Software Requirements Specification (SRS)

Autonomous Steering (EPAS)

Authors: Gregory Richard, Evan Swinehart, Tim Sloncz, John Adams, Weilong Li

Customer: Chrysler

Instructor: Dr. Betty Cheng

1 Introduction

This document provides an overview for the software requirements specification for an autonomous steering module. This document covers multiple sections including an introduction, overview, overall description of the system that is being specified, and finally a specification of requirements for the system.

1.1 Purpose

The purpose of this document is to outline and overview the description and requirements for the autonomous steering module. This document contains the purpose and features of the system that will help explain the interfaces and workings of the system. Another important part of this document is the constraints in which the system is designed and how it will react and operate in different situations with different inputs. This document is intended for system developers to fully understand the system that will be put into place.

1.2 Scope

The Electrical Power Steering System (EPAS) will be used in future Chrysler vehicles. The EPAS will be used to control power steering based on an autonomous module's input along with various other inputs. The basis of this system will later be used for autonomous driving and steering. This system can improve safety and ease of driving by taking input from an autonomous module.

This system will be embedded within the Electronic Control Unit (ECU) network of a vehicle. Communication between this system, its inputs and outputs will be done via the Controller Area Network (CAN) bus. This system is not designed to work outside of this context.

System output is calculated with multiple inputs from various sources; including speed sensors and autonomous modules in order to produce outputs for power steering. This system is not designed to take complete control of the vehicle and will ramp down any output upon user input.

1.3 Definitions, acronyms, and abbreviations

Term	Definition
User	A person who interacts with the system.
Driver	A user who operates a vehicle utilizing this system.
Rack / Pinion	A linear actuator that converts rotational motion into linear motion.
ECU	Electronic Control Unit – An embedded computer that controls a system of a vehicle.
CAN	Controller Area Network – An automotive network standard used to pass information between ECUs.
EPAS	Electrical Power Steering System – A specialized ECU that controls the steering of a vehicle.
PWM	Pulse Width Modulation is used to carry to information about the torque needed to be output.
SRS	A document that provides an overview of all system components.
Normal Control	A control mode in which the driver is in complete control of the vehicle and is only being provided assisted torque from the EPAS system.
Autonomous Control/ Autonomous Driving	A control mode in which the EPAS controls the complete steering of the vehicle with no need for driving input.
Autonomous Driver Assist Modules	These represent the different modes of autonomous functionality the Autonomous Control can handle, that includes Lane Assist, Park Assist, Traffic Jam Assist, and Emergency Object Avoidance.
Autonomous Control Torque	This is torque created as output from the Autonomous Control.

1.4 Organization

The document will be organized as follows: in the section one, this section, it will outline the SRS document and what to expect. In the next section, section two, an overall description of the system will be given in order to better orient readers on the specifics of the system. It will also go over some functions and constraints of the system. The third section outlines the specific requirements for the system that will need to be met for the system to be successful. In the fourth section, models are introduced to demonstrate the static and dynamic aspects of the requirements. Next, the fifth section shows a prototype of the system with various sample uses to explain how a final system would behave. Nearing the end in the sixth section, an array of references for further reading about different parts of the system is provided. Then lastly in the final section a point of contact will be provided for any further questions regarding the project.

2 Overall Description

This section gives an overview and explanation of the system being developed. This section covers the context of the system along with what parts are important. In addition it covers constraints the system will have to work with. This section also covers product functions and expectations of users.

2.1 Product Perspective

This system will be integrated with existing steering systems on a vehicle. The system will be part of a control program embedded in the steering ECU. This system can take inputs from other modules via the CAN bus.

The system will take inputs from various sources that are not directly controlled by the driver such as speed sensors, and also take inputs directly from the driver such as steering angle. Steering angle will be measured at the steering column from the angle that the driver turns the steering wheel. Inputs from the outside sources will be sent via the CAN bus to the ECU. All calculations must be done within the ECU and within a 500 milliseconds so that the operation of the vehicle is not interrupted. The system must not have any single point of failure that would lead to an incorrect output at the power steering motor.

2.2 Product Functions

This project's purpose is to develop an EPAS controller that can be used in the implementation of autonomous driving systems.

The EPAS consists of torque sensors, an EPAS controller, an electrical motor, rack, pinions and a steering column as shown in Figure 1. The control is performed by reading the torque sensor and providing assisting torque based on driver's torque on the steering wheel.

The actual implementation of this control uses vehicle speed and an interpolation table to modify the amount of torque assist provided. The actual output to the motor is a pulse width modulation signal and not a fixed voltage.

To facilitate autonomous functions that will drive the vehicle in specific and limited operations, various inputs from Autonomous Driver Assist Modules are needed. The autonomous driving function will require that the EPAS be in control of the "steering angle" of the vehicle. In order to control the steering angle, a closed loop control will be implement. The result will be used to generate a torque command to the wheel.

To communicate the steering angle commands from the Autonomous Driver Assist Modules the CAN bus is used. Due to safety and security concerns, validity, encryption and security restriction will be added to the messages packets with critical data content.

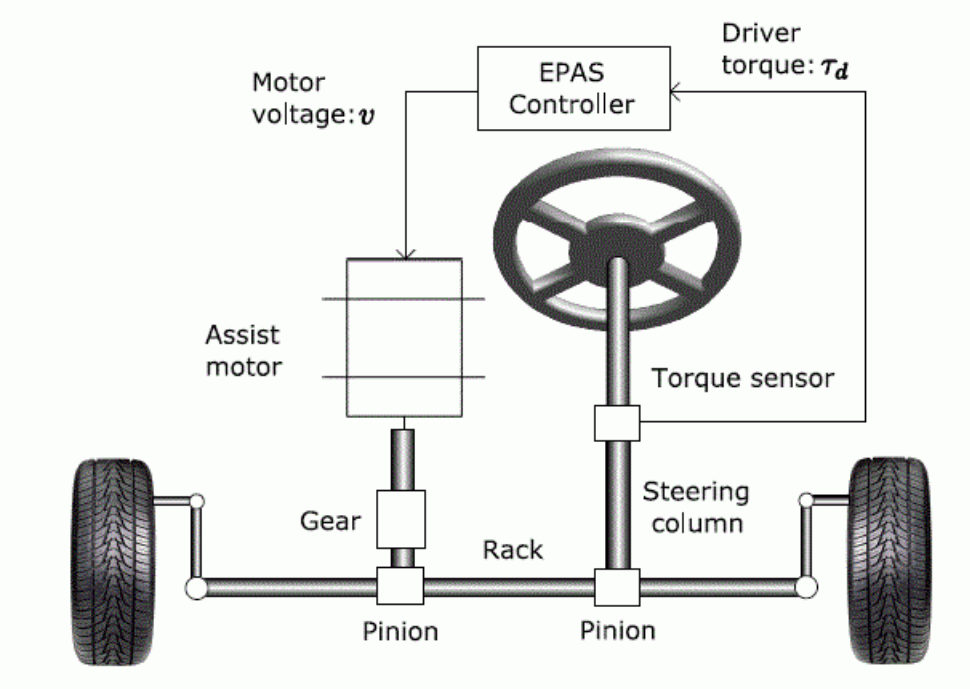

Figure 1: Steering system overview

2.3 User Characteristics

Users of this system do not need any previous knowledge of the system to safely operate the vehicle. Users are expected to know how to properly operate a motor vehicle. User input is only based on torque and vehicle speed that is in line with normal motor vehicle operation.

2.4 Constraints

Some of the multiple constraints throughout our project consist of safety critical properties our system must have, defined communication standards, and proper security implementations for safety.

Safety critical properties:

The driver shall be able to take over the control of the steering wheel from the Autonomous Control System at anytime by turning the system off with a switch or by simply applying a rotational force on the wheel. After the force disappears, within 500 milliseconds, the Autonomous Control System shall resume in control. Control will only return to the Autonomous Control System if the system override switch is not active.

When changing from Autonomous Control to Normal Control, the system shall ramp down from Autonomous Control torque to the torque required by for Normal Control within 500 milliseconds. If the Autonomous Control is running, it shall execute with a low failure rate. In order to achieve this low failure rate and the safety requirements, all circuits that could result in a single point of failure need to be monitored, and once a fault is detected, the effect shall be mitigated by termination of the function or invoking other alternative circuits.

As for invariants of the system, they are critical to prevent unrealistic values from affecting the EPAS Controls. If a failure in the system occurs, then an alert will warn the driver that the system is being terminated and the driver needs to control the steering wheel manually.

Security:

The system shall include password protection in the packets with steering commands. Security measure will also restrict inputs and only allow authorized ECU's. If there are any unused communications ports then these will be disabled for safety reasons. Also if any new calibrations or program code is written, the identification number for the person making the changes to the system shall be recorded.

2.5 Assumptions and Dependencies

For the system to function properly, there are certain assumptions made about the hardware, software, environment and user interface.

Assumptions:

Various assumptions made when dealing with the EPAS system are as follows: The driver is expect to practice safe driving standards that create a non hazardous driving environment, it is assumed that the Autonomous Driver Assist Modules will conform to proper driving and safety standards of the road. The driver is also assumed to know when to properly cancel Autonomous Mode and assume control of the vehicle. Lastly it is assumed that a failure in the hardware can be detected and the system properly shut down in such a situation.

Dependencies:

The EPAS depends on various inputs, such as torque sensors, speed sensors, and proximity sensors. The system depends on the Normal Control to make decisions on what torque to output onto the steering column. However with the Autonomous Control, it depends on receiving data from the CAN bus to make decisions about what torque force to output onto the steer column.

2.6 Approportioning of Requirements

The override switch is only for development purposes. Therefore in the future, the system will determine internally when to shut down the Autonomous Control on its own. Even if the Autonomous Control is ramped down entirely, there should remain some sort of minimum insurance that assists the driver turn the wheel manually. If an accident occurs, the system should keep the programs log record of approximately the last 15 seconds, which probably includes all valuable data, like inputs from each sensor, and the output torque force, in order to detect where the failure took place. However for this project, ensuring this data is recorded is beyond our scope.

