Team: Overdrive

Architecture Design Specification

Project: Auto Performance Analyzer

Team Members:

Gary Johns
Raul Hurtado
Brendon Harris
Zang Pham
5.1 GUI Event Handler Subsystem ................................................................. 14
5.2 Processing Event Handler Subsystem ...................................................... 15
5.3 DBMS Event Handler Subsystem ............................................................ 16
6. Processing Layer ....................................................................................... 18
   6.1 Processing Controller Subsystem .......................................................... 18
   6.2 Bluetooth Processing Subsystem ........................................................... 19
7. Bluetooth Layer ......................................................................................... 21
   7.1 Data Acquisition Subsystem ................................................................. 21
   7.2 Sync/Pair Subsystem ........................................................................... 22
8. Database Management System (DBMS) Layer .......................................... 24
   8.1 Database Controller (DB Controller) Subsystem .................................. 24
9. Inter-Subsystem Data Flow ......................................................................... 26
   9.1 Overview .............................................................................................. 26
   9.2 Data Flow Definitions ......................................................................... 26
   9.3 Producer-Consumer Relationship ......................................................... 27
10 Operating System Dependencies Section ................................................. 28
    10.1 Overview ............................................................................................ 28
    10.2 GUI Layer .......................................................................................... 28
    10.3 Processing Layer ................................................................................. 28
    10.4 DBMS Layer ....................................................................................... 28
    10.5 Bluetooth Layer .................................................................................. 28
10. Requirement Mapping .............................................................................. 29
11. Testing Considerations Section ................................................................. 30
    11.1 Overview ............................................................................................ 30
    11.2 GUI Layer .......................................................................................... 30
    11.3 Processing Layer ................................................................................. 30
11.4 Event Manager Layer ............................................................................................................................................30
11.5 DBMS Layer .........................................................................................................................................................31
11.7 Integrity Testing Approach .......................................................................................................................................31
## Document Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision Date</th>
<th>Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>8/08/2013</td>
<td>Rough Draft</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>8/10/2013</td>
<td>Revisions</td>
<td>Fixed grammatical errors</td>
</tr>
<tr>
<td>1.0</td>
<td>8/11/2013</td>
<td>Gate Review Version</td>
<td>Applied peer suggestions</td>
</tr>
<tr>
<td>1.1</td>
<td>8/13/2013</td>
<td>Connection Revisions</td>
<td>Fixed incorrect data flows</td>
</tr>
<tr>
<td>2.0</td>
<td>9/6/2013</td>
<td>Baseline</td>
<td>Final corrections made from Gate Review</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 - 1 Block diagram of system .......................................................................................... 2
Figure 2 - 1 Initial architectural design......................................................................................... 5
Figure 2 - 2 Refinement of initial architectural design................................................................. 6
Figure 3 - 1 Architectural diagram of system ................................................................................ 7
Figure 4 - 1 User Input subsystem inter-layer interfaces ............................................................. 10
Figure 4 - 2 Presentation subsystem inter-layer interfaces ......................................................... 11
Figure 4 - 3 I/O Controller subsystem inter-layer interfaces ..................................................... 13
Figure 5 - 1 GUI Event Handler subsystem inter-layer interfaces ........................................... 14
Figure 5 - 2 Processing Event Handler subsystem inter-layer interfaces .................................. 15
Figure 5 - 3 DBMS Event Handler subsystem inter-layer interfaces ......................................... 17
Figure 6 - 1 Processing Controller subsystem inter-layer interfaces ....................................... 18
Figure 6 - 2 Bluetooth Processing subsystem inter-layer interfaces ......................................... 20
Figure 7 - 1 Data Acquisition subsystem inter-layer interfaces ............................................... 21
Figure 7 - 2 Sync/Pair subsystem inter-layer interfaces ............................................................ 22
Figure 8 - 1 Database Controller (DB Controller) subsystem inter-layer interfaces ............... 24
Figure 9 – 1 Data Flow Definitions ............................................................................................ 26
Figure 9 - 2 Producer consumer relationship .............................................................................. 27
List of Tables

Table 1 - 1 Requirements..................................................................................................................3
Table 4 - 1 User Input subsystem inter-layer interfaces ....................................................................11
Table 4 - 2 Presentation subsystem inter-layer interfaces .................................................................12
Table 4 - 3 I/O Controller subsystem inter-layer interfaces ...............................................................13
Table 5 - 1 GUI Event Handler subsystem inter-layer interfaces ......................................................15
Table 5 - 2 Processing Event Handler subsystem inter-layer interfaces ..........................................16
Table 5 - 3 DBMS Event Handler subsystem inter-layer interfaces ..................................................17
Table 6 - 1 Processing Controller subsystem inter-layer interfaces ................................................19
Table 6 - 2 Bluetooth Processing subsystem inter-layer interfaces ..................................................20
Table 7 - 1 Data Acquisition subsystem inter-layer interfaces .........................................................22
Table 7 - 2 Sync/Pair subsystem inter-layer interfaces ....................................................................23
Table 8 - 1 DB Controller subsystem inter-layer interfaces ..............................................................25
Table 10 - 1 Requirements Mapping ..............................................................................................29
1. Introduction

