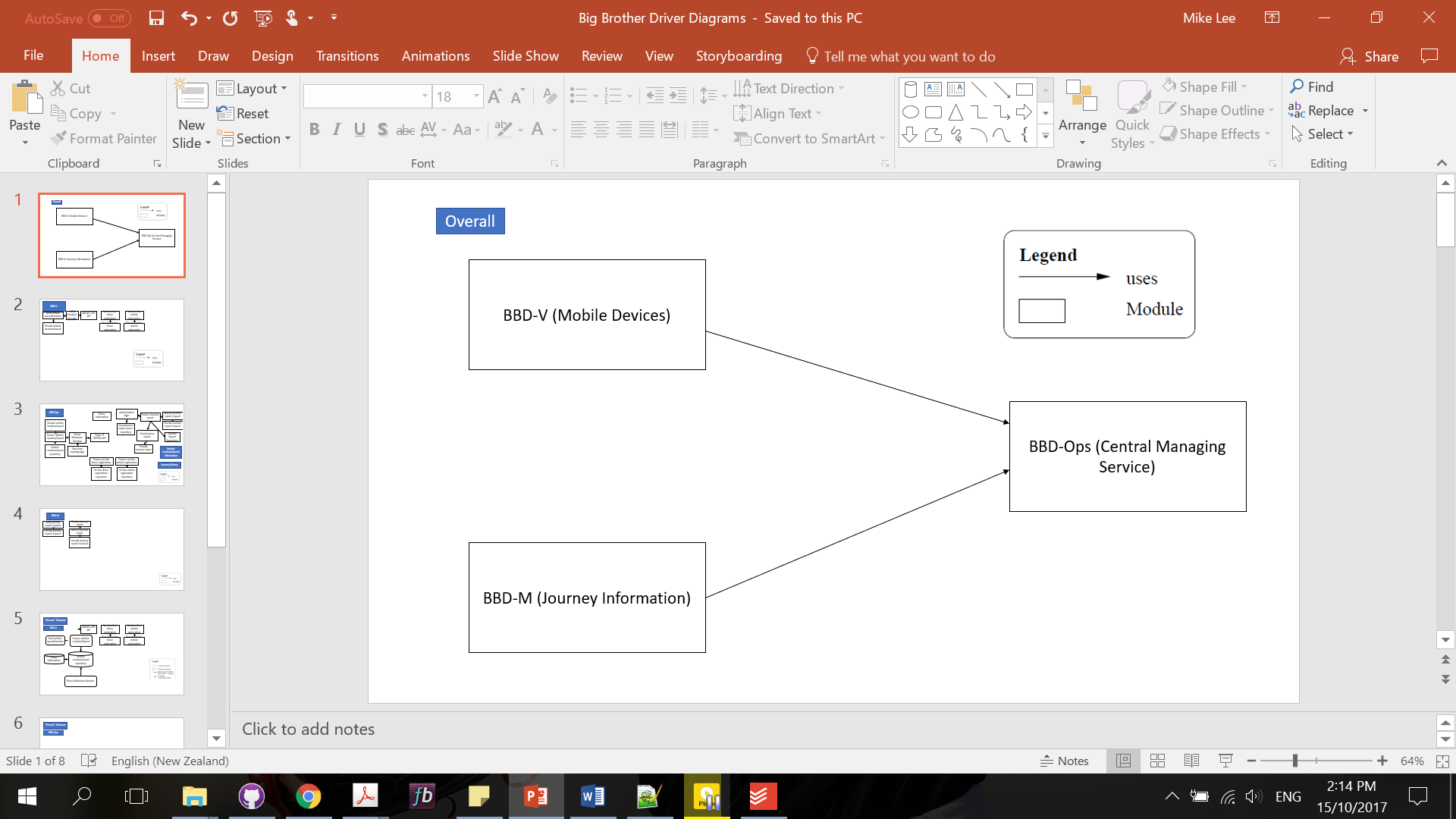
SE325 Assignment 2 Report

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#>

## Architecture Section

## 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 recording the journeys of vehicles.
* BBD-M is the app for accessing the journey information by relevant users.
* BBD-Ops is the central service for managing and distributing the information gathered by the apps.

