System. In this project, we demonstrated the use of Architecture Analysis and Design Language (AADL) in modeling the architecture of the system as well as the creation of product components and work products. We concluded by comparing the results of the system requirement verification using AGREE and JUNIT.

Introduction

Clemson University and its environs usually experience a higher level of traffic on game days, this results from the inflow of football fans from different locations in the state of South Carolina. The road users on those days tends to be subjected to a degree of frustration due to the traffic situation. In spite of the intervention of traffic officers, the hope of getting the situation under control seems improbable. In order to alleviate the impact of traffic and also to minimize the occurrence of rear-end collision, we applied the concept of software verification and validation in modeling a queue warning system for typical gameday. The principle of operation of a Queue Warning System is to inform road users or travelers of the presence of downstream traffic based on real-time traffic detection, using warning signs, flashing lights and other means of communication. This helps drivers anticipate an impending situation of emergency braking, reduce queue related collisions and avoid erratic behavior. The system features a dynamic message signs which show a symbol or word when stop-and-go traffic is near. Speed harmonization and lane control signals that provide incident management capabilities is also included in the system.

The Queue Warning application utilizes connected vehicle technologies, including vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications, to enable vehicles within the queue event to automatically broadcast their queued status information (e.g., rapid deceleration, disabled status, lane location) to nearby upstream vehicles and to infrastructure-based central entities (such as the Traffic Message Channel). The infrastructure

will broadcast queue warnings to vehicles in order to minimize or prevent rear-end or other secondary collisions[1]. The Q-WARN application performs two essential tasks: queue determination i.e. detection and/or prediction and queue information dissemination. In order to perform these tasks, the Q-WARN solutions can be vehicle-based or infrastructure-based or utilize a combination of both. The objective of this endeavour is to appropriate V&V techniques in creating the required work products, product components and the final product. This product will help in suggesting responsive courses of action that can help avoid queues that have been detected or even forecasted. Hence, making a strong case for a Connected Vehicle Reference Implementation Architecture based application to enable Cooperative Intelligent Transportation Systems.

CGDQWS Work Products

The work products created during the course of the project involves the following:

System Documentation: The system documentation contains information on the business case and rationale behind the system implementation. In addition to that, it also showcase the stakeholders of the system. Some of the identified stakeholders include: Mobile User, Field User, Center User, Developers, Testers, Project Manager, Supply Manager, Enforcing Entities, and System Analyst.

Hazard Analysis: The hazard analysis contains information on the potential hazards that may arise from the system as well as its environment. It is a systematic examination of systems and subsystem functions to identify and classify failure conditions of those functions according to their severity. The hazard analysis for the CGDQWS contains a listing of key components of the CGDQWS system and their associated hazard related information. Fig. 1 shows the major section of the hazard analysis for the CGDQWS.

Component	Error	Hazard	Severity	Likelihood
ITS Roadway Equipment	Invalid Value	Invalid roadway warning system control value sent by the Traffic Management Center	Catastrophic	Extremely Remote
ITS Roadway Equipment	No Value	Traffic Management Center doesn't send control information	Catastrophic	Extremely Remote
Traffic Management Center	Invalid Value	Incorrect or Invalid traffic flow and images from the ITS Roadway Equipment	Hazardous	Extremely Remote
Traffic Management Center	No Value	ITS Roadway Equipment fails to send data	Hazardous	Extremely Remote
Roadside Equipment	Invalid Value	Invalid or Incorrect value sent by ITS Roadway Equipment	Catastrophic	Extremely Remote

Fig. 1 Hazard analysis for the CGDQWS

Use Case Document: The use case document contains information relating to the Actors, Preconditions, Postconditions, Flows, Requirements and the use case diagram. Fig. 2 shows the use case document, while Fig. 3 shows the use case diagram.

Name of Use Case:	**Queue Warning System**
Created By:	**Hameeda Dildar Taj, Simeon Babatunde**
Date Created:	**10/31/17**
Description:	QWS is designed for Clemson game day to help with the traffic flow.
Actors:	Driver, ITS Roadway equipment, Roadside Equipment, Vehicle OBE
Preconditions:	1.Driver should be able to receive information from Roadway equipment
	2.Communications between various systems are enabled and verified.
	3.All the equipment should be working.
Post conditions:	1.Driver receives information from Roadway equipment.
	2.QWS transmits information to other vehicles and infrastructure systems.
	3.QWS predicted the queue size on the Clemson game day.
	4.QWS disseminates Q-WARN information to other dynamic mobility applications
	1.The Queue Warning (Q-WARN) System informs the Driver of a

Fig. 2

Use case diagram shows the set of actions that systems can perform in collaboration with one or more actors.It describes both behaviour of the system as well as shows the association with actors.

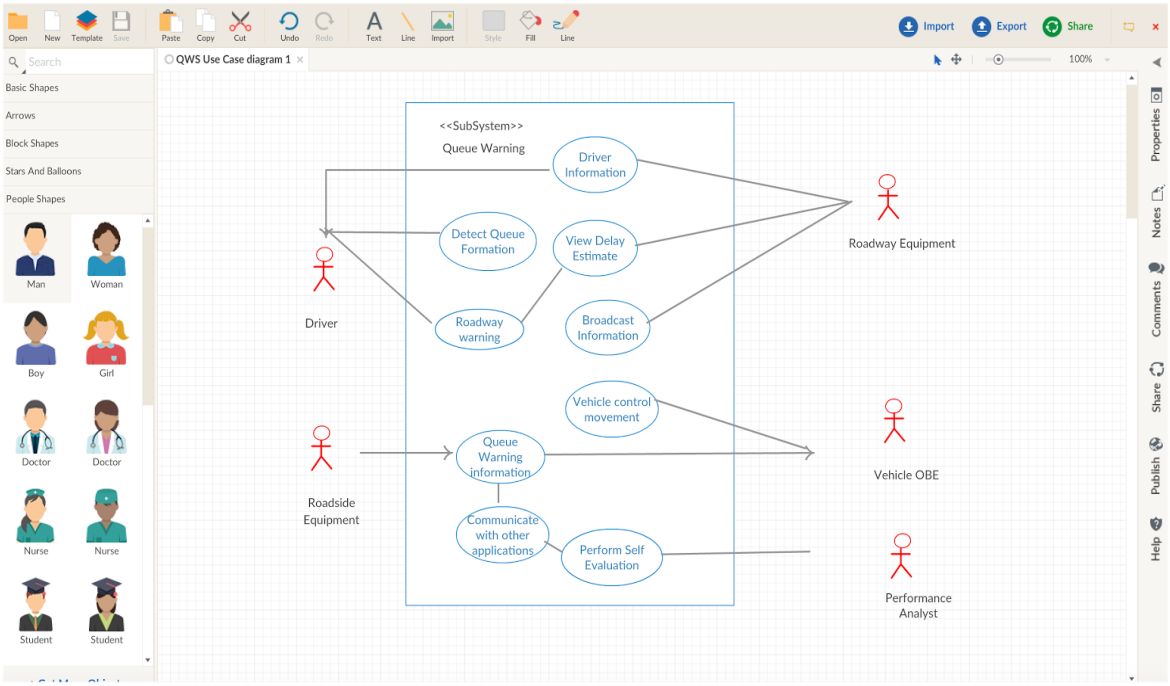

Fig. 3.Use Case Diagram

System Requirement Checklist: The requirement checklist contains information used to verify the architecture during the Architecture Tradeoff Analysis Method (ATAM). This allows the reviewer to ascertain the adequacy of the Requirement Specification Document. Fig. 4 Below shows the requirement checklist created for the CGDQWS.