1.1 Product Overview

The Auto Performance Analyzer shall consist of one mobile application, running on a smartphone or tablet, for a pre-existing COTS (Commercial Off The Shelf) device that will plug into any car’s (model 1996 or newer) On Board Diagnostics – generation II connector (OBD-II). The pre-existing module shall connect and communicate with the mobile application, only returning data that was requested, via Bluetooth. The COTS device is commercially designed to be small and can handle temperature extremes of -20°C to +75°C.

The mobile application shall query the on-board computer through the COTS device, which will obtain the data from the car and will provide such data to the mobile application. This application will give the user the option to choose the parameters of interest from a list and will display the data in real time.

The mobile application shall provide the user with the option to retrieve and clear trouble codes that are produced by the car when the check-engine light is on. The mobile application will retrieve the trouble codes data from the car and pull the information from a trouble code database stored in the device when the application is installed. The trouble code retrieval, interpretation and clear feature will be displayed in a separate screen independent of the parameter’s display screen.

1.2 Purpose and Use

The purpose of this document, Architectural Design Specification or ADS, is to provide a high level view of how the Auto Performance Analyzer will be implemented. The ADS shows the guiding principles that team Overdrive has established that must be followed throughout the design process. Included is a graphical display of the systems architecture along with a layer by layer detailed description. The ADS will define the five different architectural layers and the different subsystems contained inside each of them. The ADS will also present the responsibilities of each layer and the interactions between their subsystems. The document provides the Auto Performance Analyzers operating system dependencies with the Android operating system along with the many testing considerations that must be implemented. The ADS will lead to detailed design and finally software implementation.
1.3 Project Scope

The scope of the Auto Performance Analyzer project is to create a diagnostics module that interfaces with the OBD-II connector system in automobiles to retrieve information from that automobile. The Auto Performance Analyzer will provide a mobile user interface with enhanced tools for data logging, vehicle diagnosis, and real time statistics. The interface will be available for Android devices, such as smart phones and tablets. The application will be able to display the cars’ parameters data as digital numeric readouts and graphs to the user.

In order to populate the user interface with data, the module retrieves this data from the existing OBD-II connector, parses and reformats it. The module will then transmit it over the Bluetooth connection, to the mobile device, in real time.

![Block diagram of system](image)

**Figure 1 - 1** Block diagram of system

1.4 Requirements

The main requirement Overdrive has chosen to focus on in the ADS document are listed in Table 1-1. These requirements were used to develop the system architecture, because they are core requirements that must be implemented. Also, these requirements have a rational dataflow the team used to verify that the architecture design would work. A full list of requirements can be found in the System Requirements Specification.
<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Real-Time Data Acquisition</td>
<td>The STN1170 Bluetooth OBD-II Adapter will read the data from the vehicle's OBD-II connector, which it will then send to a mobile device through Bluetooth in quick successions.</td>
</tr>
<tr>
<td>2</td>
<td>Performance Mode – Select Readings</td>
<td>The mobile application will display performance readings, as well as sensor readings such as oil pressure, temperature (certain automobiles may not have all available readings). The mobile application will have a list of check-boxes that contains these readings that the user can select to have monitored.</td>
</tr>
<tr>
<td>3</td>
<td>Performance Mode – Monitor</td>
<td>The mobile application will have a START and STOP feature that will start recording/monitoring the selected readings and stop recording/monitoring the selected readings.</td>
</tr>
<tr>
<td>4</td>
<td>Troubleshoot Mode – Read Trouble Code</td>
<td>The mobile application will display any trouble codes detected, and display them on screen.</td>
</tr>
<tr>
<td>5</td>
<td>Store Readings</td>
<td>Data read will be stored in the internal History database. This will occur automatically when Performance Mode is running and recording, as well as when a report has been generated in the Troubleshoot Mode.</td>
</tr>
<tr>
<td>6</td>
<td>Database (Trouble Code) - Comparison</td>
<td>The mobile application will have a Trouble Code database that is stored on the phone. This database will contain all generic Trouble Codes that will be used for comparison in Troubleshoot Mode.</td>
</tr>
<tr>
<td>7</td>
<td>Database (History)</td>
<td>The mobile application will have a History database that is stored on the phone. This database will contain all past readings on a particular vehicle.</td>
</tr>
</tbody>
</table>

**Table 1 - 1 Requirements**
2. Meta Architecture

2.1 Architectural Vision

The architectural structure of the Auto Performance Analyzer was created in a way that would allow the system to be structurally independent. The goal was to make the software have low coupling and high cohesion. The system was separated into five layers: the GUI layer, the Event Management layer, the Database Management layer, the Processing layer, and the Bluetooth layer. These layers were created so that the architectural design would follow the teams guiding principles.