3 Specific Requirements

1. **Classification**: Safety
   1.1. **Requirement**: No single failure shall result in an “out of control” vehicle.
   1.2. **Requirement**: Invariants shall be enacted on all embedded systems such as the sensors, actuators, and controllers to prevent unrealistic values from occurring and affecting the EPAS. The maximum torque that can be applied to the steering wheel will be 5.773333 in Newton Meters. The maximum foot-pounds of force to the steering wheel will be 8.
   1.3. **Requirement**: An audible alert to the driver should occur upon a failure in the system where the driver needs to resume control without the support of the EPAS system.
   1.4. **Requirement**: The driver will have within reaching distance on the steering wheel a discreet override switch that shall disable the autonomous steering and return it to Normal Control. This will only be for development purposes.
   1.5. **Requirement**: Autonomous Control shall ramp down and be terminated in the event that the driver torques (applies any rotational force on) the steering wheel.
   1.6. **Requirement**: The system needs to interface with other systems in the vehicle to ensure no operation of one system jeopardizes another.
   1.7. **Requirement**: Changes from Autonomous Controls to Normal Control shall ramp down from Autonomous Control torque to the torque required by for Normal Control within 500 milliseconds.
   1.8. **Requirement**: All circuits that could result in a single point failure shall be monitored for identified faults.
   1.9. **Requirement**: There will be mandatory safety checks, run upon vehicle start up, to ensure the safest possible operation of the EPAS system.
   1.10. **Requirement**: An identified fault shall be mitigated by termination of the function. If the function can run on without the failed circuit and on another circuit then it will be invoked on other alternative circuits.
2. **Classification:** Communication
   2.1. **Requirement**: Implement network communications using CAN.
3. **Classification:** Controls
   3.1. **Requirement**: Two control schemes shall be utilized. Normal EPS Open Control (Normal Control) and Autonomous Function Wheel Angle Closed Loop Control (Autonomous Functions)
   3.2. **Requirement**: Normal Control shall operate using torque sensors on the steering column.
   3.3. **Requirement**: Normal Control shall use an interpolation table based on vehicle speed and driver’s input torque.
   3.4. **Requirement**: The EPAS will shut down fully if one of the torque sensors fail.
   3.5. **Requirement**: Torque sensors, used for Normal Control, will be discreetly wired.
   3.6. **Requirement**: Normal Control must be read the torque sensor and calculate the power assist every 500 microseconds.
4. **Classification:** Security
   4.1. **Requirement**: Password protection shall be included in the packet with steering commands.
   4.2. **Requirement**: Restrict inputs to only authorize ECU’s.
   4.3. **Requirement**: Require unused communications ports to be disabled.
   4.4. **Requirement**: Data transfers of program and calibration data require encryption in the transfer.

4.5. **Requirement**: When calibrations or program code is written, the identification number for the person making the change shall be recorded.

4 Modeling Requirements

Here are the various different models of the autonomous steering system to help depict the functionality and interactivity between the different parts of the system.

Use Case Diagram:

The system includes the sensors for the torque, the controller for automated steering, sensors to detect failure, a way to disable the automatic steering, warnings for failures, and a manual override button. When the user turns the wheel, it disables any automated input from a Autonomous Driver Assist Modules and then based on the torque given to the wheel it adjusts the steering of the vehicle. If a failure in the system is detected, the user is notified, and the automated steering is turned off. The manual override switch turns off any automated steering, and leaves the control up to the driver. The inputs from Driver Assist Modules are received, and then checked for validity. After validation of the module is received, if multiple inputs are received from Autonomous Driver Assist Modules, only one is used. When the EPAS controller is in use, the output is generated based on the current speed and torque of the vehicle, which is then used to adjust the steering angle of the vehicle. An overview of this system can be seen in Figure 2.

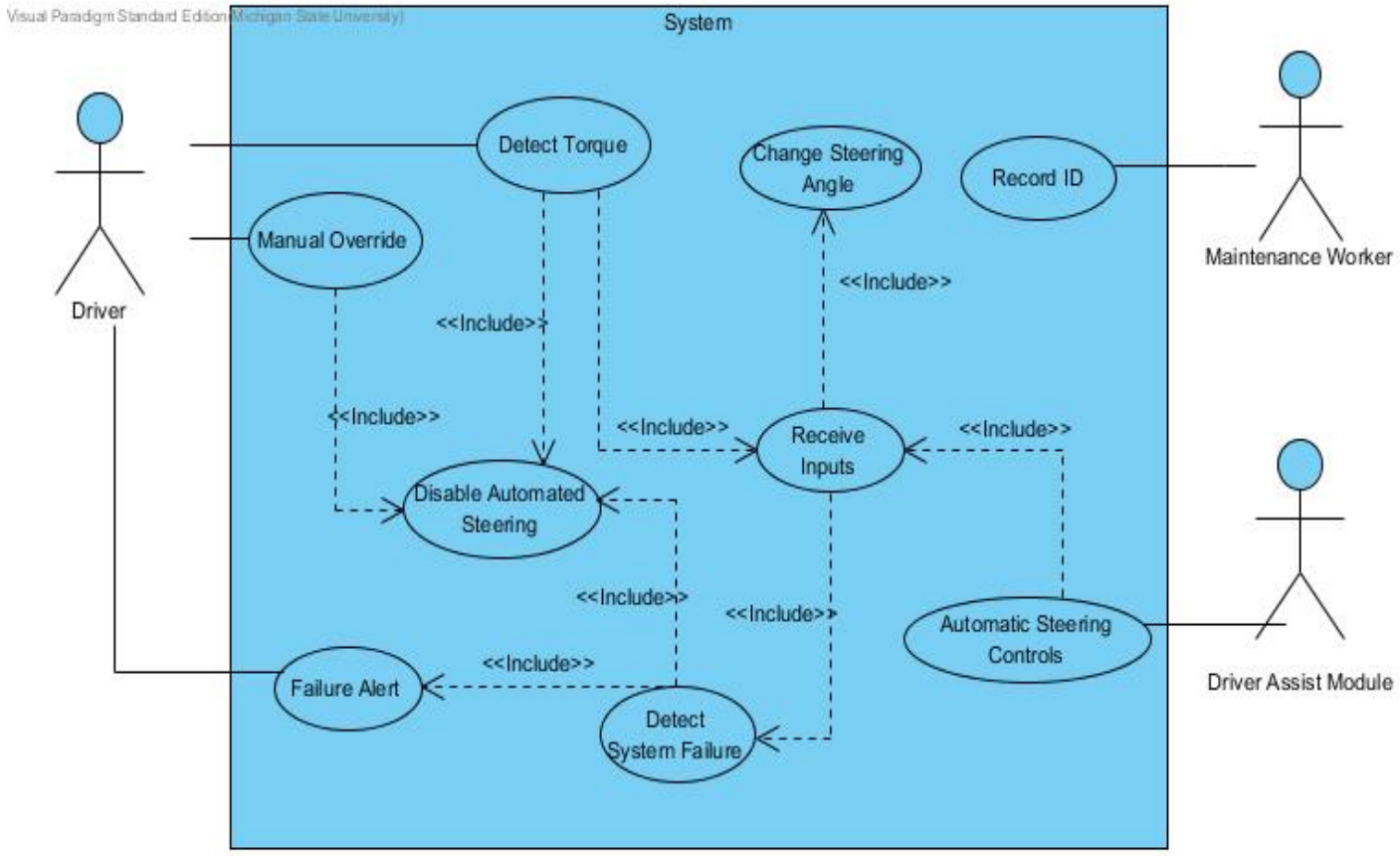

Figure 2: Use Case Diagram

Use Case	Detect Torque
Actor	Driver
Description	When the driver turns the wheel, the two torque sensors detect the torque. This will be done every 500 milliseconds in Normal Control.
Extends:	None
Cross-references Requirements	3.1, 3.2, 3.5, 3.6
Use Case	Receive Input, Disable Automated Steering

Use Case	Receive Input
Actor	None
Description	Receive the input from the Driver Assist Modules, and from the torque sensor used by Normal Control. Performs validation checks on inputs to detect defective sensors. Inputs will be sent using CAN, and packets will be encrypted and password protected to prevent outside tampering. Only inputs from valid ECUs will be accepted. Unused ports will be disabled.
Extends	None
Cross-references Requirements	1.2, 2.1. 3.1, 4.1, 4.2, 4.3, 4.4
Use Cases	Detect System Failure,

Use Case	Change Steering Angle
Actor	None
Description	Given the inputs from either the Autonomous Driver Assist Modules, or inputs from the torque applied to the steering wheel by the driver; determine how much to adjust the steering angle of the vehicle. This is done based on a function of the speed of the vehicle, how much torque should be applied, and the current angle of the vehicle.
Extends	None

Cross-references Requirements	3.1, 3.3
Use Cases	None

Use Case	Manual Override
Actor	Driver
Description	The driver can press a button on the steering wheel that will disable all automatic control of the vehicle.
Extends	None
Cross-references Requirements	1.4
Use Cases	Disable Automatic Steering

Use Case	Disable Automatic Steering
Actor	None
Description	This will disable the automated steering, causing the steering to be handled only by the driver. It will switch to Normal Control within 500 milliseconds.
Extends	None
Cross-references Requirements	1.5, 1.7, 3.1
Use Cases	None

Use Case	Detect System Failure
Actor	None
Description	Detect a system failure in a vital component, such as the torque sensor attached to the steering column, or receiving a value that is not within the intended range.
Extends	Receive Inputs
Cross-references Requirements	1.1, 1.8, 1.9, 1.10, 3.4

Use Cases	Disable Automated Steering, Failure Alert

Use Case	Failure Alert
Actor	Driver
Description	When a failure is detected, the system must shut down, an audio sound and visual light will alert the driver to the failure.
Extends	Detect System Failure
Cross-references Requirements	1.3
Use Cases	None

Use Case	Record ID
Actor	Maintenance Worker
Description	When an update is applied to the system or the system is calibrated, the ID of the person who made the changes will be recorded in the system.
Extends	None
Cross-references Requirements	4.5
Use Cases	None

Class Diagram:

The purpose of the autonomous driving system is to create calculated decisions that result in steering of the vehicle based on the different embedded systems of the vehicle. Depending on the Autonomous Assist Module chosen, such as traffic jam, autonomous parking, lane positioning correction, or traffic assist, the values of the torque, steering angle, lateral velocity, and yaw angle will be generated intuitively creating an autonomous driving experience. The software shall remain usable until there is any failure in the different embedded systems, in which it will cancel any use of the autonomous functionality of the vehicles steering. The class diagram described here can be seen in Figure 4.