ID	Defect Type	Priority	Items to Examine
Consistency and Clarity			
1	Unverifiable	Medium	Are there measurable acceptance criteria for each functional and nonfunctional system requirement?
2	Ambiguous	Medium	Does each requirement have only one interpretation, and if a term could have multiple meanings, is it defined?
3	Unverifiable	High	Is each requirement verifiable by testing, demonstration, review, or analysis?
4	Inconsistent	High	Is each requirement written in consistent, clear, concise language?
5	Inconsistent	Medium	Are the requirements free of duplication and conflict with other requirements?
6	Inconsistent	Medium	Does the specification agree with all relevant higher level documents?
Traceability			
7	Interface	Medium	Is each requirement traceable to its source (including derived requirements)?

Fig. 4 System Requirement Checklist for CGDQWS

Test Plan: The test plan document contains an outline of the planning for test process. It contains guidelines for the testing process such as approach, testing tasks, environment needs, resource requirements, schedule and constraints. It defines the strategy that will be used to verify that the system, according to the specifications and requirements. The Test Plan document was created based on the IEEE 829 test plan template. Fig. 5 shows the test plan for the CGDQWS.

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

Fig. 5 Test Plan for CGDQWS

Fault Model: The Fault Model shows a collection of things that could go wrong in the system by capturing all possible errors that could be generated or propagated among the system components before, during or after implementation. Fig. 6 below shows a section of the CGQWS fault model.

Fault Model for the Clemson Gameday Queue Warning System (CGDQWS)

The fault model for the Clemson Gameday Queue Warning System shows a collection of things that could go wrong in the CGDQWS by capturing all possible errors that could be generated or propagated among the system components before, during or after implementation. The table below shows a detailed description of CGDQWS fault model.

S/N	Error Event	Component	Error Type	Propagation (Description)
1	No environmental situation data from Roadway Equipment	Roadway Equipment	No Service	Roadway Equipment like Sensors, CCTV Cameras is the source of error which is propagated to the Traffic Management Center.
2	No signal from Roadside Equipment	Roadside Equipment	No Service	Roadside Equipment is the source of the error which is further propagated to the Traffic Management Center
3	Loss of Electrical Power to the Vehicle OBE device	Vehicle OBE	No Power	Vehicle On-Board-Equipment is the source of the error which propagates to the Roadside Equipment and to other parts of

Fig. 6 Fault Model for the CGDQWS

Requirements Modeling

System requirements can be verified through analysis of AADL models through Architecture Led incremental system assurance capability, which is part of OSATE. Stakeholders requirement details the goals and desires of stakeholders. These are referred to as Stakeholder goals, which should be mappable to specific stakeholder and maybe have sub goals, and does not conflict other goals. Stakeholder goal of clemson game day queue warning system is shown below:

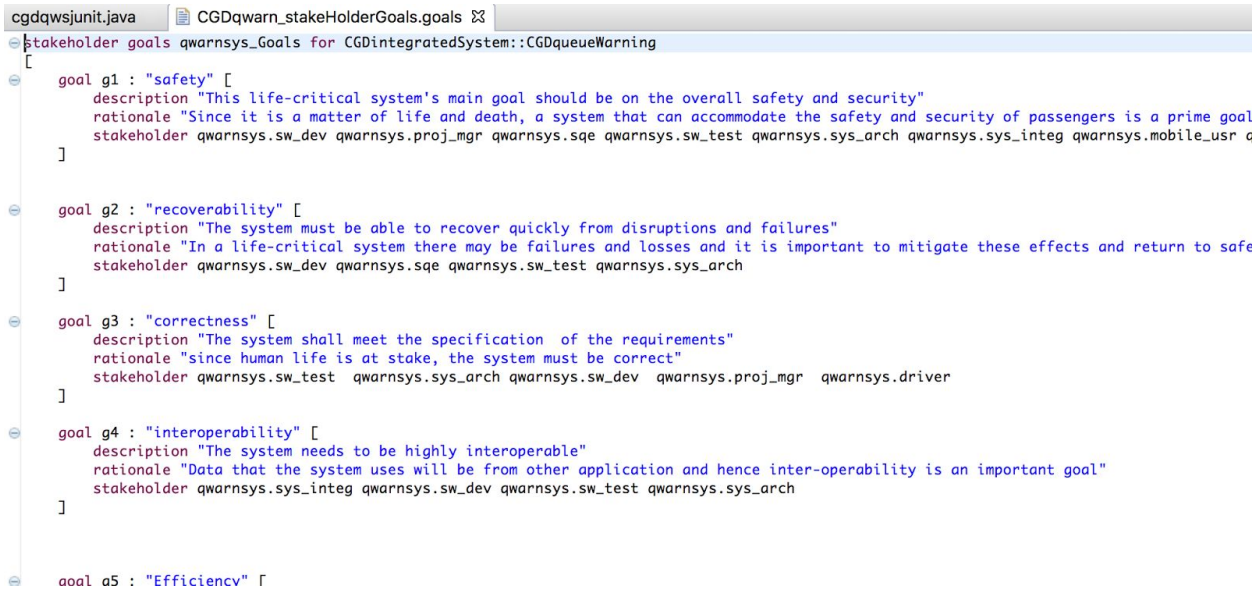

Fig.7.CGDqwarn_stakeHolderGoals.goals

Also, stakeholders of CGDQWS is shown below:

Fig.8.Stakeholders

System requirement acts as a contract which must be met by the system implementation, hence each requirement must be verifiable. In AADL we use ReqSpec to specify requirements. ReqSpec allows requirements to be formated in a document structure. Requirements can be sub divided into sub requirements. ReqSpec for clemson game day queue warning system is shown below:

CGDqwarnRequirements.reqspec

```
system requirements qwarnsys_requirements for CGDintegratedSystem::CGDqueueWarning
[
    val minimunQueue= 7.00
    val maximumSpeed=70.00
    val minimumData=2.00
    val thresholdQueue=12.00
    val minimumStrength=4.50
    val minwarnings=0.0
    val minimumresponseTime=1.10
    val maxallowedresponsetime=3.50
    val minimumQueueforCommunication=15.00
    val minimumAccuracyLevel=0.05



    requirement signalStrength : "The vehicle OBE and sensors shall have high signal strength to rece
    [
        description this " The sensors shall be able to send data to a high degree of precision"
        compute signalStrength:real
        value predicate  minimumStrength <= signalStrength
        rationale "This is a critical component because the system is dependent on the sensors in orc
        see goal qwarnsys_Goals.g1 qwarnsys_Goals.g5 qwarnsys_Goals.g6
        issues "Integrity Level 4"
    ]

    requirement operationalFailure : "The system must continue to operate even in the case of failure
    [
        description this "system must be able to operate even in the event of failure minimizing harm
        compute systemResponseTime:real
        value predicate  minimumresponseTime >= systemResponseTime
        rationale "the control systems may fail and this system must cope by either going to a safe s
        mitigates "damage caused due to failure of components"
        see goal qwarnsys_Goals.g1 qwarnsys_Goals.g2
        issues "Integrity Level 5"
    ]

    requirement detectQueueFormation : "The system shall accurately detect queue formation"
```

Fig.9.CGDqwarnRequirements.reqspec

Product Components

The physical components of the Clemson Game Day Queue Warning system are listed below:

1. Driver: Driver is the person who operates the vehicle on the roadway. Driver sends driver requests and receives driver information that reflects the interaction which is helpful to drivers regardless of vehicle type. Fig.7. Shows the aadl of driver.

CGDdriver.aadl

```
package CGDdriver
public
    with error_library;

    abstract cgdDriver
        features
            driverInput_out : out data port ;
            driverInfo_in : in data port;
            driverUpdates_in : in data port ;
        flows
            input_source : flow source driverInput_out;
            info_sink : flow sink driverInfo_in;
            update_sink : flow sink driverUpdates_in;

        annex EMV2 {**
            use types error_library;

            error propagations
                driverInput_out : out propagation {NoValue, BadValue, LateValue};
                driverInfo_in : in propagation {NoValue, BadValue, LateValue};
                driverUpdates_in : in propagation {NoValue, BadValue, LateValue};
            flows
                ef0 : error source driverInput_out{NoValue};
                ef1 : error sink driverInfo_in{NoValue, BadValue, LateValue};
                ef2 : error sink driverUpdates_in{NoValue, BadValue, LateValue};
            end propagations;
            properties
                emv2::hazards =>
                    ([failure => "Novalue";
                    description => "No data from the vehicle driver";
                    ])
            applies to driverInput_out.novalue;
        **};

    end cgdDriver;

    abstract implementation cgdDriver.impl
    end cgdDriver.impl;

end CGDdriver;
```

clemson_gameday_queue_warning/CGQW_Models/CGDdriver.aadl

Fig.7.CGDdriver.aadl

2. ITS Roadway Equipment: The components under this include traffic detectors,traffic signals,highway advisory radios,dynamic message signs, CCTV cameras and video image processing system, grade crossing warning systems. ITS roadway equipment represents the ITS equipment that is distributed on and along the roadway that monitors and controls traffic.IT also monitors and manages roadway itself. Fig.8. Shows the aadl of ITS roadway equipment.

CGDdriver.aadl | CGDitsRoadwayEquipment.aadl

```
package CGDitsRoadwayEquipment
public
    with EMV2;

    system cgditsrequip
        features
            trafficSensorandVideoControl_in : in data port;
            rodwayWarningSystemControl_in : in data port;
            environmentSensorControl_in : in data port;
            trafficSituationData_in : in data port;
            trafficFlowImgs_out : out data port;
            rdwayWarnSysStat_out : out data port;
            envSensData_out : out data port;
            vehicleSignageLocalData_out : out data port;
            DriverInfo_out : out data port;

        flows
            system_sensorandVideovc_sink : flow sink trafficSensorandVideoControl_in;
            system_warsyscntrl_sink : flow sink rodwayWarningSystemControl_in;
            system_envsenscntrl_sink : flow sink environmentSensorControl_in;
            system_situ_sink : flow sink trafficSituationData_in;
            system_flimg_source : flow source trafficFlowImgs_out;
            system_rdwarnstat_source : flow source rdwayWarnSysStat_out;
            system_sendata_source : flow source envSensData_out;
            system_vsign_source : flow source vehicleSignageLocalData_out;
            system_dinfo_source : flow source DriverInfo_out;


        annex EMV2 {**
            use types error_library;

            error propagations
                trafficSensorandVideoControl_in : in propagation {NoValue, BadValue, LateValue};
                rodwayWarningSystemControl_in : in propagation {NoValue,BadValue, LateValue};
                environmentSensorControl_in : in propagation {NoValue,BadValue, LateValue};
                trafficSituationData_in : in propagation {NoValue,BadValue, LateValue};
                trafficFlowImgs_out : out propagation {NoValue,BadValue, LateValue};
                rdwayWarnSysStat_out : out propagation {NoValue,BadValue, LateValue};
                envSensData_out : out propagation {NoValue,BadValue, LateValue};
                DriverInfo_out : out propagation {NoValue,BadValue, LateValue};
            flows
                ef0 : error sink trafficSensorandVideoControl_in{NoValue,BadValue, LateValue};
                ef1 : error sink rodwayWarningSystemControl_in{NoValue,BadValue, LateValue};
                ef2 : error sink environmentSensorControl_in{NoValue,BadValue, LateValue};
                ef3 : error sink trafficSituationData_in{NoValue,BadValue, LateValue};
                ef4 : error source trafficFlowImgs_out{NoValue,BadValue, LateValue};
```

Fig.8.CGDitsRoadwayEquipment.aadl

3. Other Traffic Management Centers: This helps in information exchange by providing a source and destination between peer traffic management functions. Fig.9. Shows the aadl of other traffic management center.

CGDdriver.aadl | CGDitsRoadwayEquipment.aadl | CGDotherTrafficMgmtCenters.aadl

```
package CGDotherTrafficMgmtCenters
public
    with EMV2;

    system cgdothertrafmgmtcent

        features
        roadNetworkConditions_in : in data port;
        roadNetworkConditions_out : out data port;

        flows
        rdNetcond_sink : flow sink roadNetworkConditions_in;
        rdNetcond_source : flow source roadNetworkConditions_out;

        annex EMV2 {**
            use types error_library;

            error propagations
                roadNetworkConditions_in : in propagation {NoValue, BadValue, LateValue};
                roadNetworkConditions_out : out propagation {NoValue,BadValue, LateValue};

            flows
                ef0 : error sink roadNetworkConditions_in{NoValue,BadValue, LateValue};
                ef1 : error source roadNetworkConditions_out{NoValue,BadValue, LateValue};
            end propagations;
            properties
                emv2::hazards =>
                    ([failure => "ValueError";
                    description => "No data , Invalid date or Late data from Traffic Management Center";
                    ])
            applies to roadNetworkConditions_in.ValueError,roadNetworkConditions_out.ValueError;
        **};

    end cgdothertrafmgmtcent;

    system implementation cgdothertrafmgmtcent.impl
    end cgdothertrafmgmtcent.impl;
end CGDotherTrafficMgmtCenters;
```

Fig.9.CGDotherTrafficMgmtCenters.aadl

4. Remote Vehicle OBE: This depicts other connected vehicles that communicates with host vehicle, including passenger cars, trucks etc. Also support basic vehicle OBE functions.