2.2 Guiding Principles

2.2.1 Accuracy

The system shall produce correct data in real time that will match the reading one would get from the car gauges.

2.2.2 Reliability

The system shall consistently perform in real time as stated in the systems requirements and do so without the system failing do to normal intended use. The system also shall not interfere with the driver’s normal operation of the vehicle where the system is installed.

2.2.3 Ease of Use

The system shall be designed so that its user interface is simple to use, with little or no knowledge of the product. Along with this, the initial setup and installation of the product shall be done with the same care in mind.

2.2.4 Maintainability

The system shall be designed so that each layer is independent of the other layers. Also, that each layer is responsible for a defined functionality and that no other layer does this functionality.

2.2.4 Portability

The system shall be designed so that it may be easily adapted to work on other systems such as IOS and Windows in future.
2.3 Design Evolution

The initial design consisted of only four layers: input, output, processing, and database layers (shown in Figure 2 - 1). Too many of the subsystems in each layer were dependent on subsystems from other layers, mainly the processing layer. That design failed to maintain low coupling and modularity. We redesigned our architecture so that there would be a designated Subsystem in each layer to handle communication with other layers. This design allows each layer to do whatever functionality it must without affecting the data sent to another layer. This change was important, because it allow for easily adding new subsystems to layers later in the implementation if there is a need.

![Figure 2 - 1 Initial architectural design](image)

The initial design called for an Input Layer, where user input and Bluetooth information were captured. After going through some scenarios, we realize this design did not make much sense. We replaced the Input layer with a GUI Layer and added a separate Bluetooth Layer. This changed created a clearly defined functionality for these layers that was missing before (Figure 2 - 2).
The other main design change was that originally the Database layer contained a History subsystem and an Error Code subsystem. The team decided that these in fact should not be subsystems, but just databases. They did not do any specific functionality. They simply were where the data was stored. The team decided to use two separated databases instead of one big database to gain performance. One database would be simpler to use and one less component of the overall system, but the team felt that queries would become slow.

Figure 2 - 2 Refinement of initial architectural design
3. Layer Definition Section

This section describes Auto Performance Analyzer’s application (APA) system architecture, the overall structure of how the system will be built. The architecture design is shown in Fig 3-1. APA’s system architecture consists of five major layers:

![Architectural diagram of system](image)

*Figure 3 - 1 Architectural diagram of system*
3.1 GUI Layer Definition

3.1.1 Definition
This layer will get the user’s input via the mobile device’s screen; pass the parameters list to the Event Management layer. It will also be in charge of displaying the information obtained from the Event Management layer.

3.1.2 Functional Description
The user will select the desired parameters on the mobile device using the provided GUI checklist in the application. Upon completing the parameter desired list, this layer will pass the list to the Event Manager Layer through the I/O controller subsystem. The Event Management layer through the processing handler; will pass the list to the processing layer, to convert such parameters to the corresponding OBD-II commands. The Processing layer will pass the converted command list to the Bluetooth OBD-II adapter connected to the car’s OBD-II port. When the car replies with the actual parameter’s information, the Bluetooth layer will send the cars’ data back to the Processing layer to be processed. This layer will receive the information from the Event Management layer adapter via the I/O controller subsystem, which then it will passed to the presentation subsystem that will display the info on the mobile device’s screen.

3.2 Processing Layer Definition

3.2.1 Definition
The Processing layer will receive the parameters list from the Event Management layer or actual parameter data from the Bluetooth Layer and code/decode the information, and will perform processing and the formatting into decimal form.

3.2.2 Functional Description
Upon receiving the parameters list from the Event Manger layer through the processing handler, the Processing layer will convert this list into a stream of commands that the car’s On-board computer can understand, and will pass the converted list to the Bluetooth layer. Then it will get from the same Bluetooth layer the actual car’s parameter data; which the Processing layer will convert and process to decimal format that it will be required either to be displayed on the mobile device’s screen through the GUI event handler or stored on the database file trough the Data Base controller in the DBMS Layer.

3.3 Event Manager Layer Definition

3.3.1 Definition
The Event Manager layer will be in charge of managing and routing the information obtained from all the other layers except the Bluetooth Layer.
3.3.2 Functional Description
This layer will route the users parameter selection list from the GUI Layer, actual parameters information data from the Processing layer and route the data necessary back to the GUI layer to be displayed in the mobile devices’ screen or to be passed to the Storage Management layer to be saved to the spreadsheet file.

3.4 Database Management System Layer Definition

3.4.1 Definition
The Database Management System (DBMS) Layer is responsible for storing the data in a spreadsheet or a file in the form of a table. This layer will also be responsible for retrieval of the Trouble Code Description information.

