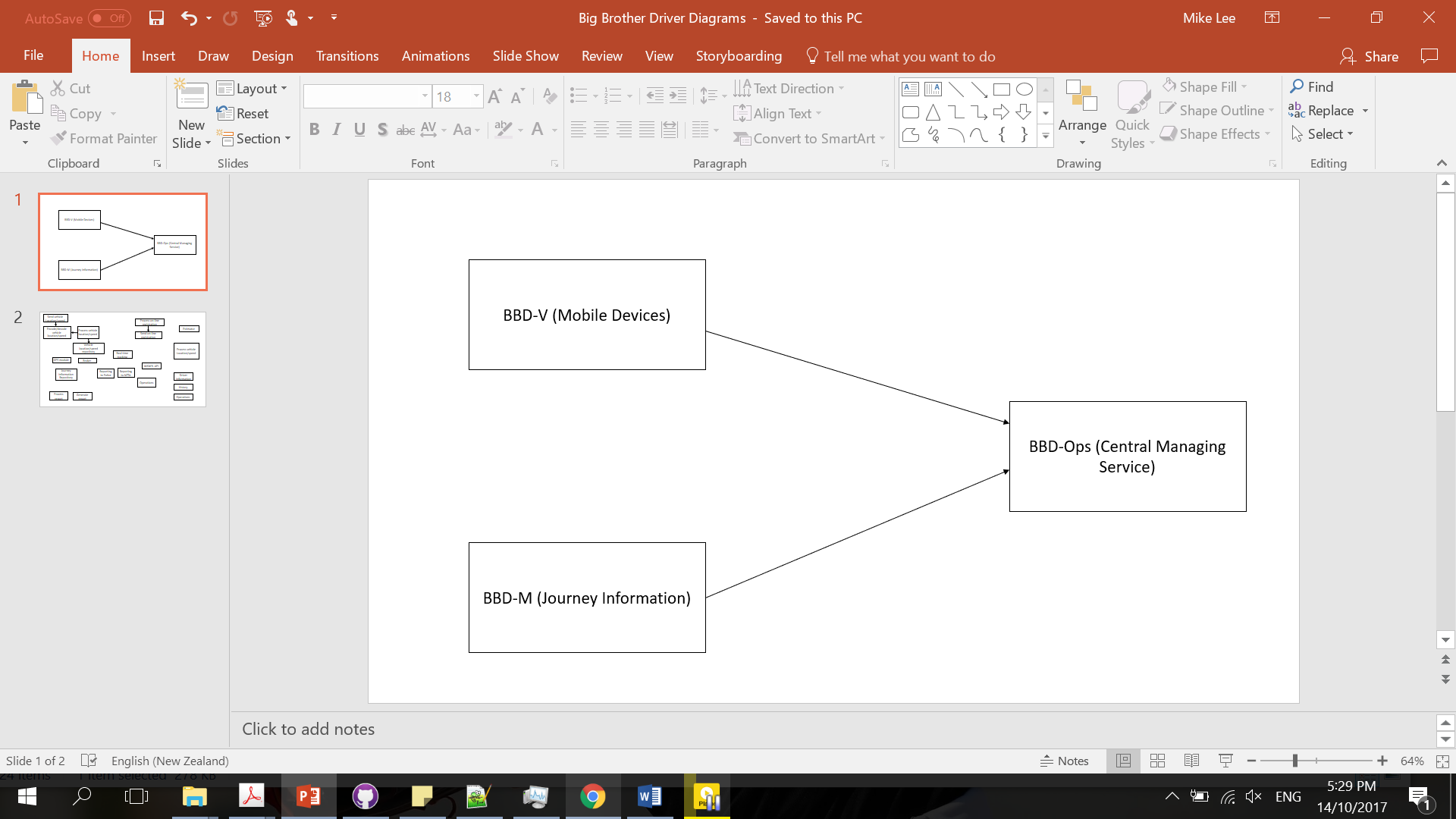
SE325 Assignment 2 Report

840454023, elee353

* A section that describes your architecture. I am assuming you will use diagrams as the main description aid, plus some explanatory text. I don't expect the text to be more than about 2-3 pages.
* A section that describes what tactics you have used in your architecture, and a brief explanation why you chose the tactics you did. I don't expect much more than a paragraph for each tactic (for example, you don't need to explain what the tactic means). However it must be clear as to how your architecture results from the use of the tactics.
* A section that justifies how your architecture meets the stated quality attributes. If you have done a good job of explaining your use of tactics, this won't need to be much. It may, for example, consist of only one paragraph, but even so should not need to be very long.
* <https://docs.google.com/document/d/149BcZhZdbkFoxrB3Hh1Sw7DNiSuq4cchPjYWak-5UCw/edit#>

Overall:



The system consists of three major components: BBD-V, BBD-M, and BBD-Ops.