## Possible minimum set:

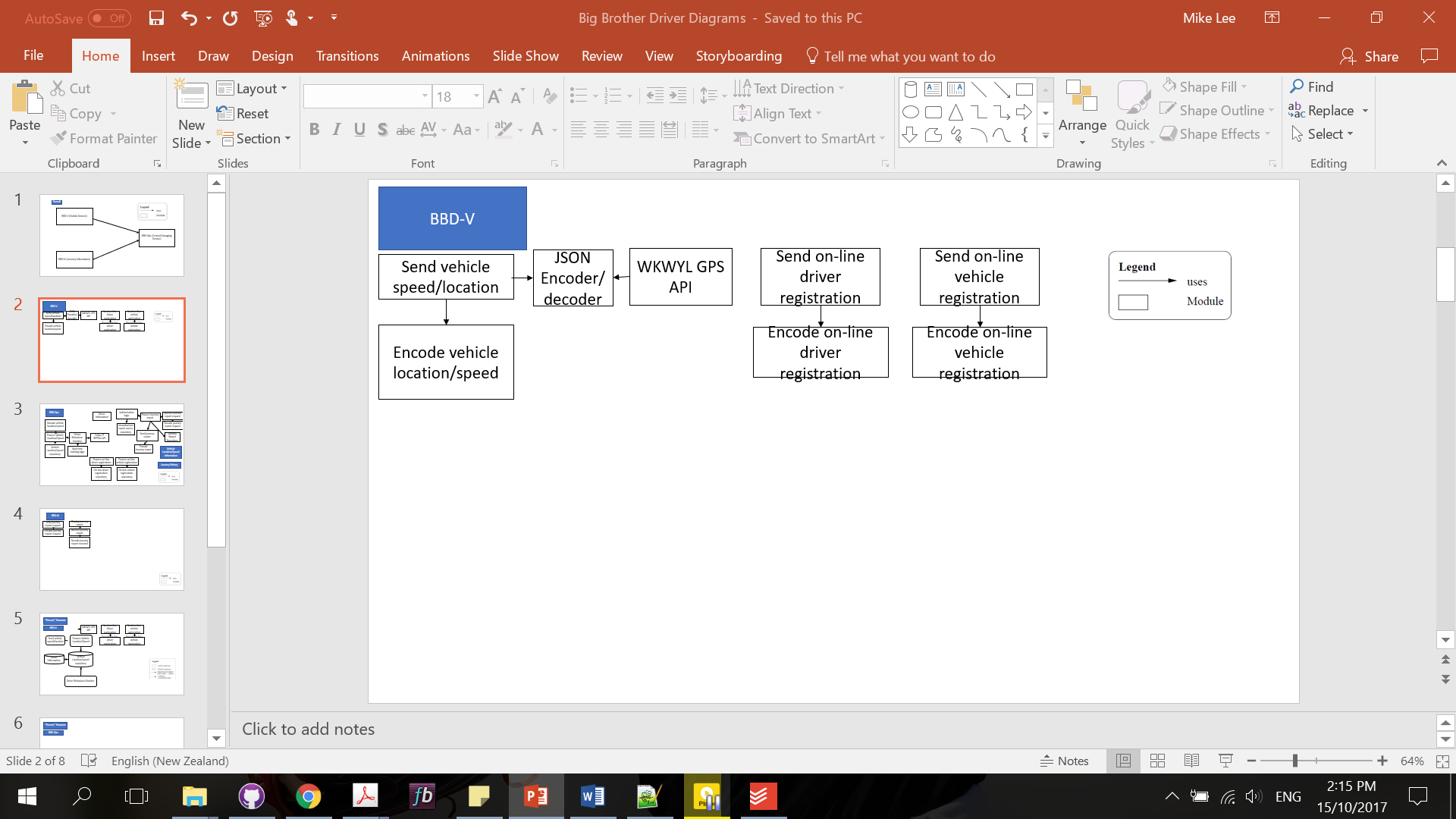
◦ Uses structures — what chunks there are and how they interact