3.4.2 Functional Description
After receiving the data from the Event Manager layer, this layer will be responsible for the formatting of the data (if necessary) in a table form (spreadsheet) that will contain the time, parameters reading, and the mobile device sensors reading (such as GPS, Accelerometers, etc. if they are available on the mobile device). This layer will also be responsible for managing and retrieval of the Trouble Code Description database information. The information from this layer will be sent back to the Event Manager Layer, through the DBMS Event Handler, to be passed subsequently to the GUI layer if needed to display it.

3.5 Bluetooth Layer Definition

3.5.1 Definition
The Bluetooth Layer will send and receive data to and from the STN1170 OBD-II Adapter. Such data will be passed to the Processing Layer to be processed.

3.5.2 Functional Description
The user’s parameter selection list will be received already converted to the right format from the Processing Layer. This list of commands will be transmitted via Bluetooth to the STN1170 OBD-II Adapter. When the car’s On-board computer responds with the car’s actual parameter data, the OBD-II Adapter will send the data to back to the application, which will be received by the Bluetooth Layer and passed back to the Processing Layer for processing.
4. GUI Layer

In this section, the GUI layer is described in some detail in terms of its specific subsystems. The GUI layer handles all interactions of the system through the mobile application, via the graphical user interface. This layer contains three subsystems: User Input, I/O Controller, and Presentation. Each subsystem is described below.

4.1 User Input

4.1.1 General
The User Input subsystem belongs to the GUI Layer. All user inputs will be taken through the Auto Performance Analyzer mobile application via touchscreen.

![User Input subsystem inter-layer interfaces](image)

**Figure 4 - 1 User Input subsystem inter-layer interfaces**

4.1.2 Assumptions
Assumptions made:

- Bluetooth adapter is connected to the vehicle’s OBD-II port
- Bluetooth adapter is in working condition
- Mobile device is in working condition
- Mobile device is synchronized with Bluetooth adapter

4.1.3 Responsibilities
The User Input subsystem will be responsible for handling all inputs from the user through the GUI on the mobile application. Its sole responsibility is to capture the inputs from the user and formats that data into simple instructions before sending it to the I/O Controller. Types of inputs include:
- Vehicle information
- List of parameters to be checked, monitored, and/or displayed
- User controls (operational and configuration controls)

4.1.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputListener</td>
<td>Method will listen for user input</td>
<td>User Input</td>
<td>Captured user inputs</td>
</tr>
<tr>
<td>formatData</td>
<td>Formats that data into instructions</td>
<td>Captured user inputs</td>
<td>Formatted user instructions</td>
</tr>
</tbody>
</table>

Table 4 - 1 User Input subsystem inter-layer interfaces

4.1.5 Subsystem Public Interfaces

External Interfaces – This subsystem will use the mobile device’s touchscreen for input.

4.2 Presentation Subsystem

4.2.1 General

The Presentation subsystem belongs to the GUI Layer. This subsystem will unpack and format the instructions received, and output the results to the device’s screen through the Auto Performance Analyzer mobile application.

![Presentation subsystem inter-layer interfaces](image)

Figure 4 - 2 Presentation subsystem inter-layer interfaces
4.2.2 Assumptions

- Bluetooth adapter is connected to the vehicle’s OBD-II port
- Bluetooth adapter is in working condition
- Mobile device is in working condition
- Mobile device is synchronized with Bluetooth adapter

4.2.3 Responsibilities

The Presentation subsystem will handle all outputs to the mobile device’s screen. Data coming in to this subsystem will be formatted before outputting to screen. Types of outputs include:

- Menus
- Charts/Graphs
- Diagnostic Trouble Codes and corresponding descriptions
- Real-time performance monitoring of parameter readings

4.2.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Format data to be outputted to screen</td>
<td>Data</td>
<td>Formatted data</td>
</tr>
</tbody>
</table>

Table 4 - 2 Presentation subsystem inter-layer interfaces

4.2.5 Subsystem Public Interfaces

External Interfaces – This subsystem will use the mobile device’s touchscreen for output.

4.3 I/O Controller Subsystem

4.3.1 General

The I/O Controller subsystem belongs to the GUI Layer. This subsystem will package data received and determines where this data will eventually go. It will also control the flow of data coming to and from the Event Manager Layer.
4.3.2 Assumptions
There are no assumptions for this subsystem.