Provides a source and destination for information transfers between connected vehicles.Fig.10. Shows the aadl of remote vehicle obe.

CGDdriver.aadl | CGDitsRoadwayEquipment.aadl | CGDotherTrafficMgmtCenters.aadl | CGDremoteVehicleOBE.aadl

```
package CGDremoteVehicleOBE
public
    with EMV2;

    system cgdremvehobe

        features
            vehicleControlEventLocationMotion_out : out data port;
            vehicleConttrolEventLocationMotion_in : in data port;

        flows
            system_vcntrleventloc_source : flow source vehicleControlEventLocationMotion_out;
            system_vcntrleventlocmot_sink : flow sink vehicleConttrolEventLocationMotion_in;

        annex EMV2 {**
            use types error_library;

            error propagations
                vehicleConttrolEventLocationMotion_in : in propagation {NoValue, BadValue, LateValue};
                vehicleControlEventLocationMotion_out : out propagation {NoValue,BadValue, LateValue};

            flows
                ef0 : error sink vehicleConttrolEventLocationMotion_in{NoValue,BadValue, LateValue};
                ef1 : error source vehicleControlEventLocationMotion_out{NoValue,BadValue, LateValue};
            end propagations;
            properties
                emv2::hazards =>
                    ([failure => "ValueError";
                    description => "No data , Invalid date or Late data from Remote Vehicle OBEs annd to Roadside Equipment";
                    ])
            applies to vehicleConttrolEventLocationMotion_in.ValueError,vehicleControlEventLocationMotion_out.ValueError;
        **};
    end cgdremvehobe;

    system implementation cgdremvehobe.impl
    end cgdremvehobe.impl;

end CGDremoteVehicleOBE;
```

Fig.10.CGDremoteVehicleOBE.aadl

5. Roadside Equipment: This constitutes the connected vehicle roadside devices that sends messages to and receives messages from, nearby vehicles using Dedicated Short Range Communications. It can be used to operate from a fixed location and can be permanently installed or can be used as a portable device which is temporarily located in traffic vicinity. It has various subcomponents such as data storage, processor etc.Fig.11. Shows the aadl of roadside equipment.

CGDroadsideEquipment.aadl

clemson_gameday_queue_warning/
CGQW_Models/CGDroadsideEquipment.aadl

```
with EMV2;

system cgdrodequip
    features
        queueWarningApplicationInformation_in : in data port;
        vehicleSignageLocalData_in : in data port;
        veEnvironmentData_in : in data port;
        veControlEventLocationMotion_in : in data port;
        trafficSituationDataEnvData_out: out data port;
        queueWarnAppStatus_out : out data port;
        trafSituationData_out : out data port;
        qWarnInfo_out : out data port;
        vehSignData_out : out data port;

    flows
        system_qwarnapp_sink : flow sink queueWarningApplicationInformation_in;
        system_vsignloc_sink : flow sink vehicleSignageLocalData_in;
        system_venv_sink : flow sink veEnvironmentData_in;
        system_vcntrlloc_sink : flow sink veControlEventLocationMotion_in;
        system_trsituenv_source : flow source trafficSituationDataEnvData_out;
        system_qwarnappstat_source : flow source queueWarnAppStatus_out;
        system_trasitu_source : flow source trafSituationData_out;
        system_qwarninfo_source : flow source qWarnInfo_out;
        system_vsign_source : flow source vehSignData_out;

    annex EMV2 {**
        use types error_library;

        error propagations
            queueWarningApplicationInformation_in : in propagation {NoValue, BadValue, LateValue};
            vehicleSignageLocalData_in : in propagation {NoValue,BadValue, LateValue};
            veEnvironmentData_in : in propagation {NoValue,BadValue, LateValue};
            veControlEventLocationMotion_in : in propagation {NoValue,BadValue, LateValue};
            trafficSituationDataEnvData_out : out propagation {NoValue,BadValue, LateValue};
            queueWarnAppStatus_out : out propagation {NoValue,BadValue, LateValue};
            trafSituationData_out : out propagation {NoValue,BadValue, LateValue};
            qWarnInfo_out : out propagation {NoValue,BadValue, LateValue};
            vehSignData_out : out propagation {NoValue,BadValue, LateValue};
        flows
            ef0 : error sink queueWarningApplicationInformation_in{NoValue,BadValue, LateValue};
            ef1 : error sink vehicleSignageLocalData_in{NoValue,BadValue, LateValue};
            ef2 : error sink veEnvironmentData_in{NoValue,BadValue, LateValue};
            ef3 : error sink veControlEventLocationMotion_in{NoValue,BadValue, LateValue};
```

Fig.11.CGDroadsideEquipment.aadl

6. Traffic Management Center: This is used to monitor and control traffic and road network. It represents the range of transportation facilities including freeway systems, rural and highway systems. It communicates with ITS roadway equipment and connected vehicle roadside equipment to monitor and manage traffic flow.Fig.12. Shows the aadl of traffic management center.

CGDroadsideEquipment.aadl | CGDtraffMgmtCenter.aadl

clemson_gameday_queue_warning/CGQW_Models/CGDroadsideEquipment.aadl

```
with EMV2;

system cgdtrafcenter
    features
        trafficFlowImgs_in: in data port;
        roadwayWarningSystemStatus_in: in data port;
        environmentSensorData_in: in data port;
        trafficEnvironmentSituationData_in: in data port;
        queueWarnApplicationStatus_in: in data port;
        roadNetworkConditions_in: in data port;
        trafficOperationInputp_in: in data port;
        trafficSensorCtrlVidSurvCtrl_out : out data port;
        roadWarnSystemCtrl_out : out data port;
        environmentSensorsCtrl_out : out data port;
        queueWarnApplicationInfo_out : out data port;
        roadNetworkConditions_out : out data port;
        roaddNetworkConditionsforTranspInfoCenter_out : out data port;
        trafficOperationData_out : out data port;

    flows
        system_trflowimg_sink : flow sink trafficFlowImgs_in;
        system_rwarnsys_sink : flow sink roadwayWarningSystemStatus_in;
        system_envsens_sink : flow sink environmentSensorData_in;
        system_trenvsitu_sink : flow sink trafficEnvironmentSituationData_in;
        system_qwarnapp_sink : flow sink queueWarnApplicationStatus_in;
        system_rdnetcond_sink : flow sink roadNetworkConditions_in;
        system_tropinp_sink : flow sink trafficOperationInputp_in;
        system_trsenscntrlvidsurv_source : flow source trafficSensorCtrlVidSurvCtrl_out;
        system_rwarnsyscntrl_source : flow source roadWarnSystemCtrl_out;
        system_envsenscntrl_source : flow source environmentSensorsCtrl_out;
        system_qwarnappinfo_source : flow source queueWarnApplicationInfo_out;
        system_roadnetcond_source : flow source roadNetworkConditions_out;
        system_rdnetcondforothers_source: flow source roaddNetworkConditionsforTranspInfoCenter_out;
        system_trafopdata_source : flow source trafficOperationData_out;

    annex EMV2 {**
        use types error_library;

        error propagations
            trafficFlowImgs_in : in propagation {NoValue, BadValue, LateValue};
            roadwayWarningSystemStatus_in : in propagation {NoValue,BadValue, LateValue};
            environmentSensorData_in : in propagation {NoValue,BadValue, LateValue};
            trafficEnvironmentSituationData_in : in propagation {NoValue,BadValue, LateValue};
            queueWarnApplicationStatus_in : in propagation {NoValue,BadValue, LateValue};
```