◦ 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’ module 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

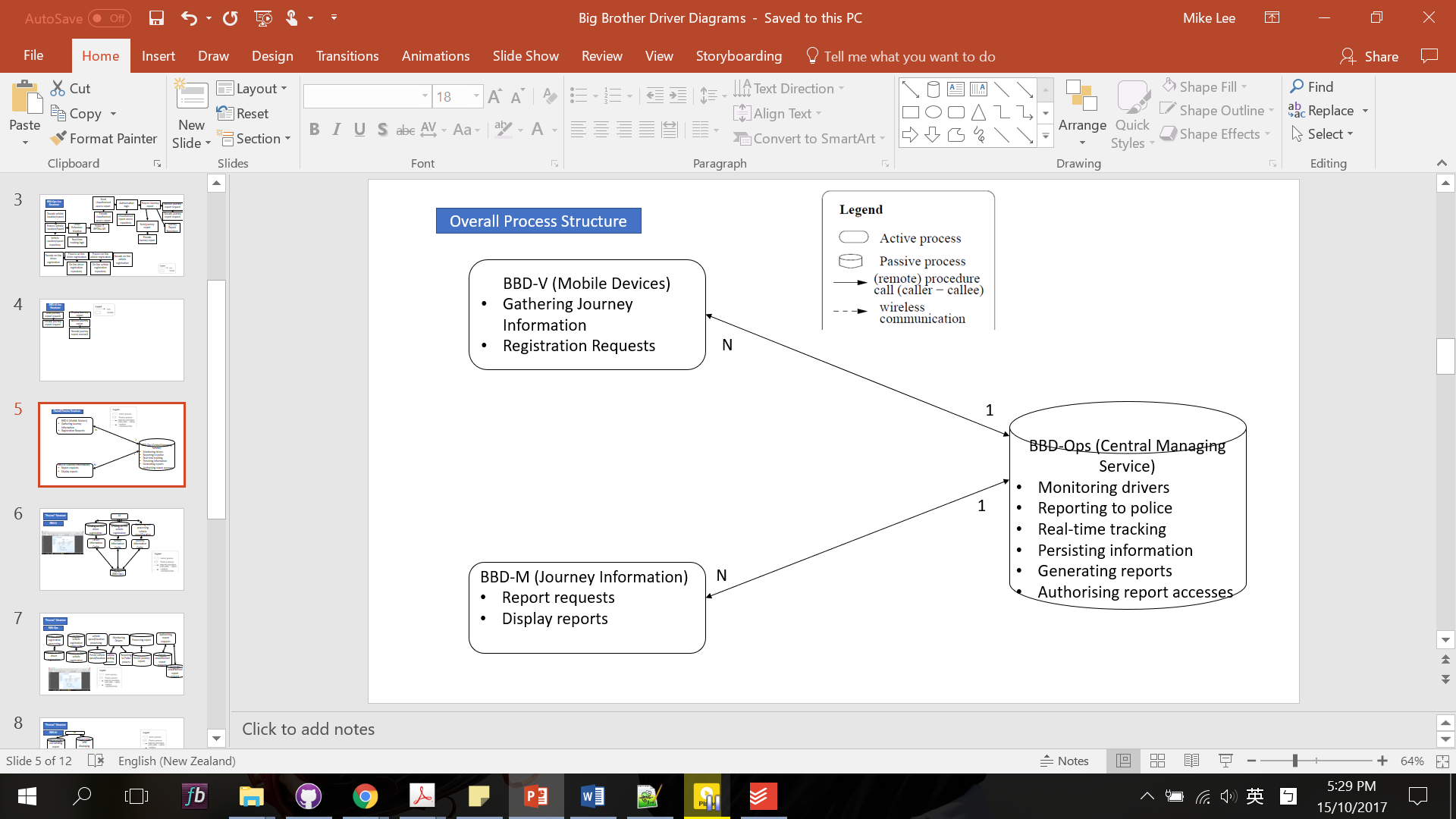
* The ‘Process Vehicle Location/Speed’ module uses the ‘decode vehicle location/speed’ module to unpack the speed/location data received. The ‘Process Vehicle Location/Speed’ module then stores the received and calculated location/speed data using the ‘vehicle Location/speed repository’ module.
* The ‘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 drivers to the police. The decisions are made based on factors such as current speed limits, unnecessary line changes, and unnecessary line-crossing.
* The ‘Real-time tracking logic’ module is used by the ‘Driver Behaviour Monitor’ module to allow real-time tracking for the police and parents for targeted drivers.
* The ‘Police IT RESTful API’ module is used by the ‘Driver Behaviour Monitor’ module to report erractical driving behaviour or excessive speed to the police immediately.
* The ‘Process on-line driver registration’ module processes the driver information received after unpacking the packet using the ‘Decode on-line driver registration’ module. The ‘Process on-line driver registration’ module then persists the driver information obtained using the ‘On-line driver registration repository’.
* The ‘Process on-line vehicle registration’ module processes the vehicle information received after unpacking the packet using the ‘Decode on-line vehicle registration’ module. The ‘Process on-line vehicle registration’ module then persists the vehicle information obtained using the ‘On-line vehicle registration repository’.
* The ‘Process Journey report’ module uses the ‘Journey Report Repository’ to persist the reports generated.
* The ‘Process Journey report’ module receives the ‘Receive journey report request’ module to receive journey report requests. The ‘Receive journey report request’ module uses the ‘Decode journey report request’ to unpack the request packet received.
* The ‘Process Journey report’ module uses the ‘Send journey report’ module and the ‘Encode Journey report’ module to package the report information and send it as a packet.
* The ‘Authorisation logic’ module is used by the ‘Process Journey report’ module to authorise journey report accesses.
* Unauthorised journey report accesses are persisted using the ‘Unauthorised report access repository’ module.
* The ‘Send Unauthorised access report’ and ‘Encode Unauthorised access report’ modules are used to package and send the unauthorised access to journey reports.