4.3.3 Responsibilities
Incoming data will be passed on to where it is needed, and will be packaged if needed. This subsystem acts much like a router would. Its functionalities include:

- Package data
- Determine destination
- Control data flow to and from Event Manager layer

4.3.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>packData</td>
<td>Packages data</td>
<td>Data</td>
<td>Packaged data</td>
</tr>
</tbody>
</table>

*Table 4 - 3 I/O Controller subsystem inter-layer interfaces*

4.2.5 Subsystem Public Interfaces
N/A
5. Event Manager Layer

In this section, the Event Manager layer is described in some detail in terms of its specific subsystems. This layer will determine where data should go, as well as directing that flow of data between other layers. This layer contains three subsystems: GUI Event Handler, Processing Event Handler, and DBMS Event Handler. Each subsystem is described below.

5.1 GUI Event Handler Subsystem

5.1.1 General
The GUI Event Handler subsystem belongs to the Event Manager layer. This subsystem handles the flow of data between the GUI layer and the Event Manager layer, as well as directing the input data to either the Processing Event Handler or the DBMS Event handler.

5.1.2 Assumptions
There are no assumptions for this subsystem.

5.1.3 Responsibilities
Any data that needs to be sent to the GUI layer will have to first be handled by the GUI Event Handler to ensure proper formatting is kept before directing it the GUI layer. If the data is to be fetched and/or processed, the GUI Event Handler will direct that data to the Processing Event Handler. If the data is to be fetched and/or processed, the GUI Event Handler will direct that data to the Processing Event Handler. If the data is to be fetched and/or processed, the GUI Event Handler will direct that data to the Processing Event Handler.
be stored or fetched in or from the database, such as vehicle information, the event handler will pass that data to the DBMS Event Handler.

### 5.1.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>routeData</td>
<td>Determine where data will be sent to</td>
<td>Packaged data</td>
<td>Packaged data</td>
</tr>
</tbody>
</table>

*Table 5-1 GUI Event Handler subsystem inter-layer interfaces*

### 5.1.5 Subsystem Public Interfaces

N/A

### 5.2 Processing Event Handler Subsystem

#### 5.2.1 General

The Processing Event Handler subsystem belongs to the Event Manager layer. This subsystem handles the flow of data between the Processing layer and the Event manager layer. The subsystem will also interact with the GUI and DBMS Event Handler to ensure that the data is directed to the correct place.
5.2.2 Assumptions
There are no assumptions for this subsystem.

5.2.3 Responsibilities
The Processing Event Handler will take the data received, such as a list of parameters to be monitored, and will send those requests to the Processing layer to be fetched. Once the requested data is returned, the subsystem will direct that data to the GUI Event Handler for proper formatting before sending it to the GUI layer. Data being transmitted by the vehicle’s onboard diagnostic system will also be stored in the database, thus the Processing Event Handler will have to interact with the DBMS Event Handler as well. Responsibilities of the Processing Event Handler include:

- Directing the flow of data to and from the Processing layer
- Determine where the data received should go, i.e. storing into database and/or return data to GUI for display

5.2.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>routeData</td>
<td>Handles data to be sent to the Processing layer</td>
<td>Packaged data</td>
<td>Packaged data</td>
</tr>
</tbody>
</table>

Table 5 - 2 Processing Event Handler subsystem inter-layer interfaces

5.2.5 Subsystem Public Interfaces
N/A

5.3 DBMS Event Handler Subsystem

5.3.1 General
The DBMS subsystem belongs to the Event Manager layer. This subsystem will handle all data sent to and from the database.
5.3.2 Assumptions
There are no assumptions for this subsystem.

5.3.3 Responsibilities
Send and receive requests to the database will be managed by the DBMS Event Handler. This subsystem will format query requests into simple instructions that will be passed on to the Database Controller in the DBMS layer. Storing data will be done in a similar manner. The DBMS Event Handler will also communicate with the other two event handlers in the Event Manager layer to ensure the proper flow of data. This subsystem will be responsible for:

- Handling communication with the DBMS layer
- Formatting all requests to simple store/fetch instructions to be sent to DB Controller

5.3.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryRequest</td>
<td>Requests data from the database</td>
<td>Data requests</td>
<td>Requested Data</td>
</tr>
<tr>
<td>storeData</td>
<td>Data to be stored will first be handled here</td>
<td>Data</td>
<td>Data</td>
</tr>
</tbody>
</table>

Table 5 - 3 DBMS Event Handler subsystem inter-layer interfaces

5.3.5 Subsystem Public Interfaces
N/A
6. Processing Layer

In this section, the Processing layer is described in some detail in terms of its specific subsystems. This layer contains two subsystems: Processing Controller and Bluetooth Processing. Each subsystem is described below.

6.1 Processing Controller Subsystem

6.1.1 General
The Processing Controller Subsystem belongs to the Processing Layer. This subsystem will do the actual calculations and data processing of the user’s input as well as the data received from the Bluetooth adapter.