Simple UML Model Key:

This is a simple key to understand the relationships in the different UML diagrams represented in this SRS [5].

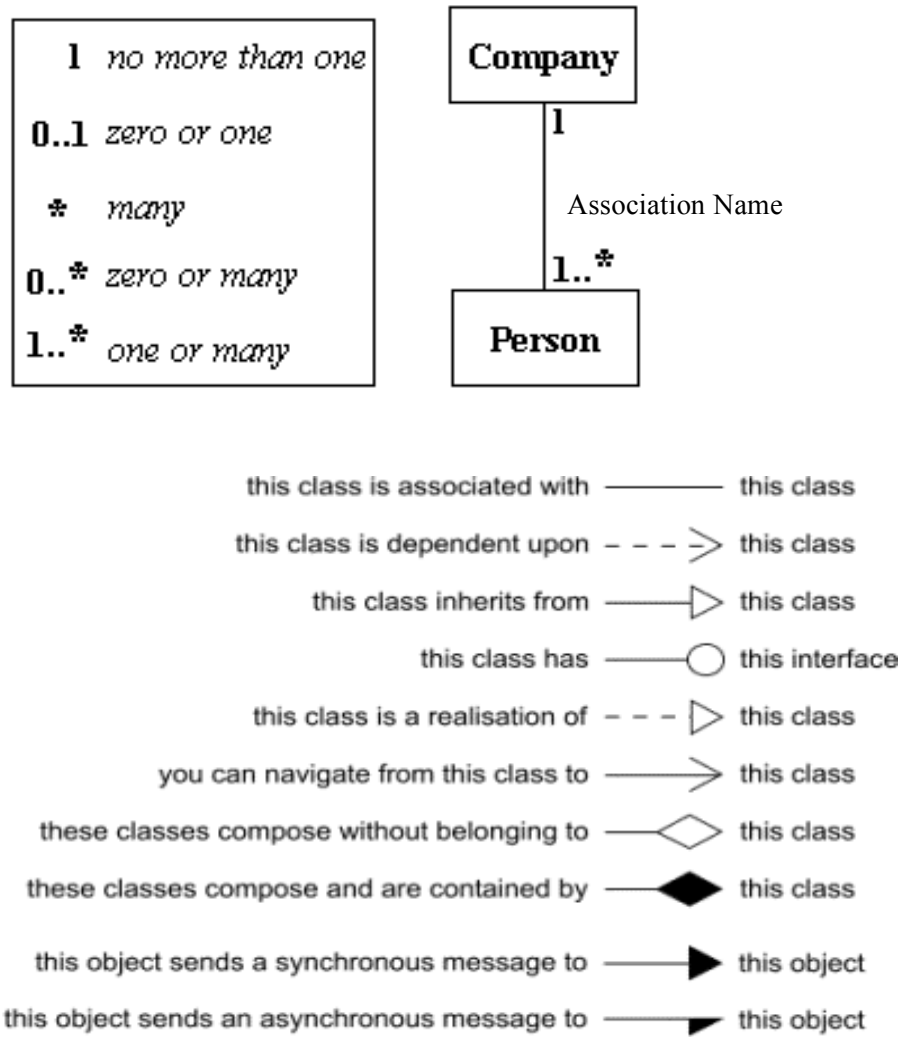

Figure 3: UML Key

Class Diagram Data Dictionary:

Element Name		**Description**
Autonomous Control		This class manages all autonomous based controls used by the system. . In this class, the proper mode for the autonomous steering is selected. Also it keeps track of which autonomous modes are not selected in case it must override to mode for another dependent on priority.
Attributes		
IsLanePositionCorrection	IsLanePositionCorrection**:** Bool	This is set to true or false dependent on which Autonomous Control mode is requested. Set to true if this is chosen.
IsTrafficAssist	IsTrafficAssist**:** Bool	This is set to true or false dependent on which Autonomous Control mode is requested. Set to true if this is chosen.
IsEmergencyObjectAvoidance	IsEmergencyObjectAvoidance: Bool	This is set to true or false dependent on which Autonomous Control mode is requested. Set to true if this is chosen.
IsParkAssist	IsParkAssist: Bool	This is set to true or false dependent on which Autonomous Control mode is requested. Set to true if this is chosen.

AutonomousMode	AutonomousMode: ENUM	This is an enumeration sent in from the driver to tell what mode was specified before entering into the autonomous driving system.
Operations		
UpdateAutonomousSteering	UpdateAutonomousSteering (ENUM): void	This functionality is called to make adjustments in the steering for any reason.
RequestControlMode	RequestControlMode (ENUM): void	This operation is called by the driver to let the Autonomous Control know that autonomous driving was selected and the specific mode is this being passed in as an enumeration.
SetAutonomousMode	SetAutonomousMode (ENUM): void	This operation takes in the enumeration of the autonomous mode needed and sets the Boolean attributes and the AutonomousMode accordingly.
OverrideAutonomousMode	OverrideAutonomousMode (ENUM): void	This operation is used when one mode is set and must be changed to a different autonomous mode.
Calibrate	Calibrate (): void	This operation is used to calibrate the system with a software update.
Relationships	This class has a relationship with the Driver, Worker, Override, and Sensor Controller classes. It gathers the Autonomous Control	

	mode from the driver, updates software from the worker, can override the Autonomous Control with override, and lastly can checks the various sensors on the vehicle with the sensor controller.	

Element Name		**Description**
Alert		This class controls the execution of the visual and audible alert for when the Autonomous Control fails.
Operations		
VisualAlert	VisualAlert (): void	Used to execute the visual alert when the Autonomous Control failed.
AudibleAlert	AudibleAlert (): void	Used to execute the audible alert when the Autonomous Control failed.
AlertDriver	AlertDriver (): void	This is a method called and is used to call the visual and audible alerts.
Relationships	This class has a relationship with Failure where that class calls functions in the Alert class if the system is failing and needs to be disabled.	

Element Name		**Description**
Driver		This is a class to represent the drivers' decision on which mode of driving they will expect from the vehicle.
Operations		
ChooseNormalControl	ChooseNormalControl (): void	This is an operation that sets the control chosen by the driver to be Normal Control.

ChooseAutonomousControl	ChooseAutonomousControl (): void	This is an operation that sets the control chosen by the driver to be Autonomous Controls.
DisableAutonomous	DisableAutonomous (): void	This is used to disable the autonomous steering in the system.
Relationships	This class has relationships with Autonomous Control, Normal Control, Failure, and Steering Wheel. The Driver class can apply torque to the Steering Wheel class, request a certain autonomous mode from the Autonomous Control, set the mode to Normal Control, or disable the automated steering.	

Element Name		**Description**
EPAS Controller		This class is used to hold and alter the different steering variables that are picked up by the various sensors. Then with those values it alters the steering.
Attributes		
YawVelocity	YawVelocity: float	Holds the value for the yaw velocity.
SteeringAngle	SteeringAngle: float	Holds the value for the steering angle.
LateralAcceleration	LateralAcceleration: float	Holds the value for the speed/ lateral acceleration of the vehicle.
Torque	Torque: float	Holds the value of the torque rendered on by the autonomous mode or driver on the wheel.
Operations		

ManipulateSteeringOutput	ManipulateSteeringOutput (): void	This operation is used to manipulate the steering properly depended on values.
EncryptSteeringInfo	EncryptSteeringInfo (): void	This operation is called to take the steering info that will be sent around and encrypt the message so no outside party can see the messages.
DeterminePriority	DeterminePriority (ENUM): vector<ENUM>	This operation determines the proper order of commands to execute dependent on their priority for the autonomous steering.
Relationships	This class has a relationship with Sensor Controller, Torque Sensor, and Motor Power Amplifier. EPAS Controller is using all these classes to both gather proper steering information and executing on that information through the Motor Power Amplifier class.	

Element Name		**Description**
Electric Motor		This class represents the electric motor that assists the driver in applying torque to turn the wheel of the vehicle.
Operations		
ElectricMotorAssist	ElectricMotorAssist (float): void	This operation is called to generate power to the create torque to help turn the wheel and vehicle in the proper direction.
Relationships	This class has a relationship with Motor Power Amplifier where the Motor Power Amplifier class sends a value to this class for the motor to	

	output the actual torque assist.	

Element Name		**Description**
Failure		This class is used when a failure in the system is detected.
Operations		
FailureDetected	FailureDetected (): void	This operation is called when a failure is detected and sends the system to Normal Control rather than allowing autonomous mode to continue.
DisableAutonomousSteering	DisableAutonomousSteering (): void	This operation is used to completely disable the use of autonomous steering and set the system to only operate under Normal Control.
CallAlert	CallAlert (): void	This operation is used to call the functionality in the Alert class to inform the driver that the Autonomous Control has failed and is shutting down.
Relationships	This class has a relationship with Sensor Controller, Normal Control, Driver, adn Alert. From the Sensor Controller the Failure class gets told if there is a failure in one of the sensors. Then the failure class must call methods on both the Alert and Normal Control class.	

Element Name		**Description**
Motor Power Amplifier		This class is used adjust the Pulse Width Modulations (PWM) which is used to send signal to the assist motor to apply

		the proper assist for the torque.
Attributes		
AmplifyPower	AmplifyPower: float	Holds the value of the amplified power.
Operations		
AmplifySteeringPower	AmplifySteeringPower (float): float	This operation is called to increase to power according to the specification passed into it from the EPAS Controller message through PWM.
Relationships	This class has a relationship with EPAS Controller and Electric Motor. The EPAS Controller sends a Pulse Width Modulation that is amplified by this class that in turn creates the assisted power. The assisted power is then generated by the Electric Motor.	