## BBD-M

## 

* The ‘**Display journey Report’** module receives and unpackages the report packets and displays them. It uses the **‘receive journey report’ and the ‘decode journey’ modules.**
* The ‘Send journey report request’ and ‘Encode journey report request’ modules are used to package and send journey report requests.

### Component and Connector structures



## Overall:

There are two different types of components:

“active” processes and “passive” processes.

Active processes run independently from other components and can initiate communication similarly to a thread.

On the other hand, passive processes cannot initiate communication and can be regarded as “servers.

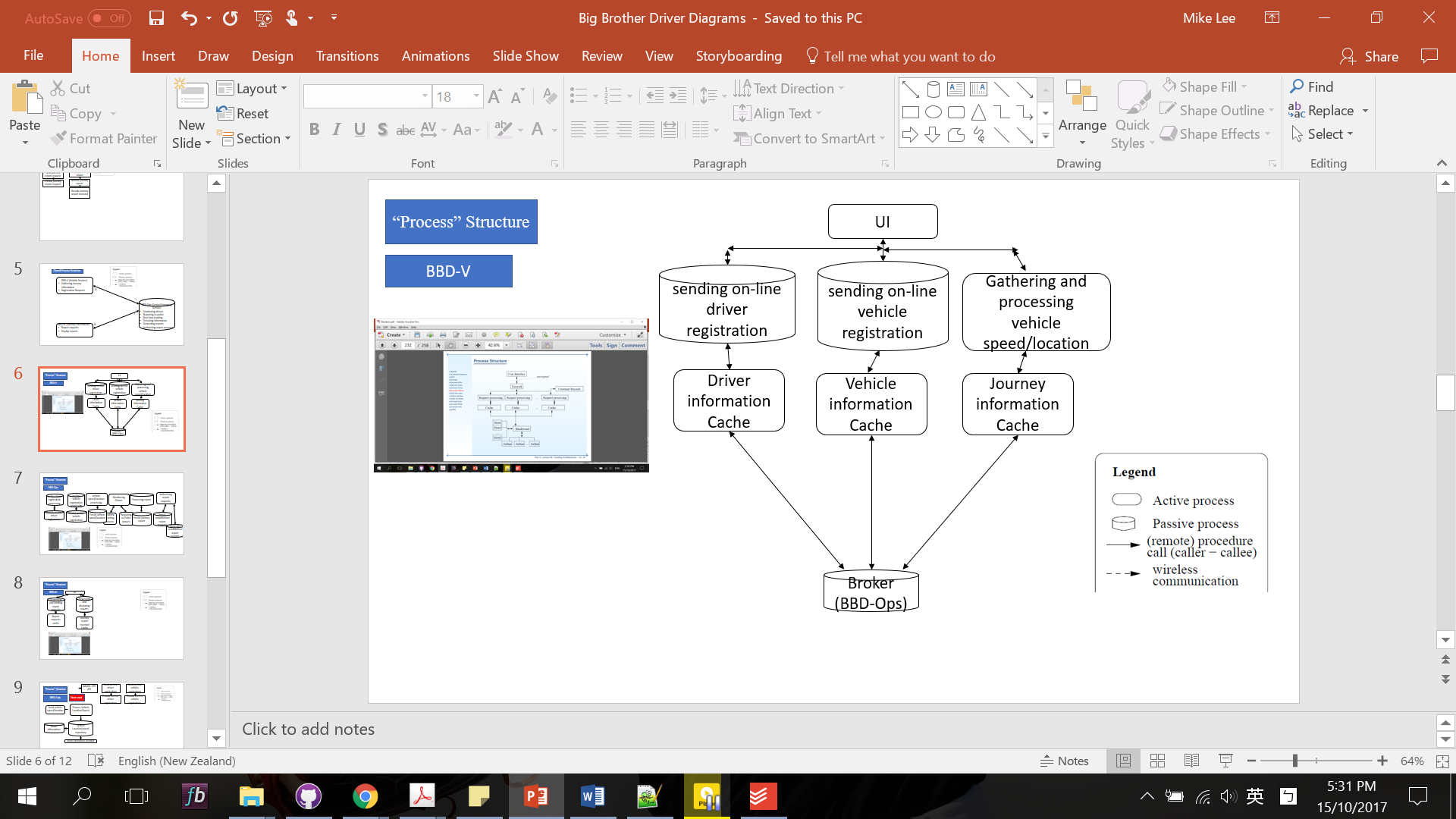
There are multiple BBD-V and BBD-M clients communicating with the BBD-Ops.

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.

## BBD-V



There are only a single UI process communicating with the following processes: sending on-line driver registration, sending on-line vehicle registration, Gathering and processing vehicle speed/location. There are multiple instances of these processes communicating with the UI process. There are multiple caches used to store relevant information such as cookies. The registration requests and speed/location information are then sent to the broker, which is the BBS-Ops in this context.