* BBD-V is the mobile app installed on the mobile devices for use recording the journeys of vehicles.
* BBD-M is the app for use by relevant users for accessing the journey information.
* BBD-Ops is the central service for managing and distributing the information gathered by the apps.
* The app that is installed on the mobile devices (BBD-V) for use recording the journeys of vehicles
* an app for use by relevant users for accessing the journey information (BBD-M),
* and a central service for managing and distributing the information gathered by the apps (BBD-Ops).

Possible minimum set:

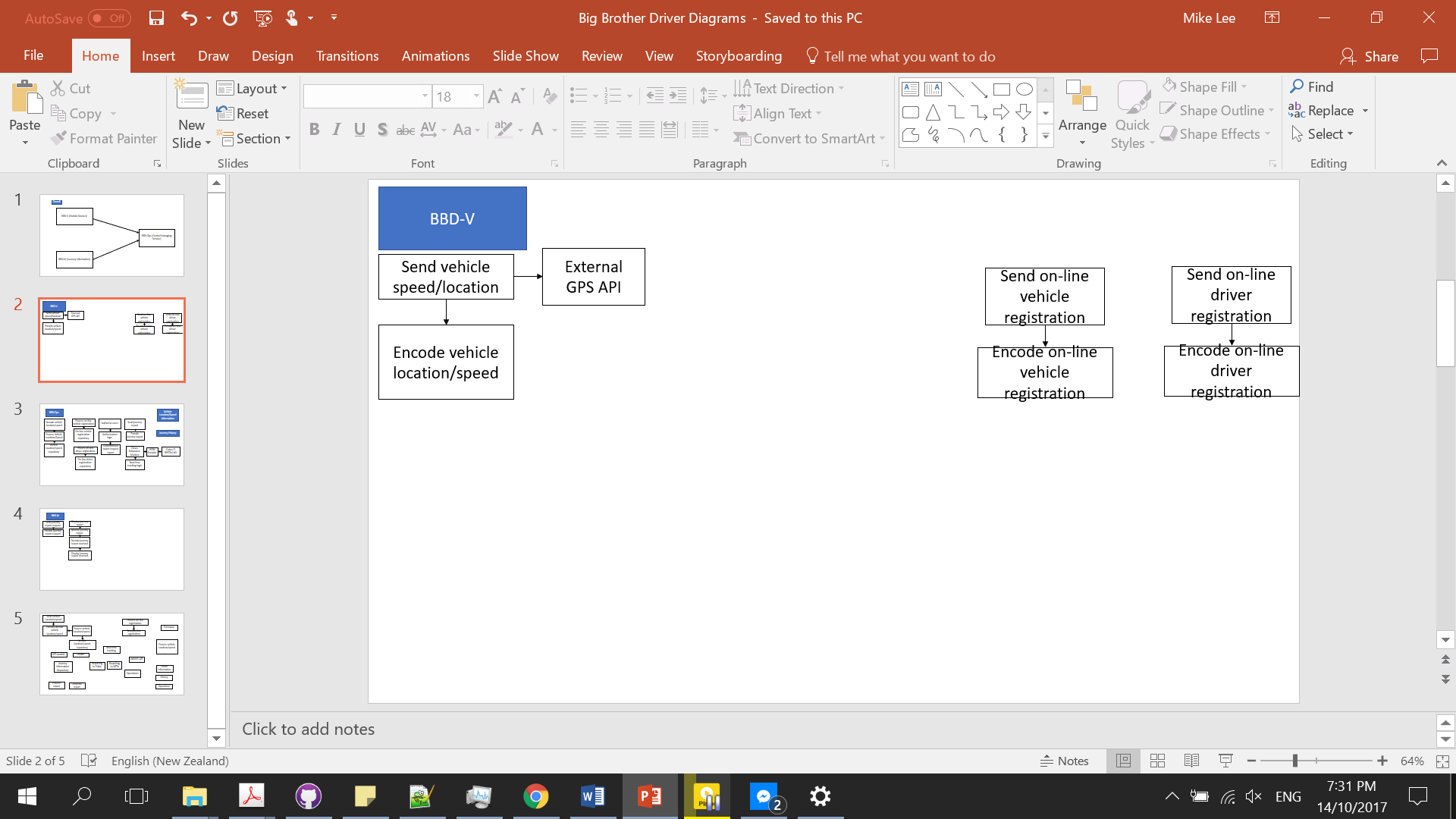
◦ Uses structures — what chunks there are and how they interact