Element Name		**Description**
Normal Control		This class is enacted whenever Autonomous Control is not set. This is also chosen whenever the driver applies any torque to the wheel. Normal Control takes inputs from the driver rather than the Autonomous Control System to determine steering output.
Operations		
SetNormalControl	SetNormalControl (): void	This operation is called to set the mode to Normal Control mode, where the input for the torque, and steering angle is determined by the driver

		rather than generated autonomously.
Relationships	This class has a relationship with the Driver, Override, and Failure class. The Driver, Failure, and Override class can set the control mode to normal by calling this class's method.	

Element Name		**Description**
Override		This class is used to override the autonomous mode and force the vehicle into Normal Control.
Attributes		
OverrideAutonomous	OverrideAutonomous: Bool	This is a Boolean set to keep track it the system is set to override the autonomous mode.
Operations		
ChangeSteeringControl	ChangeSteeringControl (): void	This is the operation that sets the actual control mode to normal from autonomous.
Relationships	This class has a relationship with Autonomous Control and Normal Control. When an autonomous mode is active, at any time there is the option to override the Autonomous Control and set the control to Normal Control.	

Element Name		**Description**
Sensor Controller		This is the controller that determines which autonomous mode the system will enter into and it determines which sensors to check depended on the autonomous mode entered.

Attributes		
AutonomousMode	AutonomousMode: ENUM	This attribute represents the mode of the system that it is currently in; whether it is park assist, traffic assist, etc.
Operations		
CheckDifAutonomousSensors	CheckDifAutonomousSensors (ENUM): void	This operation is called to check the different sensors according to which autonomous mode the system is in.
OverTorqued	OverTorqued (): void	This operation is used to determine if the torque applied to the wheel was past any variants.
CheckSensorFailure	CheckSensorFailure (): Bool	This operation is called to see if any of the sensors has failed and returned failure values.
Relationships	This class has a relationship with the Autonomous Control, Steering Wheel, Torque Sensor, and EPAS Controller class. This class controls the different sensors available and send information to the EPAS Controller class.	

Element Name		**Description**
Steering Wheel		This class is used to determine the torque applied to the steering wheel. And check to see if the wheel was torqued passed the invariant set.
Operations		
TorqueApplied	TorqueApplied (): void	This operation conveys torque being applied to the steering wheel.

Relationships	This class has a relationship with the Driver and Sensor Controller class. The driver can apply torque to the steering wheel. Then the Steering Wheel class talks to the Sensor Controller class that in turn gets the information from the torque sensors to determine how much torque was applied to the steering wheel.	

Element Name		**Description**
Torque Sensor		This class is actually the sensors used to determine how much torque was applied.
Attributes		
Torque	Torque: float	This is used to store the value of the torque that was applied.
Operations		
TorqueSensed	TorqueSensed (): void	This operation is used to get the value from the embedded system sensors for the torque applied.
Relationships	This class has a relationship with the Sensor Controller that calls methods upon this class, and a relationship with the EPAS Controller. The EPAS Controller gets the value of the torque.	

Element Name		**Description**
Worker		This class is used to represent a software update to the Autonomous Control through calibration.

Attributes		
PersonalID	PersonalID: int	This is used to store the workers personal ID to identify later for various reasons.
Operations		
CreateCalibration	CreateCalibration (): void	This operation is used to represent the composing of the software update.
RecordID	RecordID (): void	This set the PersonalID attribute with the workers unique personal ID.
Relationships	This class has a relationship with the Autonomous Control class that it sends new calibrations too.	

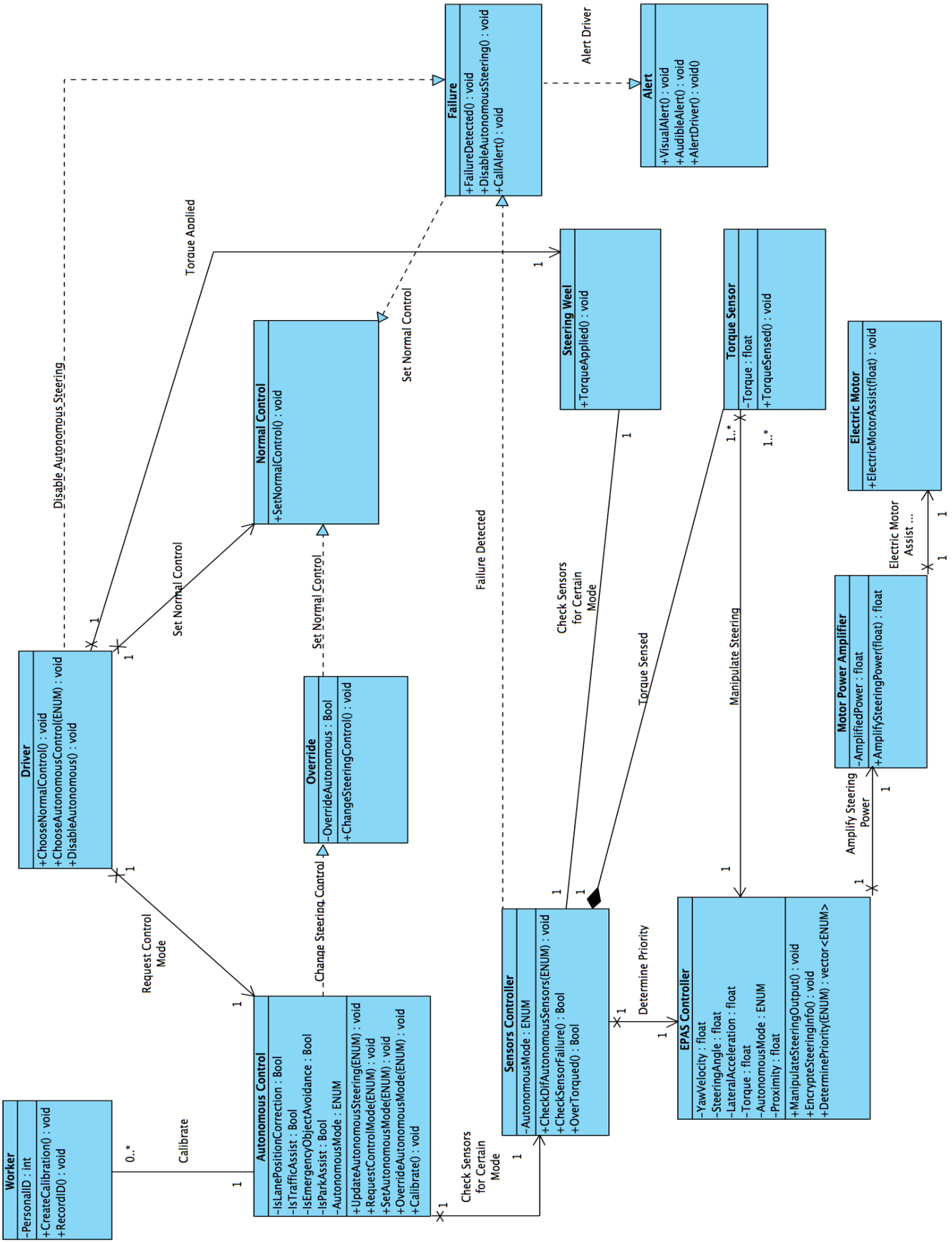

Figure 4: Class Diagram

Scenarios of Autonomous Steering

Use Case (Detect Torque): When the driver turns the wheel, the two torque sensors detect the torque. This will be done every 500 milliseconds in Normal Control. As seen in Figure 5.