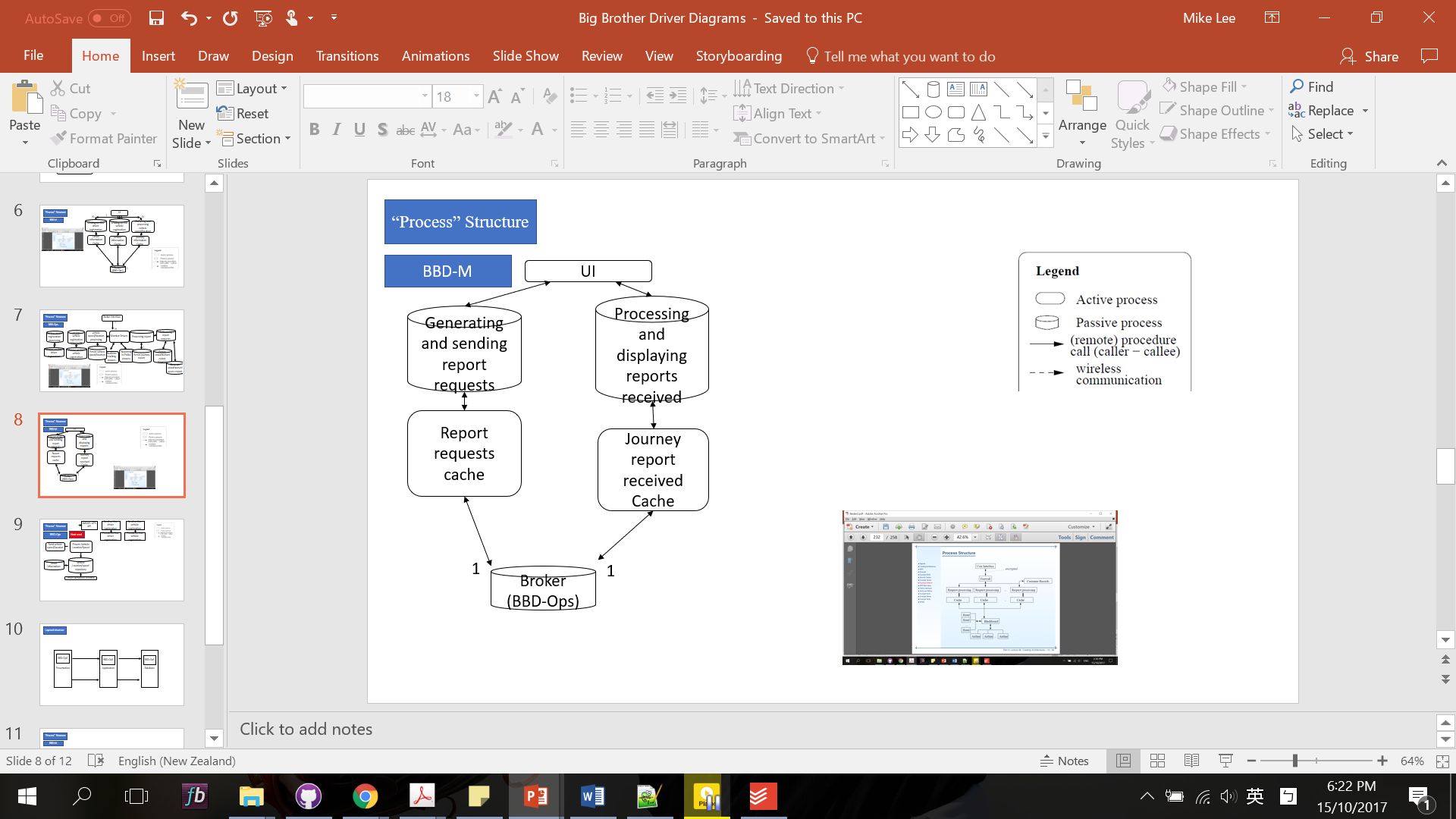
## BBD-Ops

The BBD-Ops is a ‘broker’ and contains a single Broker interface. External clients interact with this broker interface.

There are multiple processes running. These processes include ‘on-line driver registration processing’, ‘on-line vehicle registration processing’, ‘vehicle speed/location processing’, ‘Monitor Drivers’, ‘Processing report’, and ‘Authorising report requests’. There can be multiple processes for the reporting, tracking, and generation of the unauthorised access reports.

All communication links are synchronous.