◦ Process structure — how the chunks actually run and communicate

◦ Deployment structure — where the chunks actually run

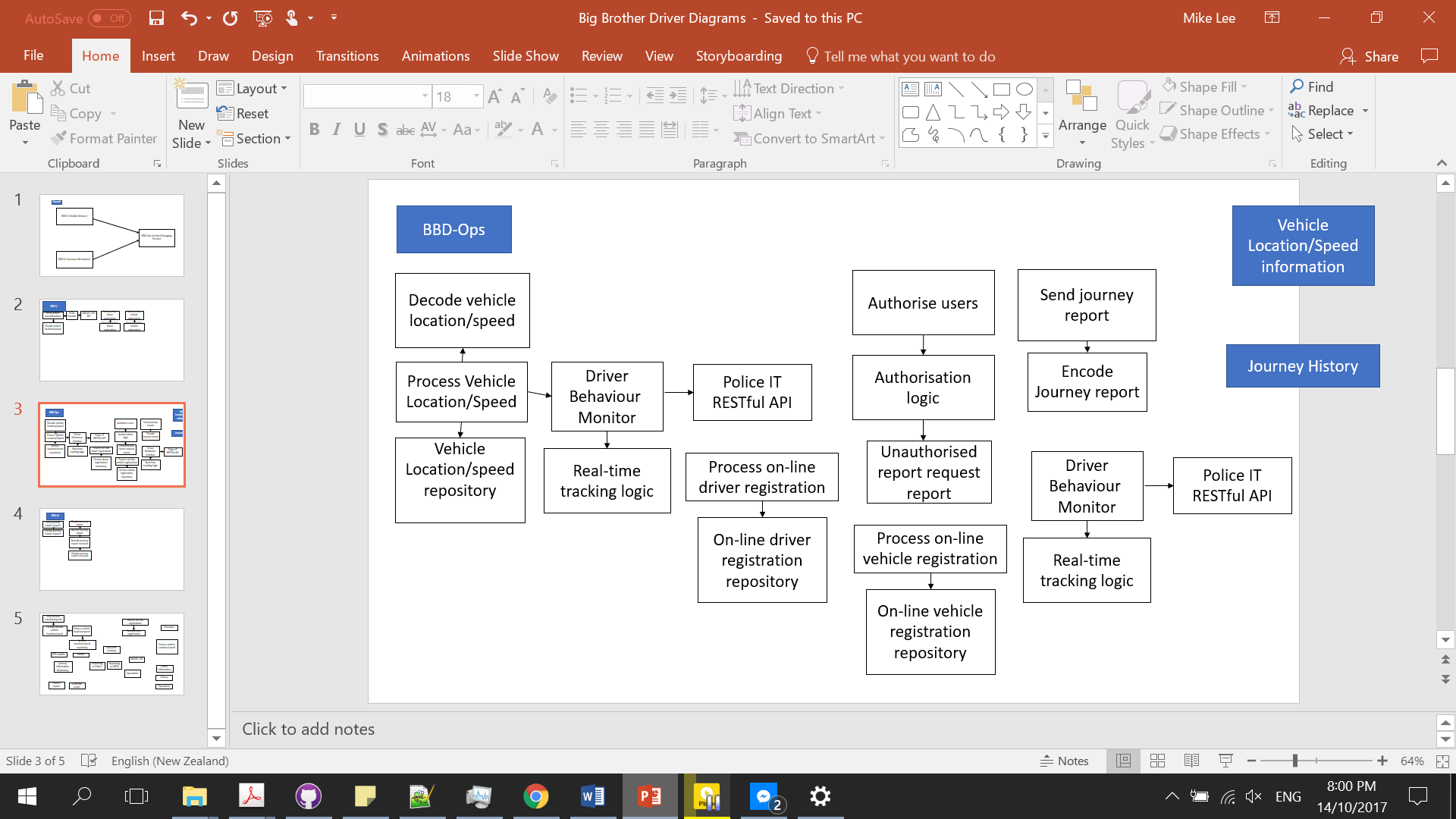
### Module structures

## BBD-V



* The ‘send vehicle speed/location’ module uses the ‘JSON encoder’ and the WKWYL GPS API to receive real-time details about the speed limit and road-layout for the specified GPS location. The ‘send vehicle speed/location’ module also uses the ‘Encode vehicle location/speed’ to package the speed/location data before sending.
* The ‘send on-line vehicle registration’ module is responsible for sending encoded vehicle registration information. The ‘send on-line vehicle registration’ module uses the ‘Encode on-line driver registration’ module to package the data before sending it.