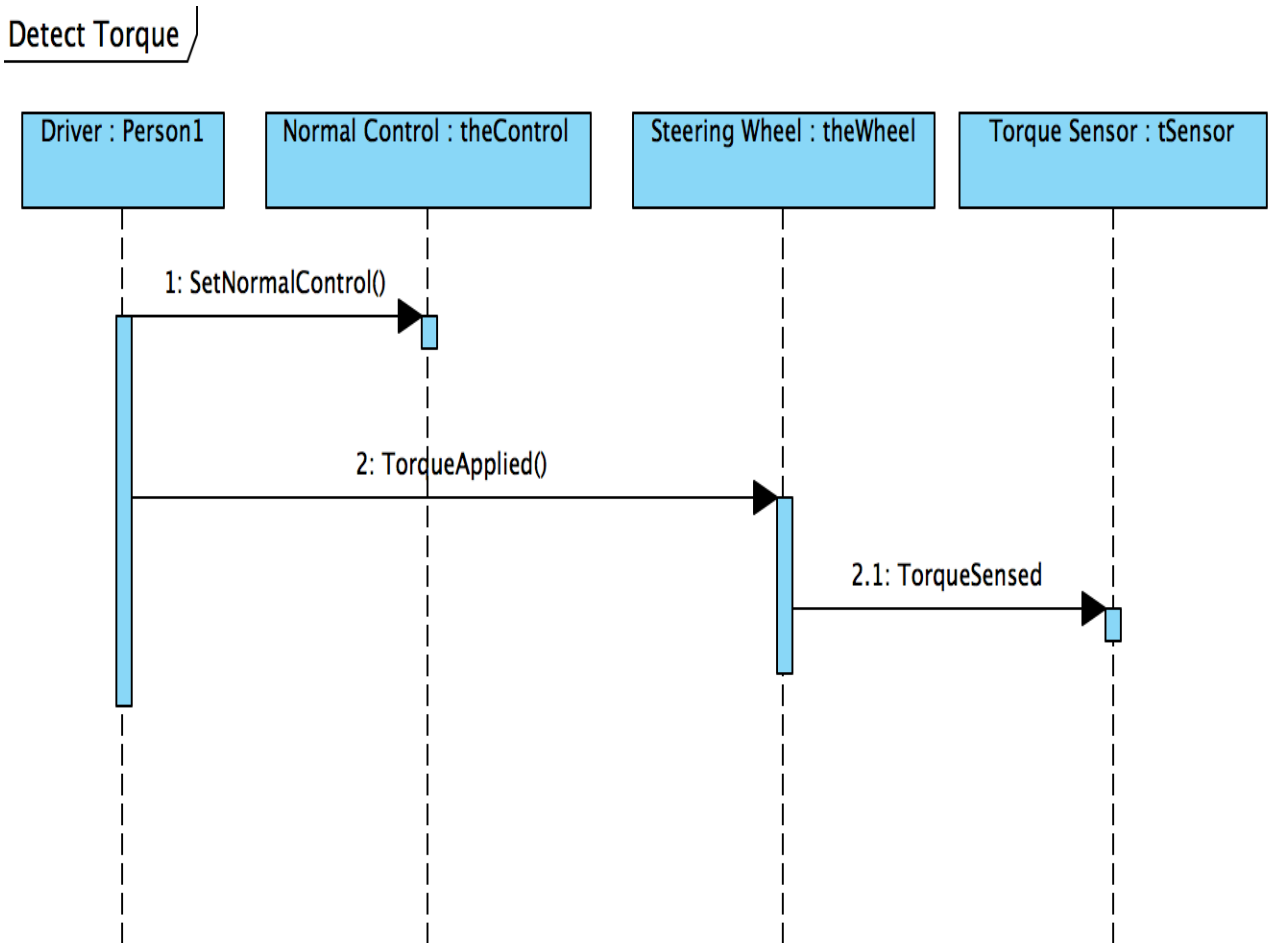

Figure 5: Sequence Diagram for Detect Torque

Use Case (Receive Input): Receive the input from the Driver Assist Modules, and from the torque sensor used by Normal Control. Performs validation checks on inputs to detect defective sensors. Inputs will be sent using CAN, and packets will be encrypted and password protected to prevent outside tampering. Only inputs from valid ECUs will be accepted. Unused ports will be disabled. As seen in Figure 6.

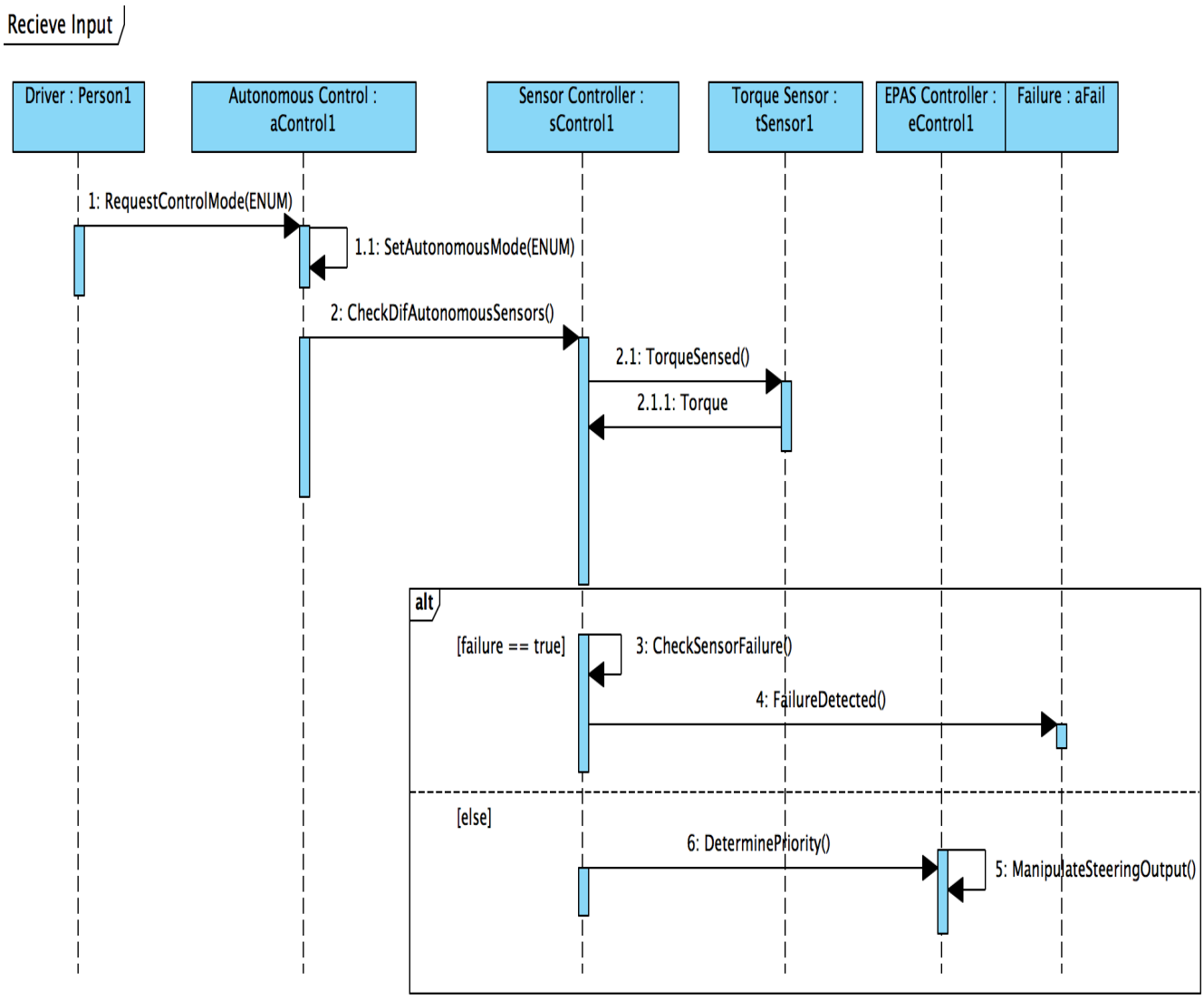

Figure 6: Sequence Diagram for Receive Input

Use Case (Change Steering Angle): Given the inputs from either the driver assist module, or inputs from the torque applied to the steering wheel by the driver; determine how much to adjust the steering angle of the vehicle. This is done based on a function of the speed of the vehicle, how much torque should be applied, and the current angle of the vehicle. As seen in Figure 7.

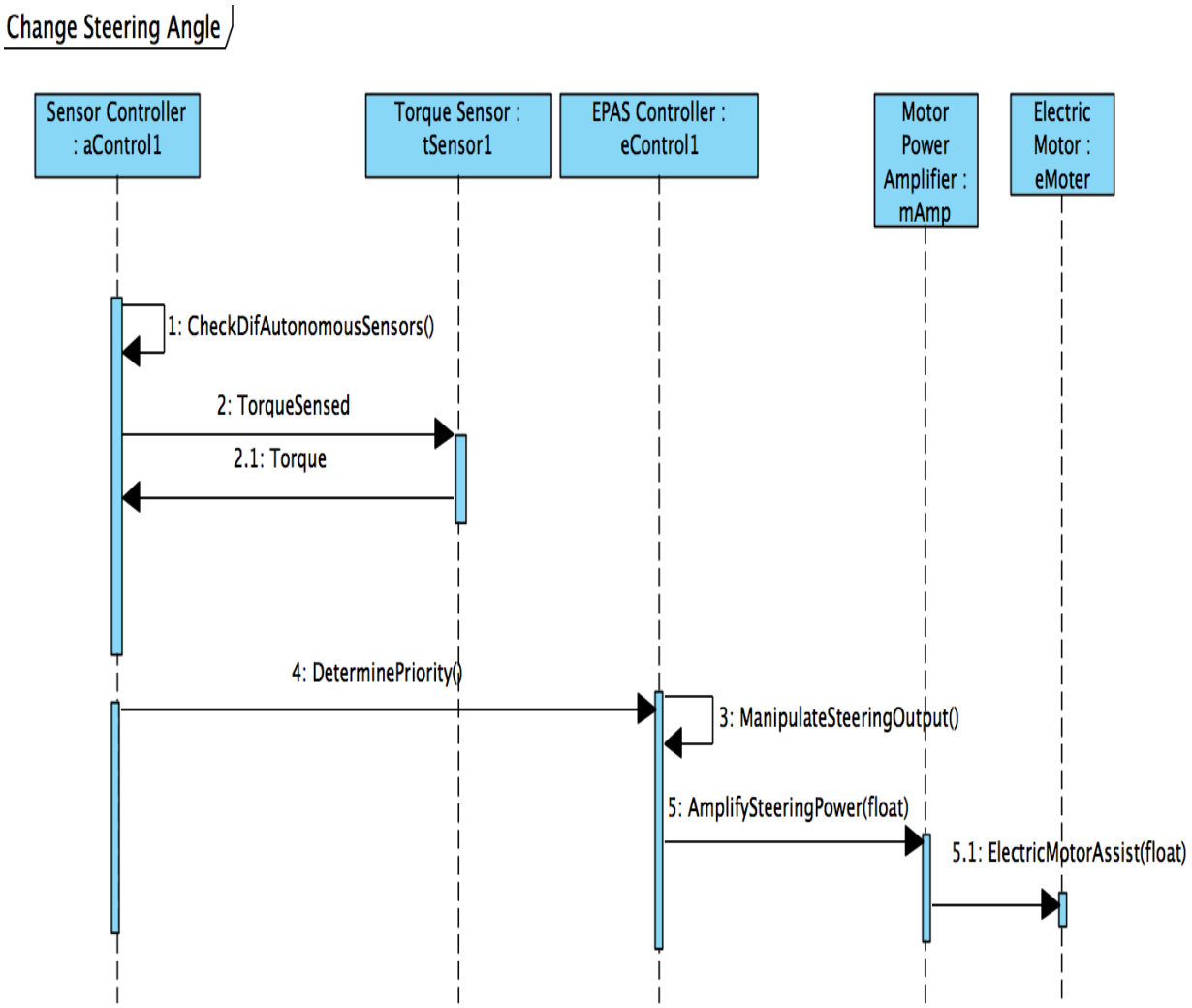

Figure 7: Sequence Diagram for Change Steering Angle

Use Case (Manual Override): The driver can press a button in reaching distance that will disable all automatic control of the vehicle, shown in Figure 8