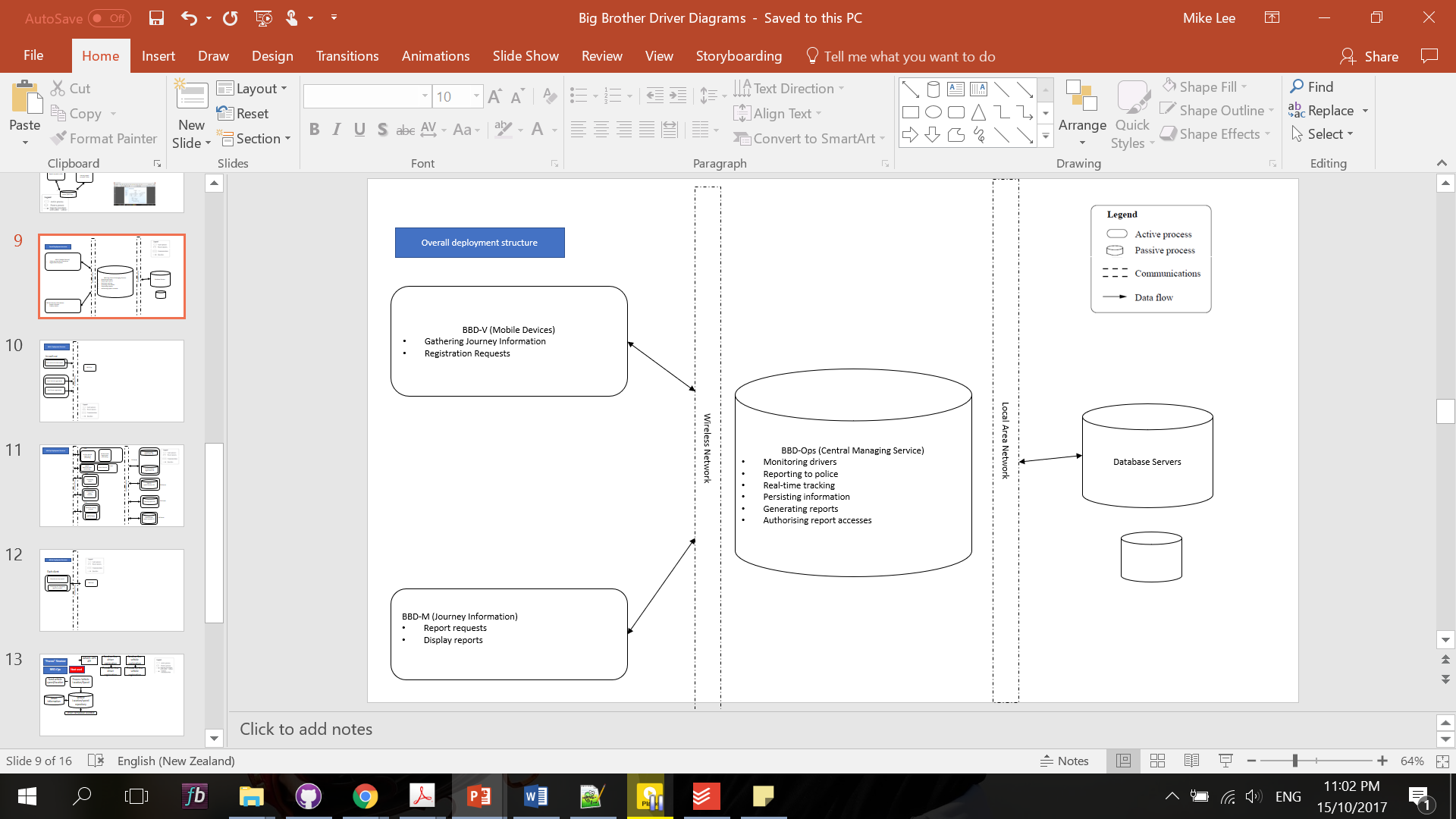
## BBD-M



BBD-M has a single UI process, ‘Generating and sending report requests’, and ‘Processing and displaying reports received’ processes. The UI process communicates with multiple report request and report received processes. The information related to the processes is cached. These processes then communicate with the single broker process to send or receive data.