* Similarly, the ‘send on-line driver registration’ module is responsible for sending encoded vehicle registration information. The ‘send on-line driver registration’ module uses the ‘Encode on-line driver registration’ module to package the information before sending.

## BBD-Ops



The ‘Process Vehicle Location/Speed’ module uses the ‘decode vehicle location/speed’ module to unpack the data received. The ‘Process Vehicle Location/Speed’ module then stores the received and calculated location/speed data using the ‘vehicle Location/speed repository’ module.

The ‘Driver Behaviour Monitor’ module receives the data from the ‘Process Vehicle Location/Speed’ module. The ‘Driver Behaviour Monitor’ module then decides if it should perform real-time tracking or report to the police.

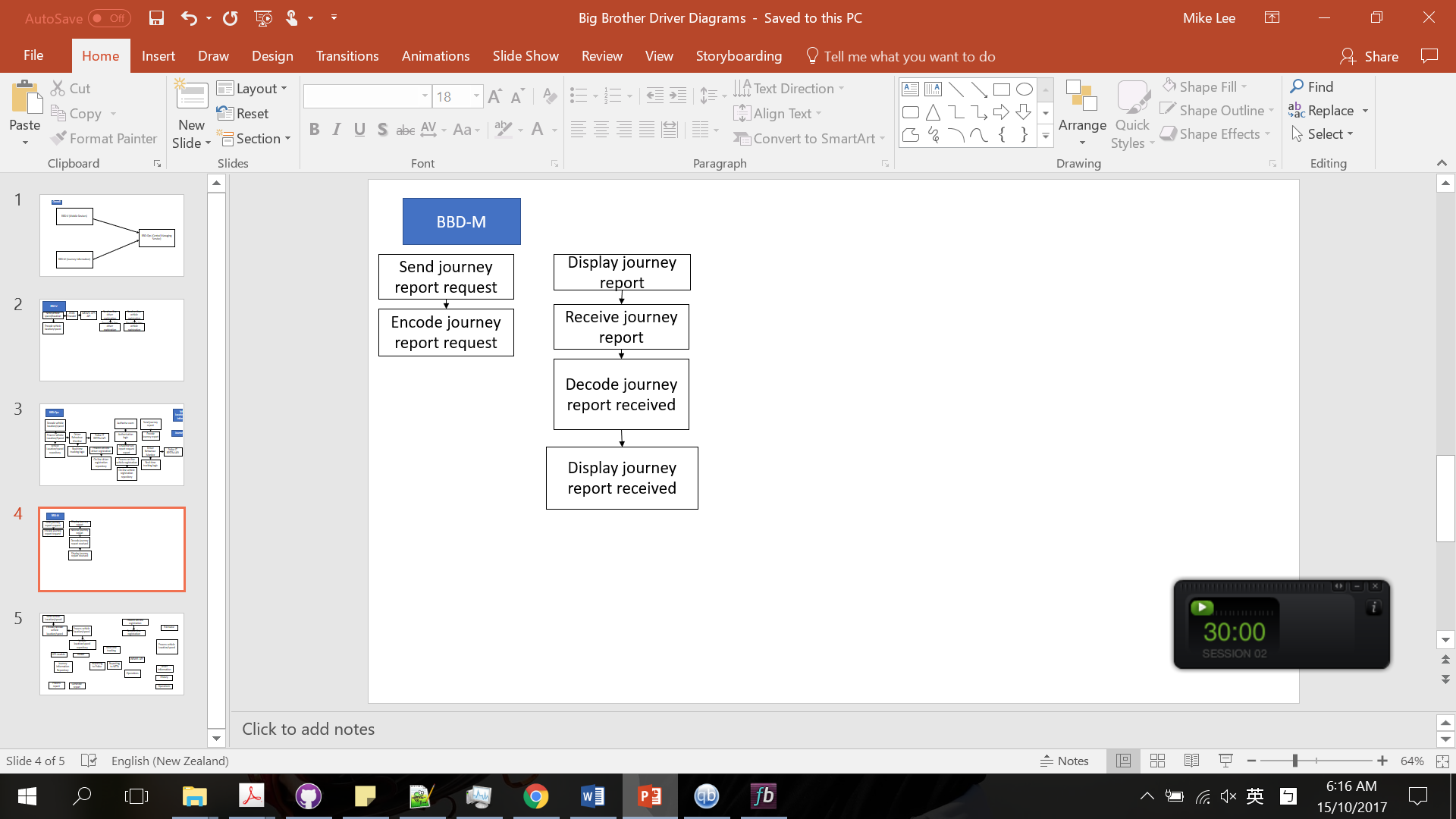
The ‘Real-time tracking logic’ module is used by the ‘Driver Behaviour Monitor’ module to allow real-time tracking for the police and the parents for special driver types.