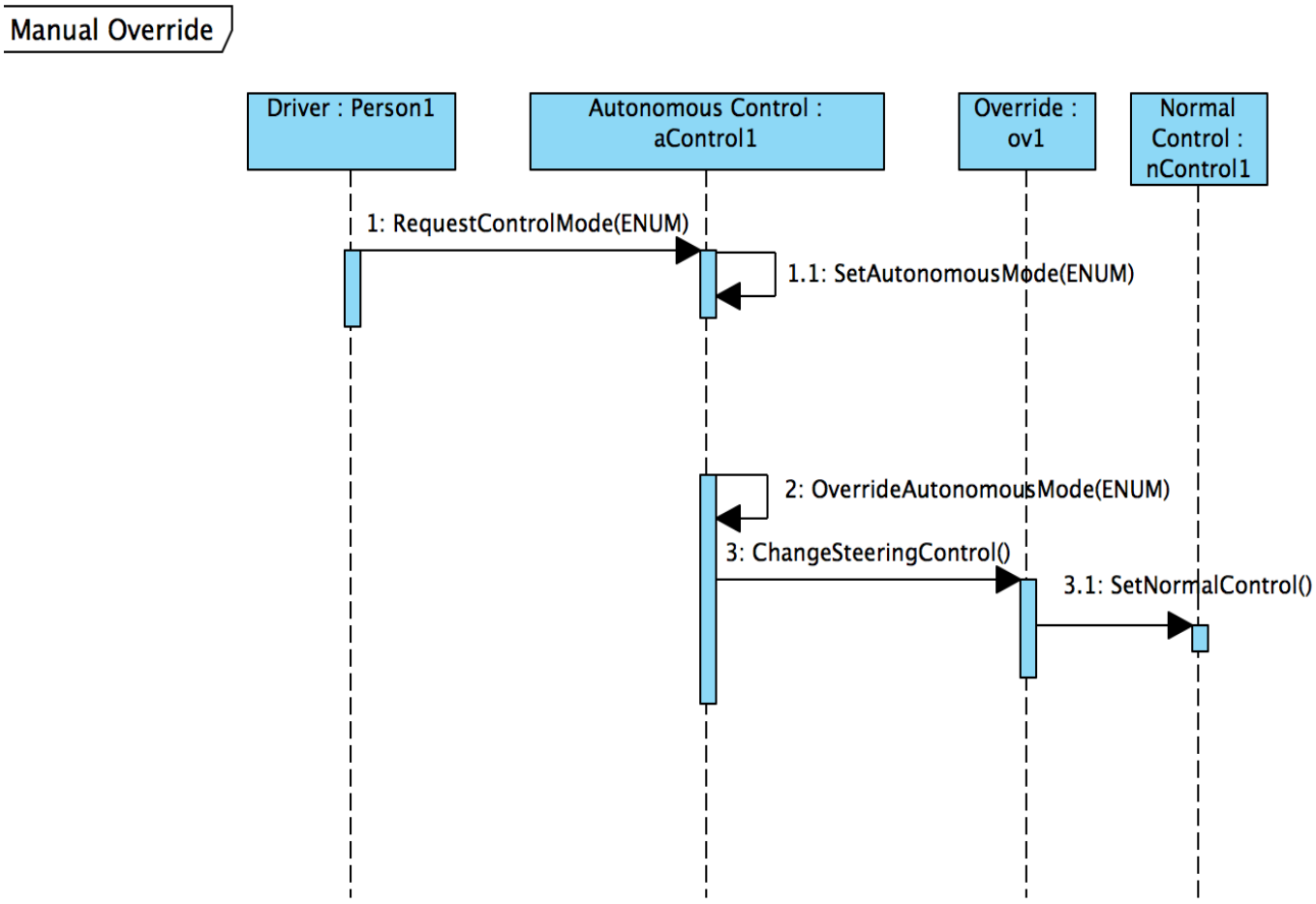

Figure 8: Sequence Diagram for Manual Override

Use Case (Disable Automatic Steering): This will disable the automated steering, causing the steering to be handled only by the driver. It will switch to Normal Control within 500 milliseconds. As seen in Figure 9.

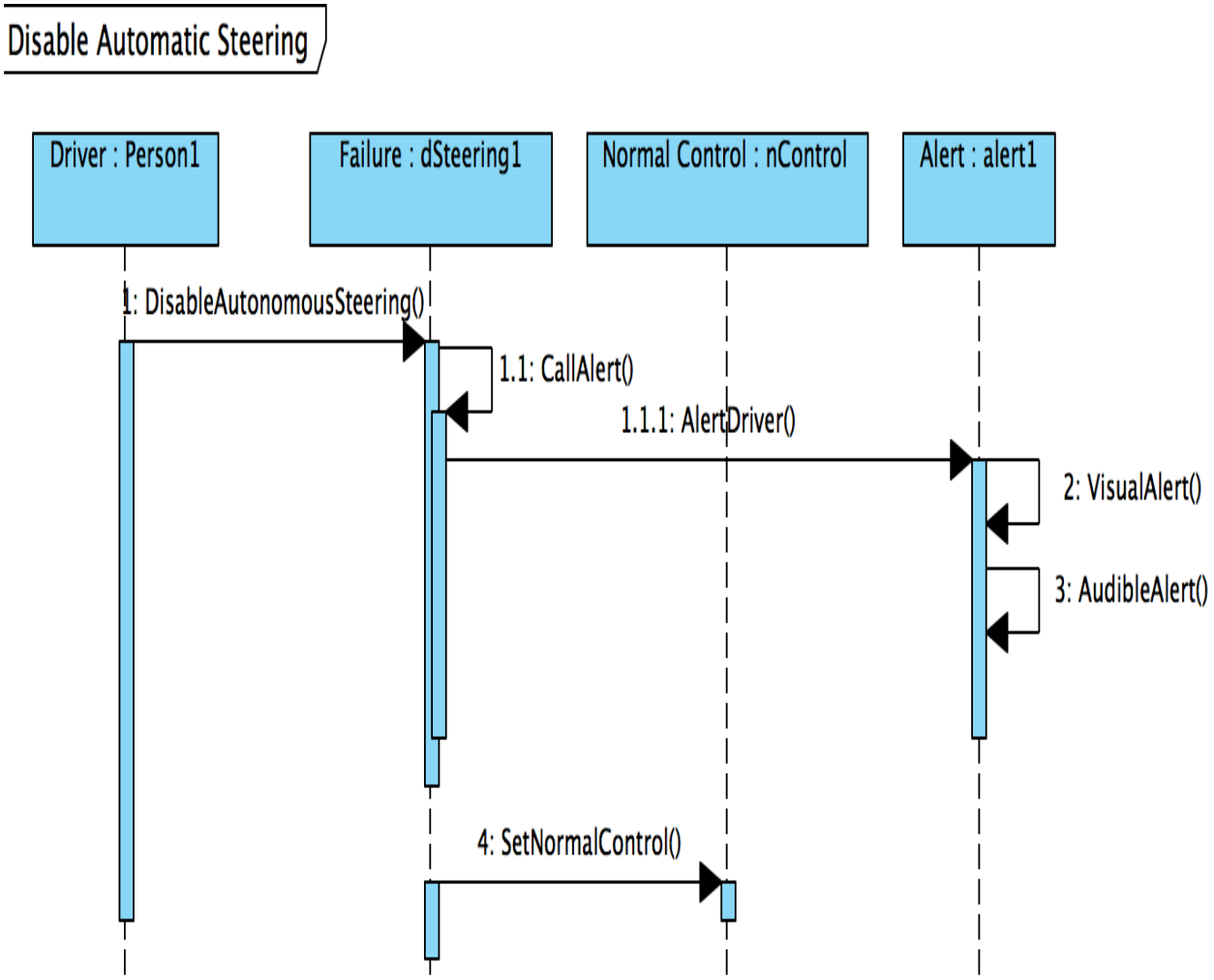

Figure 9: Sequence Diagram for Disable Automatic Steering

Use Case (Detect System Failure): Detect a system failure in a vital component, such as the torque sensor attached to the steering column, or receiving a value that is not within the intended range. Shown below in Figure 10.