Fig.12.CGDtraffMgmtCenter.aadl

7. Traffic Operations Personnel: Personnels interact with other traffic control systems, surveillance systems. They provide command inputs to direct system operations and operate data to varying levels based on the system type and deployment scenario.Fig.13. Shows the aadl of traffic operations personnel.

CGDroadsideEquipment.aadl | CGDtraffMgmtCenter.aadl | CGDtraffOppPersonel.aadl

```
package CGDtraffOppPersonel
public
    with EMV2;

    system cgdtrafOpPers
        features
            trafficOperationInput_out :out data port;
            trafficOperationData_in : in data port;

        flows
            trafopdata_sink : flow sink trafficOperationData_in;
            trafopinp_source : flow source trafficOperationInput_out;

        annex EMV2 {**
            use types error_library;

            error propagations
                trafficOperationData_in : in propagation {NoValue, BadValue, LateValue};
                trafficOperationInput_out : out propagation {NoValue,BadValue, LateValue};

            flows
                ef0 : error sink trafficOperationData_in{NoValue,BadValue, LateValue};
                ef1 : error source trafficOperationInput_out{NoValue,BadValue, LateValue};
            end propagations;
            properties
                emv2::hazards =>
                    ([failure => "ValueError";
                    description => "No data , Invalid date or Late data from and to Traffic Management Center";
                    ])
            applies to trafficOperationData_in.ValueError,trafficOperationInput_out.ValueError;
        **};
    end cgdtrafOpPers;

    system implementation cgdtrafOpPers.impl
    end cgdtrafOpPers.impl;

end CGDtraffOppPersonel;
```

Fig.13.CGDtraffOppPersonel.aadl

8. Traffic Information Center: All traffic data are collected, processed, stored by traffic information center and these informations are disseminated to system operators and public. This physical object plays various roles such as provides data collection, fusing and collecting information from transportation system operators and redistributes the information to other system operators and other TICs. Fig.14. Shows the aadl of traffic information center.

CGDroadsideEquipment.aadl | CGDtraffMgmtCenter.aadl | CGDtraffOppPersonel.aadl | CGDtransInfoCenter.aadl

```
package CGDtransInfoCenter
public
    with EMV2;

    system cgdtransinfocent
        features
            roadNetworkConditions_in : in data port;

        flows
            rdnetcond_source : flow sink roadNetworkConditions_in;

        annex EMV2 {**
            use types error_library;

            error propagations
                roadNetworkConditions_in : in propagation {NoValue, BadValue, LateValue};

            flows
                ef0 : error sink roadNetworkConditions_in{NoValue,BadValue, LateValue};
            end propagations;
            properties
                emv2::hazards =>
                    ([failure => "ValueError";
                    description => "No data , Invalid date or Late data from Traffic Management Center";
                    ])
            applies to roadNetworkConditions_in.ValueError;
        **};
    end cgdtransinfocent;

    system implementation cgdtransinfocent.impl
    end cgdtransinfocent.impl;

end CGDtransInfoCenter;
```

Fig.14.CGDtransinfoCenter.aadl

9. Vehicle Databus: This acts as the interface to vehicle data bus which enable communication between vehicle OBE and other vehicle systems. Vehicle system status, sensor outputs available on databus vary based on equipment installed on the vehicle and availability on data bus. This component is also used to indicate onboard interactions. Fig.15. Shows the aadl of vehicle databus.

CGDvehicleDatabus.aadl | CGDvehicleOBE.aadl

```
package CGDvehicleDatabus
public
    with EMV2;

    system cgdvehicledbus
        features
            collisionWarnInfoDrivUpdateInfo_in : in data port;
            hostvehStatDrivInpInfo_out : out data port;

        flows
            system_colwarninfodupdate_sink : flow sink collisionWarnInfoDrivUpdateInfo_in;
            system_hvstatdinpinfo_source : flow source hostvehStatDrivInpInfo_out;

        annex EMV2 {**
            use types error_library;

            error propagations
                collisionWarnInfoDrivUpdateInfo_in : in propagation {NoValue, BadValue, LateValue};
                hostvehStatDrivInpInfo_out : out propagation {NoValue,BadValue, LateValue};

            flows
                ef0 : error sink collisionWarnInfoDrivUpdateInfo_in{NoValue,BadValue, LateValue};
                ef1 : error source hostvehStatDrivInpInfo_out{NoValue,BadValue, LateValue};
            end propagations;
            properties
                emv2::hazards =>
                    ([failure => "ValueError";
                    description => "No data , Invalid date or Late data from and to Vehicle OBE";
                    ])
            applies to collisionWarnInfoDrivUpdateInfo_in.ValueError,hostvehStatDrivInpInfo_out.ValueError;
        **};
    end cgdvehicledbus;

    system implementation cgdvehicledbus.impl
    end cgdvehicledbus.impl;
end CGDvehicleDatabus;
```

Fig.15.CGDvehicleDatabus.aadl

10. Vehicle On-Board Equipment: This provides the vehicle based processing , storage and communications functions necessary to support connected vehicle operations.V2V and V2I are 2 important component of vehicle OBE. Communication platform is augmented with processing and data storage capability.Fig.16. Shows the aadl of vehicle on board equipment.