6.1.2 Assumptions
There are no assumptions for this subsystem.

6.1.3 Responsibilities
The Processing Controller subsystem will be the bridge that connects the Event Manager layer and the Processing layer. This subsystem will send requests, such as speedometer readings, to the Bluetooth adapter through the Bluetooth Processing subsystem. Once the data is received from the Bluetooth...
Processing subsystem, the Processing Controller will have to format and package that data to be sent back to the GUI or Database layer. This subsystem will be responsible for:

- Direct flow to and from Event Manager layer
- Format data
- Calculations

### 6.1.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>unpack</td>
<td>Unpacks data to be processed</td>
<td>Packaged data</td>
<td>Unpackaged data</td>
</tr>
<tr>
<td>packData</td>
<td>Packages data to be returned</td>
<td>Converted data</td>
<td>Formatted data</td>
</tr>
</tbody>
</table>

*Table 6-1 Processing Controller subsystem inter-layer interfaces*

### 6.1.5 Subsystem Public Interfaces

N/A

### 6.2 Bluetooth Processing Subsystem

#### 6.2.1 General

The Bluetooth Processing Subsystem belongs to the Processing Layer. This subsystem will handle the requests being sent to the Bluetooth adapter, by first formatting the requests to the correct format, so that the Bluetooth adapter will send the codes that the ECU can understand. Once the requested data is returned from the Bluetooth module, this subsystem will have to convert that raw data into readable data that will be returned.
6.2.2 Assumes
- Bluetooth adapter is connected to the vehicle’s OBD-II port
- Bluetooth adapter is in working condition

6.2.3 Responsibilities
The Bluetooth Processing subsystem will be responsible for:
- Converting parameter requests into proper format that the Bluetooth will send to the ECU
- Converting raw data to readable and comprehensible format
- Communicating with the Bluetooth layer

6.2.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>requestReadings</td>
<td>Requests data from Bluetooth</td>
<td>Parameter requests</td>
<td>Raw data from Bluetooth</td>
</tr>
<tr>
<td>dataConversion</td>
<td>Converts the raw data from the Bluetooth adapter into comprehensible data</td>
<td>Raw data from Bluetooth</td>
<td>Converted data</td>
</tr>
</tbody>
</table>

Table 6 - 2 Bluetooth Processing subsystem inter-layer interfaces

6.2.5 Subsystem Public Interfaces
N/A
7. **Bluetooth Layer**

In this section, the Bluetooth layer is described in some detail in terms of its specific subsystems. The Bluetooth layer captures all data transmitted from the vehicle’s onboard diagnostic system. This layer contains two subsystems: Data Acquisition and Sync/Pair. Each subsystem is described below.

### 7.1 Data Acquisition Subsystem

#### 7.1.1 General

The Data Acquisition Subsystem belongs to the Bluetooth Layer. This subsystem will work with the mobile device’s Bluetooth system to send requests and receive data.

![Data Acquisition subsystem inter-layer interfaces](image)

**Figure 7 - 1** Data Acquisition subsystem inter-layer interfaces

#### 7.1.2 Assumptions

- Bluetooth adapter is connected to the vehicle’s OBD-II port
- Bluetooth adapter is in working condition

#### 7.1.3 Responsibilities

The sole responsibility of the Data Acquisition subsystem is to fetch data from the vehicle’s onboard diagnostic system. Data fetched from the Bluetooth adapter will most likely be raw data that must be formatted (this will be done in the Bluetooth Processing subsystem in the Processing layer).
7.1.4 **Subsystem Inter-layer Interfaces**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetchData</td>
<td>Fetch data from the vehicle’s OBD-II system via Bluetooth</td>
<td>Data requests</td>
<td>Raw requested data</td>
</tr>
</tbody>
</table>

*Table 7 - 1* Data Acquisition subsystem inter-layer interfaces

7.1.5 **Subsystem Public Interfaces**

N/A

7.2 **Sync/Pair Subsystem**

7.2.1 **General**

The Sync/Pair Subsystem belongs to the Bluetooth Layer. Synchronizing the mobile device’s Bluetooth system to the OBD-II Bluetooth Adapter will be handled by the mobile device.

![Sync/Pair subsystem inter-layer interfaces](image1)

*Figure 7 - 2* Sync/Pair subsystem inter-layer interfaces

7.2.2 **Assumptions**

- Device must be within 30 feet of OBD-II STN1170 to maintain connection
- User has entered correct pairing code during initial setup, which will be provided
7.2.3 Responsibilities
This subsystem can be considered to be a separate entity from the mobile application, for its functions are handled by the mobile device and not that of the Auto Performance Analyzer mobile application. This subsystem is responsible for:

- Synchronizing the mobile device to the Bluetooth adapter

7.2.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkSync</td>
<td>Verify valid connection with a Bluetooth device</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 7 - 2 Sync/Pair subsystem inter-layer interfaces

7.2.5 Subsystem Public Interfaces
External Interfaces – STN1170 Bluetooth OBD-II Adapter attached to vehicle’s OBD-II Connector
8. Database Management System (DBMS) Layer

In this section, the DBMS layer is described in some detail in terms of its specific subsystems. The DBMS layer handles the storing and retrieval of information in the database. This layer contains three subsystems: DB Controller, History, and Diagnostic Trouble Code. Each subsystem is described below.