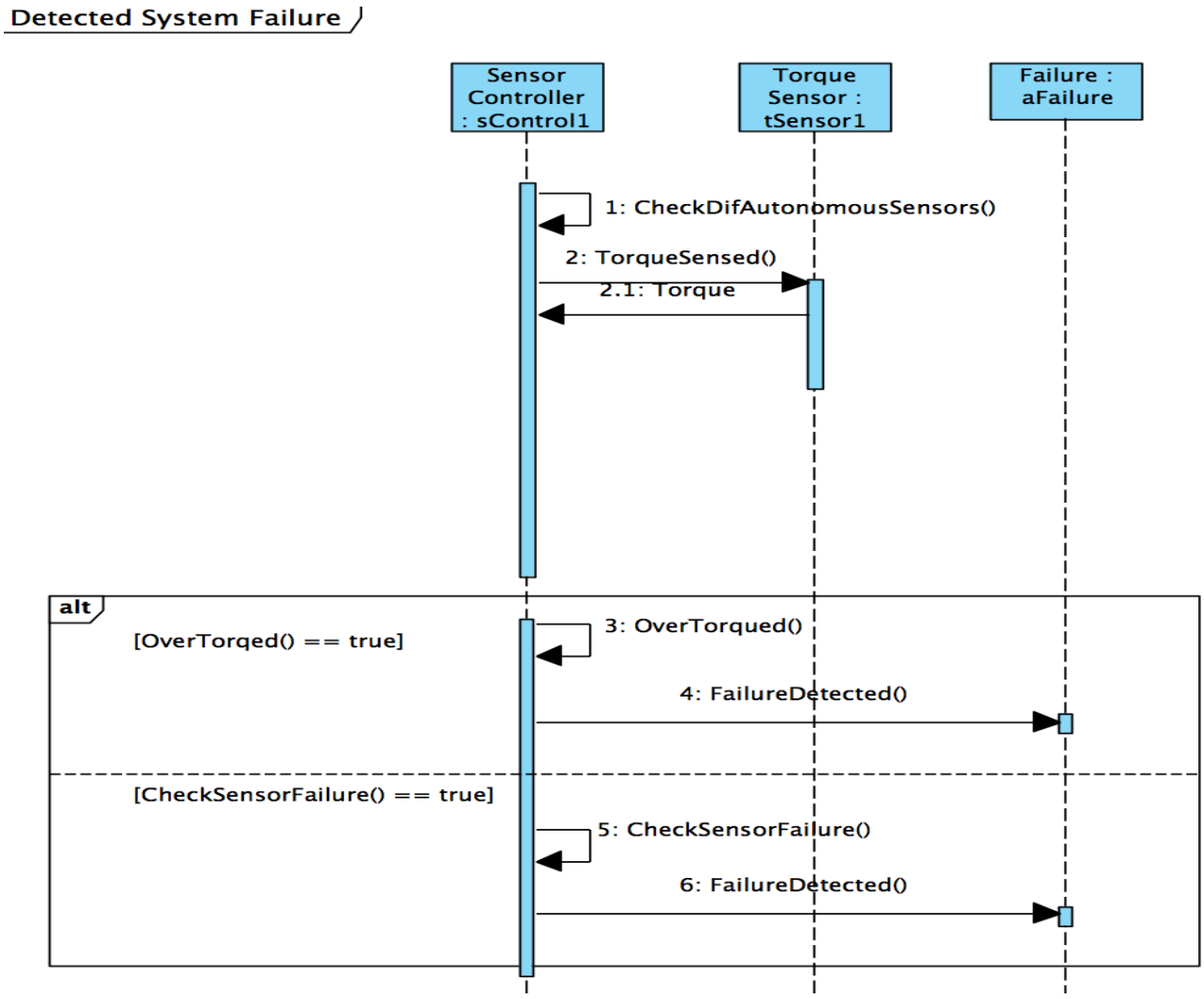

Figure 10: Sequence Diagram for Detected System Failure

Use Case (Record ID): When an update is applied to the system or the system is calibrated, the ID of the person who made the changes will be recorded in the system. Shown in Figure 11.

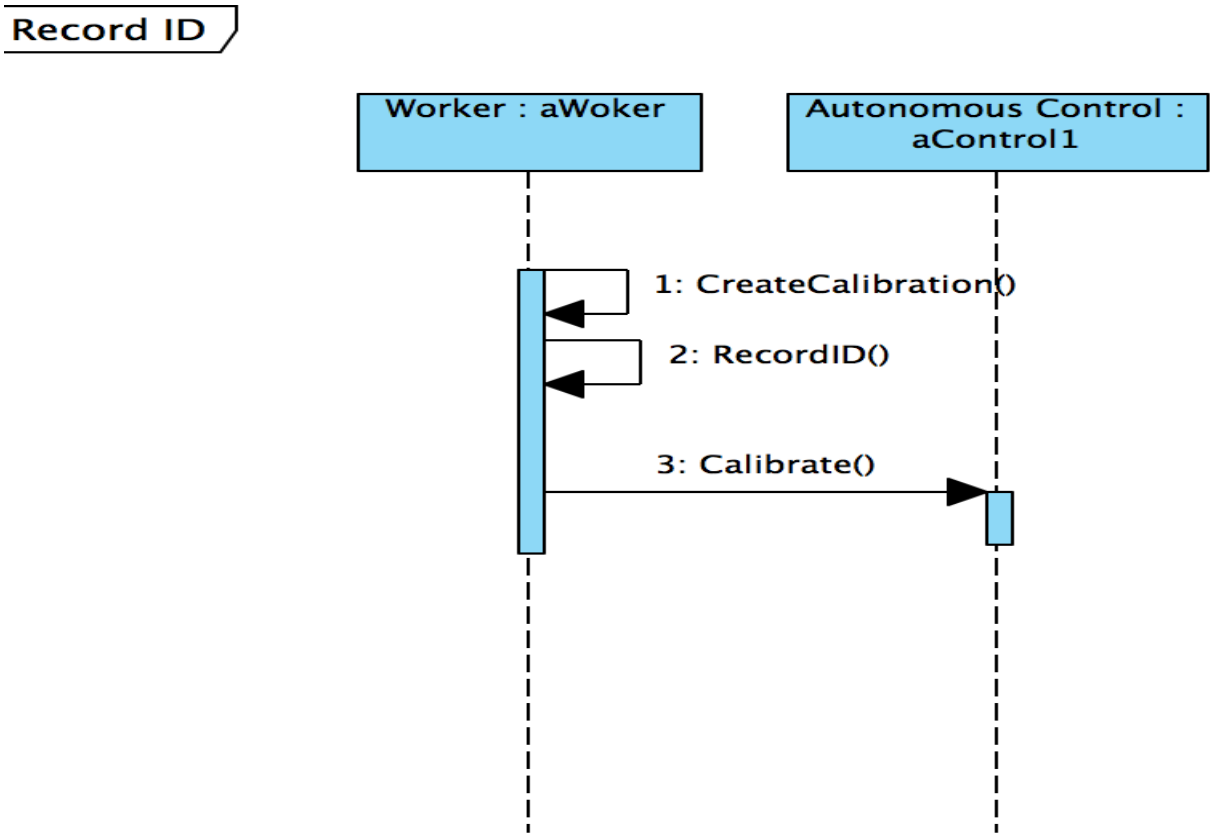

Figure 11: Sequence Diagram for Record ID

Use Case (Failure Alert): When a failure is detected, the system must shut down, an audio alert sound and visual light that will alert the driver to the failure. Shown in Figure 12.

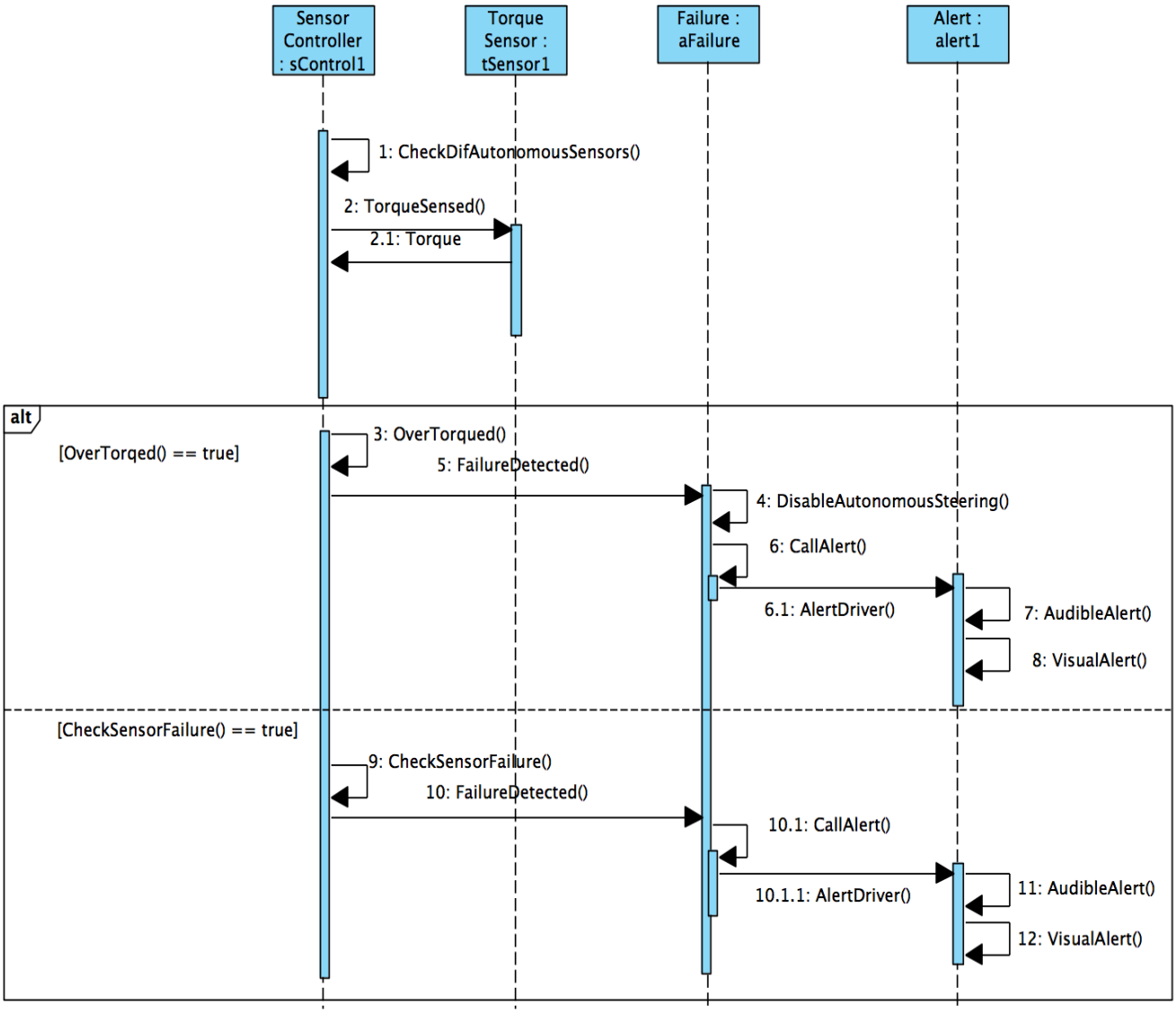

Figure 12: Sequence Diagram for Failure Alert

State Diagram:

The system starts in Autonomous Control, receiving inputs from the various modules and passing their inputs along. If the system switches to Normal Control at any point, the Autonomous Control will cease the current operation and wait until Autonomous Control resumes. If the system remains in AutonomousMode, once these inputs are received and parsed, the system moves on to pass the steering outputs to the EPAS controller. Then based on past values, the system will either return to Autonomous Mode or ManualMode. This can be seen below in Figure 13.