CGDvehicleDatabus.aadl | CGDvehicleOBE.aadl

clemson_gameday_queue_warning/
CGQW_Models/CGDvehicleDatabus.aadl

```
with EMV2;

system cgdvobe
    features
        queueWarningInformation_in:in data port;
        vehicleSignageData_in : in data port;
        vehicleControlEventLocationMotion_in : in data port;
        driverInput_in : in data port;
        hostVehicleStatuseDriverInputInformation_in : in data port;
        vehEnvironmentData_out: out data port;
        vehControlEventLocationMotionForSurv_out : out data port;
        vehCtrlEventLocMotion2_out : out data port;
        driverUpdates_out : out data port;
        collisionWarnDriverUpdate_out : out data port;

    flows
        system_qwarninfo_sink : flow sink queueWarningInformation_in;
        system_vsign_sink : flow sink vehicleSignageData_in;
        system_vcntrlloc_sink : flow sink vehicleControlEventLocationMotion_in;
        system_dinput_sink : flow sink driverInput_in;
        system_hvstatdrip_sink : flow sink hostVehicleStatuseDriverInputInformation_in;
        system_venv_source : flow source vehEnvironmentData_out;
        system_vcntrlloc_source : flow source vehControlEventLocationMotionForSurv_out;
        system_vcntrllocmot2_source : flow source vehCtrlEventLocMotion2_out;
        system_dupdate_source : flow source driverUpdates_out;
        system_colwarndupdate_source : flow source collisionWarnDriverUpdate_out;

    annex EMV2 {**
        use types error_library;

        error propagations
            queueWarningInformation_in : in propagation {NoValue, BadValue, LateValue};
            vehicleSignageData_in : in propagation {NoValue,BadValue, LateValue};
            vehicleControlEventLocationMotion_in : in propagation {NoValue,BadValue, LateValue};
            driverInput_in : in propagation {NoValue,BadValue, LateValue};
            hostVehicleStatuseDriverInputInformation_in : in propagation {NoValue,BadValue, LateValue};
            vehEnvironmentData_out : out propagation {NoValue,BadValue, LateValue};
            vehControlEventLocationMotionForSurv_out : out propagation {NoValue,BadValue, LateValue};
            vehCtrlEventLocMotion2_out : out propagation {NoValue,BadValue, LateValue};
            driverUpdates_out : out propagation {NoValue,BadValue, LateValue};
            collisionWarnDriverUpdate_out : out propagation {NoValue,BadValue, LateValue};
        flows
            ef0 : error sink queueWarningInformation_in{NoValue,BadValue, LateValue};
```

Fig.16 CGDvehicleOBE.aadl

Product (CGDQWS)

The expected final product of the project endeavor is a Queue Warning System customized for game days on Clemson University campus and environs. A detailed architecture of a queue warning system created by CVRIA shows the interaction and data flow among the participating components. Major components of the system have been highlighted in the section above.

The IMV diagram of the integrated system from OSATE is shown below:

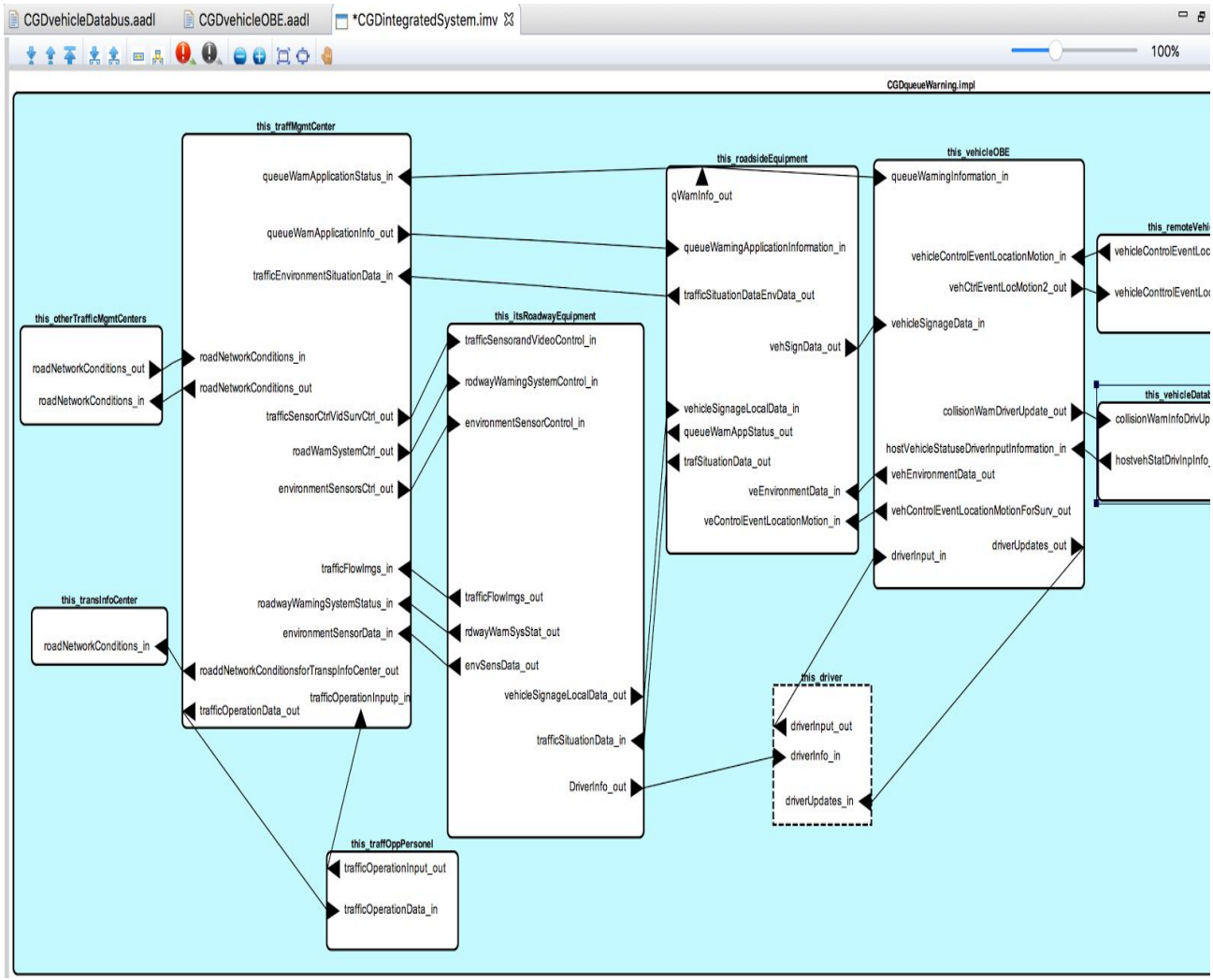

Fig. 17.CGDintergratedSystem.imv

In the AADL model of the system, the final product is represented by the CGDintegratedSystem.aadl as shown in fig.18.