The ‘Police IT RESTful API’ module is used by the ‘Driver Behaviour Monitor’ module to report erractical driving behaviours or excessive speed to the police immediately.

The ‘Process on-line driver registration’ module processes the driver information received and persists it using the ‘On-line driver registration repository’.

## BBD-M

The **Display Report** module receives and unpackages the reports and displays them. It uses the **Encode/Decode** module)



It uses the Bus Location/speed repository to store the information it has received.

Figure 1

### Component and Connector structures

There are two different types of components:

“active” processes and “passive” processes. Active processes run independently from all other components

and can initiate communication, in the same way (for example) a Java Thread can. Passive processes cannot

initiate communication; they can merely respond to requests. They could be regarded a “servers”, and so be in a

client/server relationship with active processes, but they may not exist a real processes, depending upon how they

are deployed (see below).

All processes in the system correspond to a module (although there are a couple of modules that aren’t also

processes), and so the same names are used for both the modules and processes. There are as many **Send Bus**

**Location/speed** processes as there are buses and as many **Display Estimates** processes as there are displays. The

number of **Process Bus Location/speed** processes is currently unspecified—see the discussion below. There are

as many **Estimator** processes as there are buses, and there is one each of **Transmitter** and **Operations** processes.

All wireless communication is asynchronous and all other communication is synchronous.

### Allocation Structures

### Patterns

Client-Server / N-Tier Systems

### Tactics

### Scenario 1

* The mobile devices are expected to communicate using the owner's data plans.
* Data sent or received during the course of an average year is not more than 5MiB.
* The details are delivered formatted in JSON, taking not more than 512 bytes.
* Assume send once every minute: 512 bytes \* 30 days \* 60 min \* 20 journeys = 18432000 bytes = 18.4 Mb
* This should be affordable for most mobile phone users with average data plans.
* 5 Mib / 512 bytes gives 10240 times to send message

### Scenario 2

### Scenario 3

### Scenario 4

### Scenario 5

### Justification

### Calculation

The details are delivered formatted in JSON, taking not more than 512 bytes.

### From last year:

#### increase computational efficiency

There are several places where use of an inefficient algorithm may result in

the system not meeting the performance requirements, such as the estimation algorithm, or the algorithm used to construct the messages send around the system.

#### reduce computational overhead

There are choices to be made as to whether to co-locate a passive process with an active process. Doing so will reduce the overhead of the remote communication that would be required if they aren’t co-located.

#### manage event rate

There is a choice to be made about the frequency with which the bus subsystem sends out its location and speed.

#### control frequency of sampling

Even if the bus sends out its location and speed at the stated rate, there is a choice to be made about how many of the messages are actually processed.

#### introduce concurrency

There are at least as many processes as there are buses. There are several ways in which

these processes can be allocated to processors. Some aspects of the performance can be improved by using more processors.

#### increase available resources

Another choice that can be made is that faster processors can be used, rather than more processors (or both). There is also the possibility of using separate communication links for different parts of the system.

#### maintain multiple copies of either data or computations (caching?)

Multiple copies of the Estimator process are used, one per bus. It is also possible to duplicate the route information or history information.