8.1 Database Controller (DB Controller) Subsystem

8.1.1 General
The DB Controller Subsystem belongs to the DBMS Layer. This subsystem will determine where to store the data received as well as locating the desired data.

![Database Controller Subsystem Diagram]

Figure 8-1 Database Controller (DB Controller) subsystem inter-layer interfaces

8.1.2 Assumptions
- Android device has minimum of 500mb of available storage space
8.1.3 Responsibilities
The DB Controller subsystem will handle all query requests and directing them to the appropriate database.

8.1.4 Subsystem Inter-layer Interfaces

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>routeRequest</td>
<td>Determine which database to route request to</td>
<td>Data requests</td>
<td>Actual Data from DB file</td>
</tr>
</tbody>
</table>

*Table 8 - 1 DB Controller subsystem inter-layer interfaces*

8.1.5 Subsystem Public Interfaces
N/A
# 9. Inter-Subsystem Data Flow

## 9.1 Overview

Relationship Mapping describes the data flow between layers and subsystems of the Auto Performance Analyzer.

## 9.2 Data Flow Definitions

The table below provides a description of the data that flows between each subsystem shown in figure 10-1 (located on next page). Data originating from the GUI Layer will begin with a G, Event Manager Layer will begin with E, Processing Layer will begin with P, DBMS Layer will begin with D, and Bluetooth Layer will begin with B.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Data</th>
<th>Element Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GUI Layer</strong></td>
<td>G1</td>
<td>User request sent to GUI by phone/tablet.</td>
</tr>
<tr>
<td></td>
<td>G2</td>
<td>User event sent from User Input to I/O Controller.</td>
</tr>
<tr>
<td></td>
<td>G3</td>
<td>User Input event sent from I/O Controller to GUI Event Controller.</td>
</tr>
<tr>
<td></td>
<td>G4</td>
<td>Update Presentation event sent from I/O Controller to Presentation.</td>
</tr>
<tr>
<td></td>
<td>G5</td>
<td>Send updated GUI Presentation to Android Device.</td>
</tr>
<tr>
<td><strong>Event Management Layer</strong></td>
<td>E1</td>
<td>Receive user input from GUI Event Controller.</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>Send Processing Event.</td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>Return Processed Event to GUI Event Handler.</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>Send Processed Data Event.</td>
</tr>
<tr>
<td></td>
<td>E5</td>
<td>Send/Retrieve Data.</td>
</tr>
<tr>
<td></td>
<td>E6</td>
<td>Return Data Event to Processing Event Handler.</td>
</tr>
<tr>
<td></td>
<td>E7</td>
<td>Return Data Event to GUI Event Handler.</td>
</tr>
<tr>
<td></td>
<td>E8</td>
<td>Return Display Data Event.</td>
</tr>
<tr>
<td></td>
<td>E9</td>
<td>Send Data Event to DBMS Event Handler.</td>
</tr>
<tr>
<td><strong>Processing Layer</strong></td>
<td>P1</td>
<td>Send Formatted Data</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>Send Formatted Bluetooth Data</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>Return Processed Data to Processing Event Handler.</td>
</tr>
<tr>
<td></td>
<td>P4</td>
<td>Return Collected Data From Bluetooth</td>
</tr>
<tr>
<td><strong>DBMS Layer</strong></td>
<td>D1</td>
<td>Query Historical Data</td>
</tr>
<tr>
<td></td>
<td>D2</td>
<td>Query Error Codes Data</td>
</tr>
<tr>
<td><strong>Bluetooth Layer</strong></td>
<td>B1</td>
<td>Send bluetooth data.</td>
</tr>
<tr>
<td></td>
<td>B2</td>
<td>Pass bluetooth values.</td>
</tr>
<tr>
<td></td>
<td>B3</td>
<td>Synchronize</td>
</tr>
<tr>
<td></td>
<td>B4</td>
<td>Transfers bluetooth data.</td>
</tr>
<tr>
<td></td>
<td>B5</td>
<td>Return Bluetooth Data</td>
</tr>
</tbody>
</table>

*Figure 9 – 1 Data Flow Definitions*
9.3 Producer-Consumer Relationship

The data flows between different layers and subsystems are defined in Table 10.1. The producer-consumer relationship is described in Table 10.2. The producer is the source of data element; the consumer is the destination of the data element.

![Diagram of Producer-Consumer Relationship]

**Figure 9-2** Producer consumer relationship
10 Operating System Dependencies Section

10.1 Overview

Each layer will be developed in Java using the Android Development SDK with the Eclipse IDE and will run on the Android OS for mobile phones or tablets.