CGDqwarnRequirements.reqspec | CGDintegratedSystem.aadl

```
package CGDintegratedSystem
public

    with CGDdriver;
    with CGDitsRoadwayEquipment;
    with CGDotherTrafficMgmtCenters;
    with CGDremoteVehicleOBE;
    with CGDroadsideEquipment;
    with CGDtraffMgmtCenter;
    with CGDtraffOppPersonel;
    with CGDtransInfoCenter;
    with CGDvehicleDatabus;
    with CGDvehicleOBE;

    system CGDqueueWarning
    end CGDqueueWarning;

    system implementation CGDqueueWarning.impl
        subcomponents

            this_driver : abstract CGDdriver::cgdDriver;
            this_itsRoadwayEquipment : system CGDitsRoadwayEquipment::cgditsrequip.impl;
            this_otherTrafficMgmtCenters : system CGDotherTrafficMgmtCenters::cgdothertrafmgmtcent.im
            this_roadsideEquipment : system CGDroadsideEquipment::cgdrodequip.impl;
            this_traffMgmtCenter : system CGDtraffMgmtCenter::cgdtrafcenter.impl;
            this_traffOppPersonel : system CGDtraffOppPersonel::cgdtrafOpPers.impl;
            this_transInfoCenter : system CGDtransInfoCenter::cgdtransinfocent.impl;
            this_vehicleDatabus : system CGDvehicleDatabus::cgdvehicledbus.impl;
            this_vehicleOBE : system CGDvehicleOBE::cgdvobe.impl;
            this_remoteVehicleOBE : system CGDremoteVehicleOBE::cgdremvehobe.impl;

        connections
```

Fig. 18 Integrated CGDQWS model

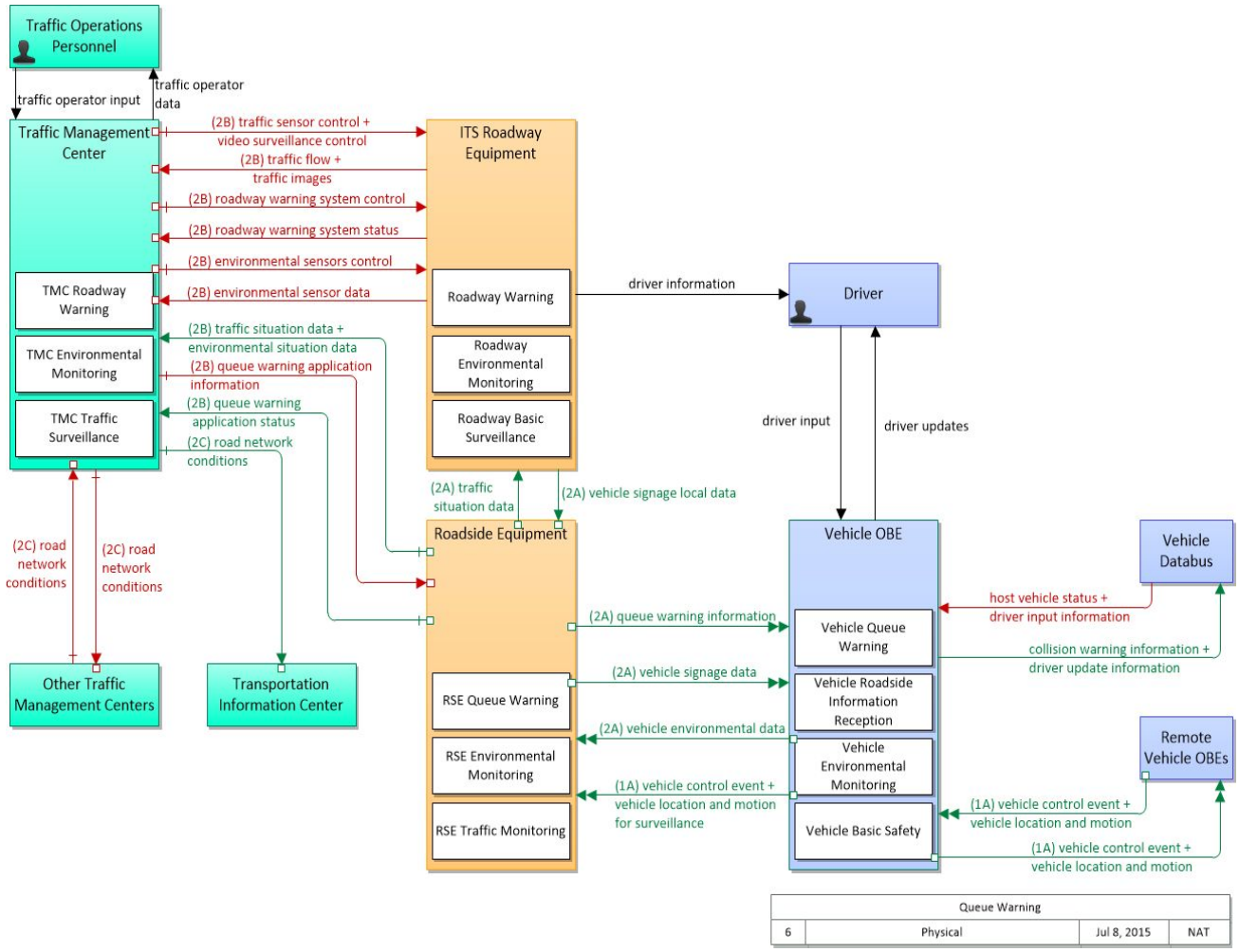

Fig. 19 Queue Warning System Architecture by CVRIA

Model Verification and Validation

The model verification for the Clemson Gameday Queue warning system was carried out using both the AGREE and JUnit framework. The rationale for this is to compare the output of both verification frameworks and decide on whether the AGREE framework outperform the JUnit in terms of coverage, performance, ease of use etc.

AGREE

The Assume Guarantee REasoning Environment uses a composition which is performed based on the assumptions and guarantees that was provided in the CGDQWS components. We created the assumptions and the guarantees based critical requirements identified in the requirement specification document. The figure below shows the results generated after executing the AGREE Verify All Layers on the implementation of the system components.