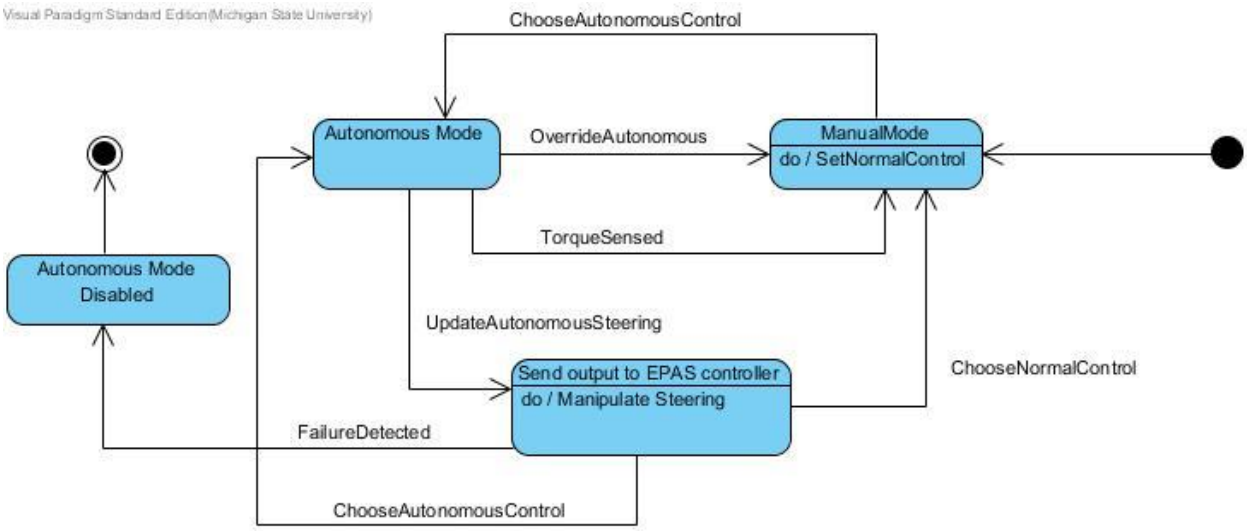

Figure 13: Autonomous Control State Diagram

The Sensor Controller starts by checking every 500 microseconds for a change in the torque sensor. Once a change in torque is detected, the system ramps down into Normal Control and the two torque sensors are compared to check for a failure. If no failure is detected, the output is sent to the EPAS Controller to be used to change the steering angle. If a failure is detected in one of the torque sensors, the entire EPAS is shut down. This can be seen in Figure 14.

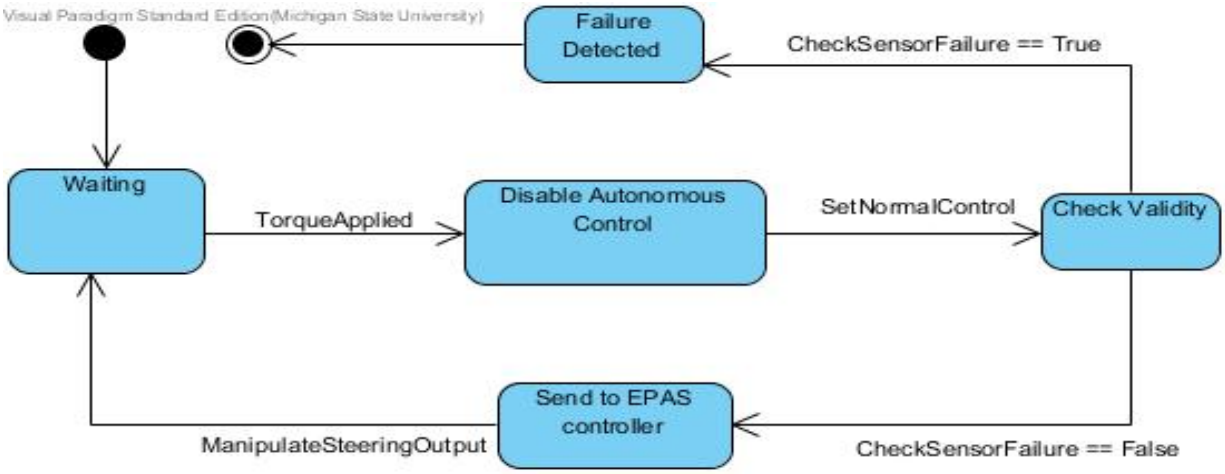

Figure 14: Sensor Controller State Diagram

5 Prototype

High-Level Autonomous Prototype Description:

This is a user interface prototype for an EPAS system. As seen in the UI prototype in Figure 15. There are multiple variables that may be manipulated to simulate an expected output. The main objective of this prototype is to calculate the expected assisted torque and steering angle that needs to be generated by the systems at any given point for any mode.

5.1 How to Run Prototype

1. The prototype runs in many web browsers running on most major platforms.
    1.1. Windows: Internet Explorer, Google Chrome, Mozilla Firefox
    1.2. Mac: Safari, Google Chrome, Mozilla Firefox
2. It can be accessed at this URL: https://www.cse.msu.edu/~cse435/Projects/F2014/Groups/STEERING1/web/prototype.html
3. JavaScript must be enabled.
4. The Control Mode section lists all modes the EPAS supports. Set the desired mode by clicking on the name in the Control Mode section.
5. Adjust the sliders to choose desired input and torque as shown in Figure 15.
6. Click the green calculate button
7. System output will display in the Output section.

5.2 Sample Scenario

Detect System Failure: A system failure will occur if a torque sensor malfunctions. The Torque Sensors sections allows user to manipulate how the torque sensors are functioning. Click the torque sensor will toggle between a properly functioning and malfunctioning. If the box is unchecked that means the torque sensor has malfunctioned. Unchecking both torque sensor boxes on the system interface (Figure 15) simulates a scenario in which the system should shut down.

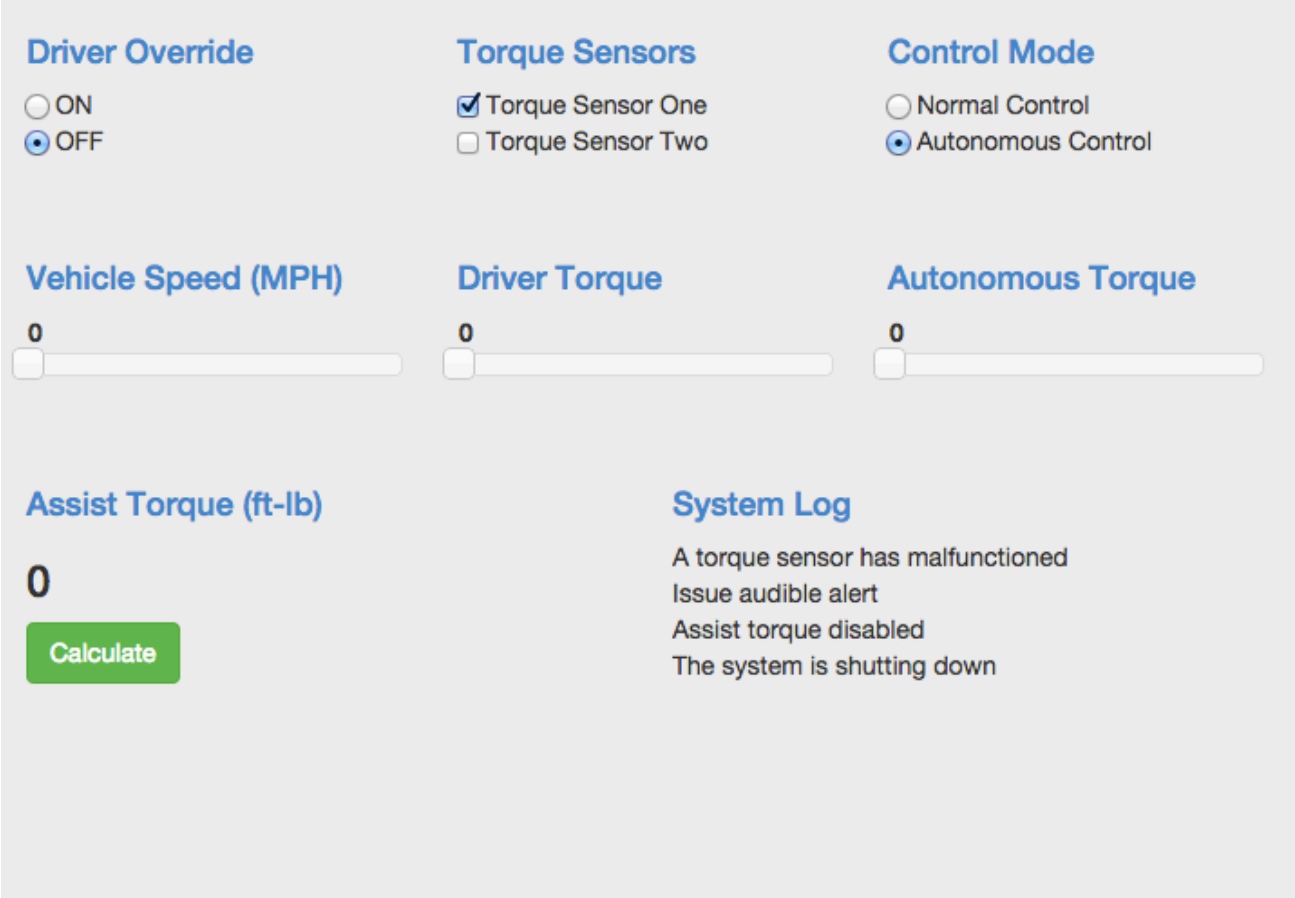

Figure 15: System Failure Scenario

6 References

[1] D. Thakore and S. Biswas, “Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks,” Proceedings of IEEE Military Communication, Atlantic City, October 2005.

[2] Don Sherman, "Are We Losing Touch? A Comprehensive Comparison Test of Electric and Hydraulic Steering Assist", January 2012, available: http://www.caranddriver.com/features/electric-vs-hydraulic-steering-a-comprehensive-comparison-test-feature

[3] Lawrence Ulrich, "Top Tech Cars 2013: Infiniti Q50", 29, March, 2013, available: http://spectrum.ieee.org/transportation/advanced-cars/infiniti-q50

[4] Matthew Beecham, "RESEARCH ANALYSIS: A review of electric power steering systems ", available: http://www.just-auto.com/analysis/a-review-of-electric-power-steering-systems_id92107.aspx

[5] Nigel, “Architects and Software Architects”, available: http://www.ivencia.com/index.html?/softwarearchitect/chapter1/chapter1.htm

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Betty H.C. Cheng** at Michigan State University (chengb at cse.msu.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.