10.2 GUI Layer

The GUI Layer will be dependent on the GUI libraries for Android. The Android application will be dependent upon the android.view.accessibility, android.graphics, and android.view.inputservice packages.

10.3 Processing Layer

The Data Processing Layer is dependent on Android multi-threading capabilities to synchronize displaying information to the user in the monitoring requirement with the DBMS layer that stores the data during the runs. The Data Processing layer will also make use of the Java Math Libraries to convert the requested data into a hexadecimal format the Bluetooth module can use to request the information from the OBD-II system of the vehicle.

10.4 DBMS Layer

The storage layer will be dependent on Android application storage for the device memory subsystem. The database subsystem will be dependent on Java and SQLite for the database storage using SQL to make requests and store data in the database. The files will be handled by Android OS and stored on the phone locally.

10.5 Bluetooth Layer

The Bluetooth will be developed with Android SDK to interface with the STN1170 Bluetooth OBDII and will be dependent on the conversion from the Processing Layer to communicate back and forth.
10. Requirement Mapping

This section describes the relationships between the requirements outlined in the SRS document and the architecture defined in this document. It will help our group to ensure that the requirements are covered by the architecture. Table 10-1 shows the relationships between each layer and each requirement. An X means that the requirement is covered by the layer with an X.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>GUI Layer</th>
<th>Event Manager Layer</th>
<th>Processing Layer</th>
<th>DBMS Layer</th>
<th>Bluetooth Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Real-Time Data Acquisition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.2</td>
<td>Performance Mode - Select Readings</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.3</td>
<td>Performance Mode - Monitor</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.4</td>
<td>Troubleshoot Mode - Vehicle Information</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Troubleshoot Mode - Comparison</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.6</td>
<td>Store Readings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.7</td>
<td>Database (Trouble Codes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>Database (History)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5.1</td>
<td>Real-Time Output</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5.2</td>
<td>Reliable Data Transfer</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 10 - 1 Requirements Mapping
11. Testing Considerations Section

11.1 Overview

The system architecture will be tested by team Overdrive to verify that the Auto Performance Analyzer fulfills all of the requirements laid out in the System Requirements Specification and in this Architectural Design Specification. Each component and subsystem will be designed to work independently and this independence will allow each module to be tested individually. Then it will be integrated to the final application which then it will be tested again to make sure there will not be integrity issues.

11.2 GUI Layer

This layer will get the user’s input via the mobile device’s screen; pass the parameters list to the Event Management layer. It will also be in charge of displaying the information obtained from the Event Manager layer.

11.2.1 Layer Testing Approach
Testing of this layer will verify that the GUI layer will be able to pass the user selection list to the Event Manager layer, and also be able to display the information obtained from the same layer.

11.3 Processing Layer

The Processing layer will receive the parameters list from the Event Manager layer or actual parameter data from the Bluetooth Layer and code/decode the information, and will perform processing and the formatting into decimal form.

11.3.1 Layer Testing Approach
The tests for this layer will verify that receives the parameters list from the GUI layer and translates this list into a command stream of the proper OBD-II codes that will be passed to the Bluetooth layer. There will also be tests that will verify that the data obtained from the car by the Bluetooth layer will be correctly translated to decimal format.

11.4 Event Manager Layer

The Event Manager layer will be in charge of managing and routing the information obtained from all the other layers except the Bluetooth Layer.
11.4.1 Layer Testing Approach
This layer will be tested to make sure that it will correctly route the information obtained from the processing layer or the Storage Management layer. Next, that data is properly format for the GUI Layer. Last is that the data is not corrupted when passes to and from the GUI to the Processing Layer and to the DBMS Layer.

11.5 DBMS Layer
The DBMS Layer is responsible for storing the data in a spreadsheet or a file in the form of a table. This layer will also be responsible for retrieval of the Trouble Code Description information.

11.5.1 Layer Testing Approach
The tests for this layer will include obtaining the data from the processing layer and verify that the data will preserve its integrity when is formatted and stored in the spreadsheet. Additional tests will be included to verify that the Storage management layer deliver the right information from the Diagnostic Trouble Code database to the GUI layer through the DBMS Handler in the Event Manager Layer.

11.6 Bluetooth Layer
The Bluetooth Layer will send and receive data to and from the STN1170 OBD-II Adapter. Such data will be passed to the Processing Layer to be processed.

11.6.1 Layer Testing Approach
This layer will be tested within the tests performed for the Processing Layer. That is, it will verify that the application can communicate to the OBD-II adapter and received information from it, and that it will not lose connection to the OBD-II adapter.

11.7 Integrity Testing Approach
The Integrity testing will verify proper operation of all of the functions and menus of the Auto Performance Analyzer Application to ensure that it will satisfy all of the requirements from the customer and that it will not be interlayer compatibility issues when integrated to the main application.