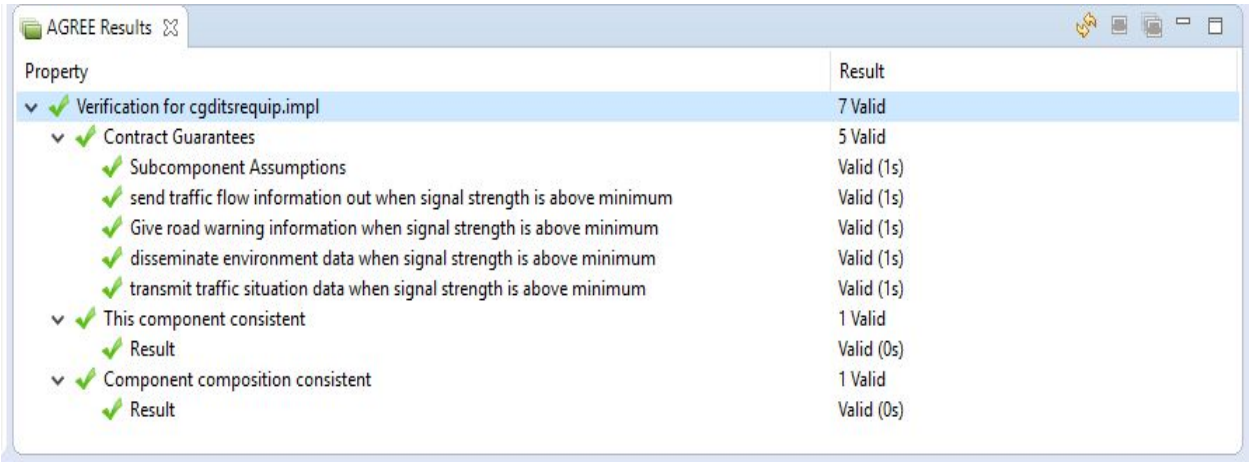

Fig. 20 Agree Results for the ITS Roadway Equipment

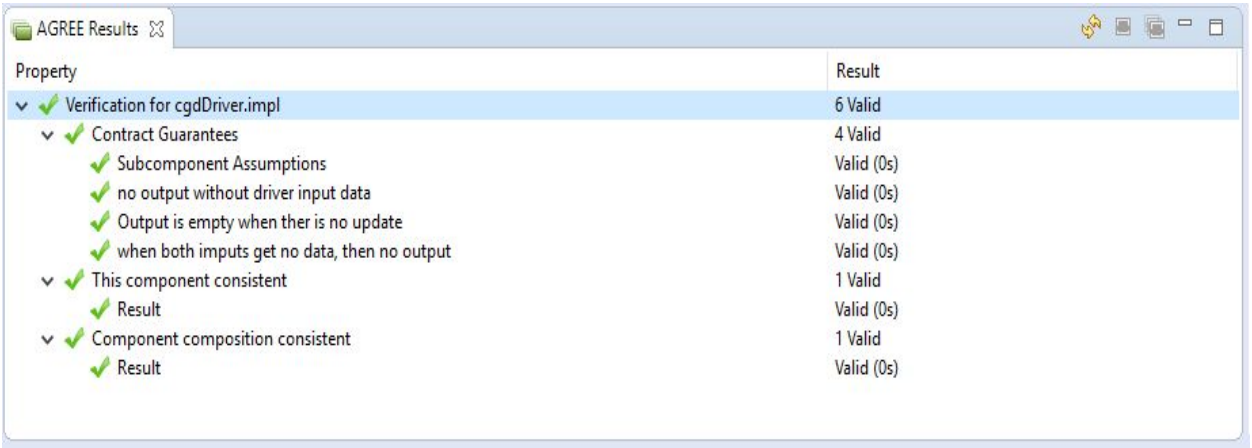

Fig. 21 Agree Results for the Driver

JUNIT

Junit is a unit testing framework used for java programming language. It promotes the idea of first testing then coding.Junit increases the stability of the program and increases the productivity of the programmer. Junit tests are supported as verification activities in ALISA, which is accomplished by registering a JUNIT test class . We have designed a junit class to test key functionality of the Clemson gameday queue warning system. Fig. 20 shows the cgdqwsjunit.java:

Console | cgdqwsjunit.java

```
package junittest;

import static org.junit.Assert.assertEquals;

public class cgdqwsjunit {



    @Test

    public void testingSignalStrength() {
        assertEquals("Signal strength is greater", signalStrength(6.0,4.65));
    }
    @Test
    public void testingOperationalFailure() {
        assertEquals("Operational failure occured", operationalFailure(0.8,1.10));
    }
    @Test
    public void testingQueueFormation() {
        assertEquals("Queue Formed", detectQueueFormation(10,7));
    }
    @Test
    public void testingIndividulized_warning() {
        assertEquals("Appropriate number of warnings", individulized_warning(6.0,0.0));
    }
    @Test
    public void testingAccurate_system() {
        assertEquals("Good accuracy level", accurate_system(1.0,0.05));
    }
    @Test
    public void testingAccurateSpeed() {
        assertEquals("Accurate speed maintained", accurateSpeed(70.00,5.00,80.00,7.00));
    }
    @Test
    public void testingPredictQueue() {
        assertEquals("Queue Predicted", predictQueue(6.0,4.65));
    }
    @Test
    public void testingGenerate_respStrategies() {
        assertEquals("Good response time", generate_respStrategies(6.0,1.10));
    }

    public String signalStrength(double signalStrength,double minimumStrength) {
        if(minimumStrength <= signalStrength){
            System.out.println("Signal strength is greater than minimum value");
            return "Signal strength is greater";
```

Fig.20. Cgdqwsjunit.java

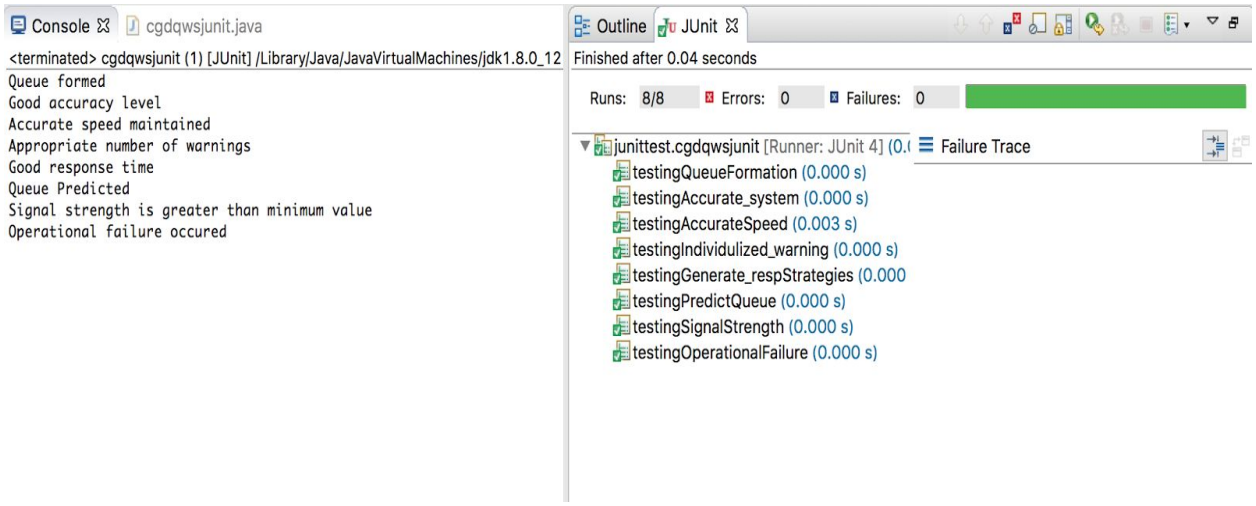

Fig.21 Junit execution result

Conclusion

In this project we have demonstrated the use of Software Verification and Validation Tools and Techniques in modeling a queue warning system customized for use on Clemson game days. We started by creating various work products necessary to accomplish the objective. Furthermore, we modeled every subsystems and components based on the identified system requirement specification. We evaluated the model using both the AGREE and JUnit frameworks. We discovered that the AGREE model checking activities seems easy to develop compared to other model checking frameworks like Resolute and JUnit.