### Allocation Structures

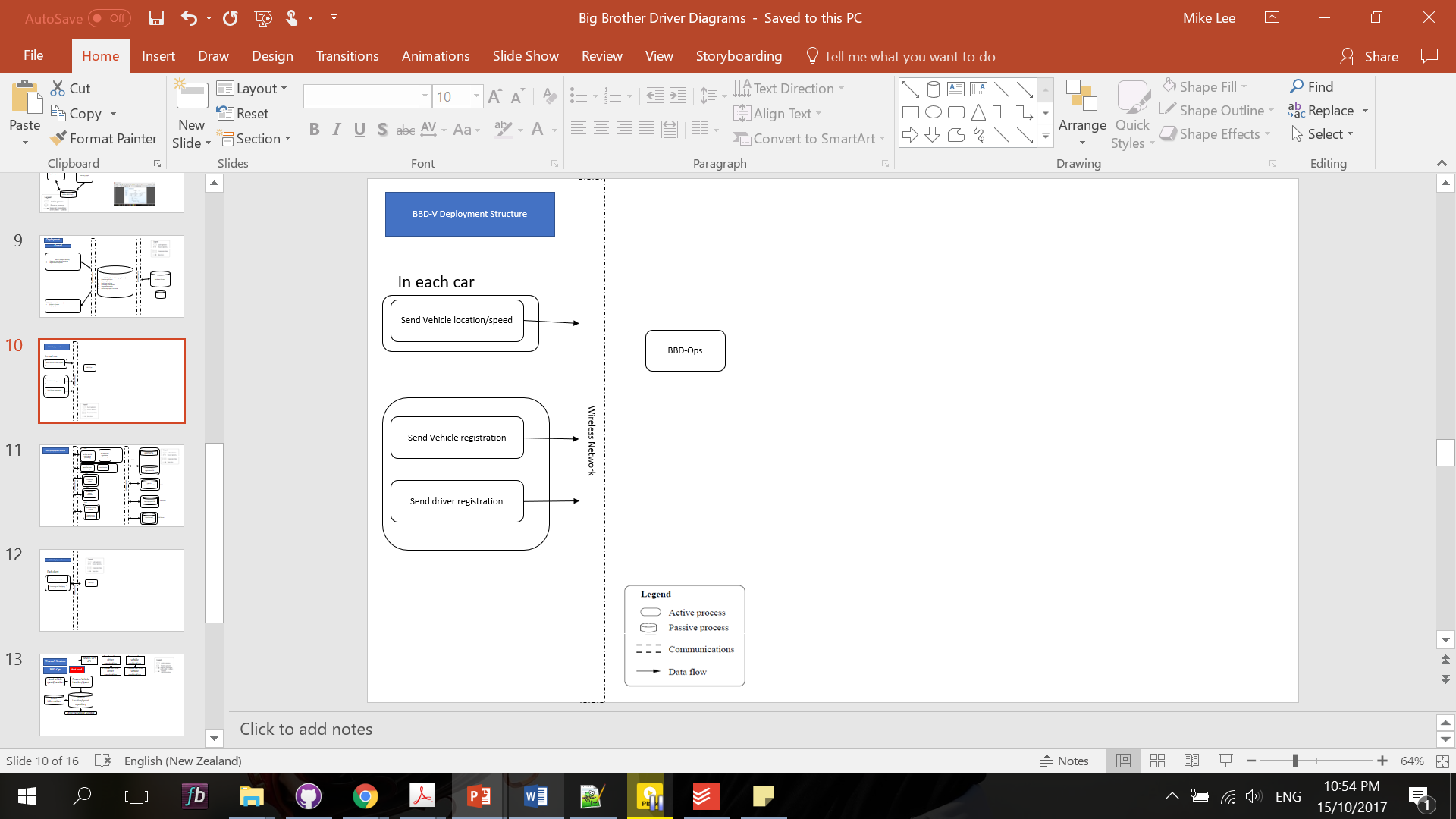
## overall



The BBD-V and BBD-M communicate with the BBD-Ops using wireless communication.

The BBD-Ops communicate to the database servers using LANs.

## BBD-V

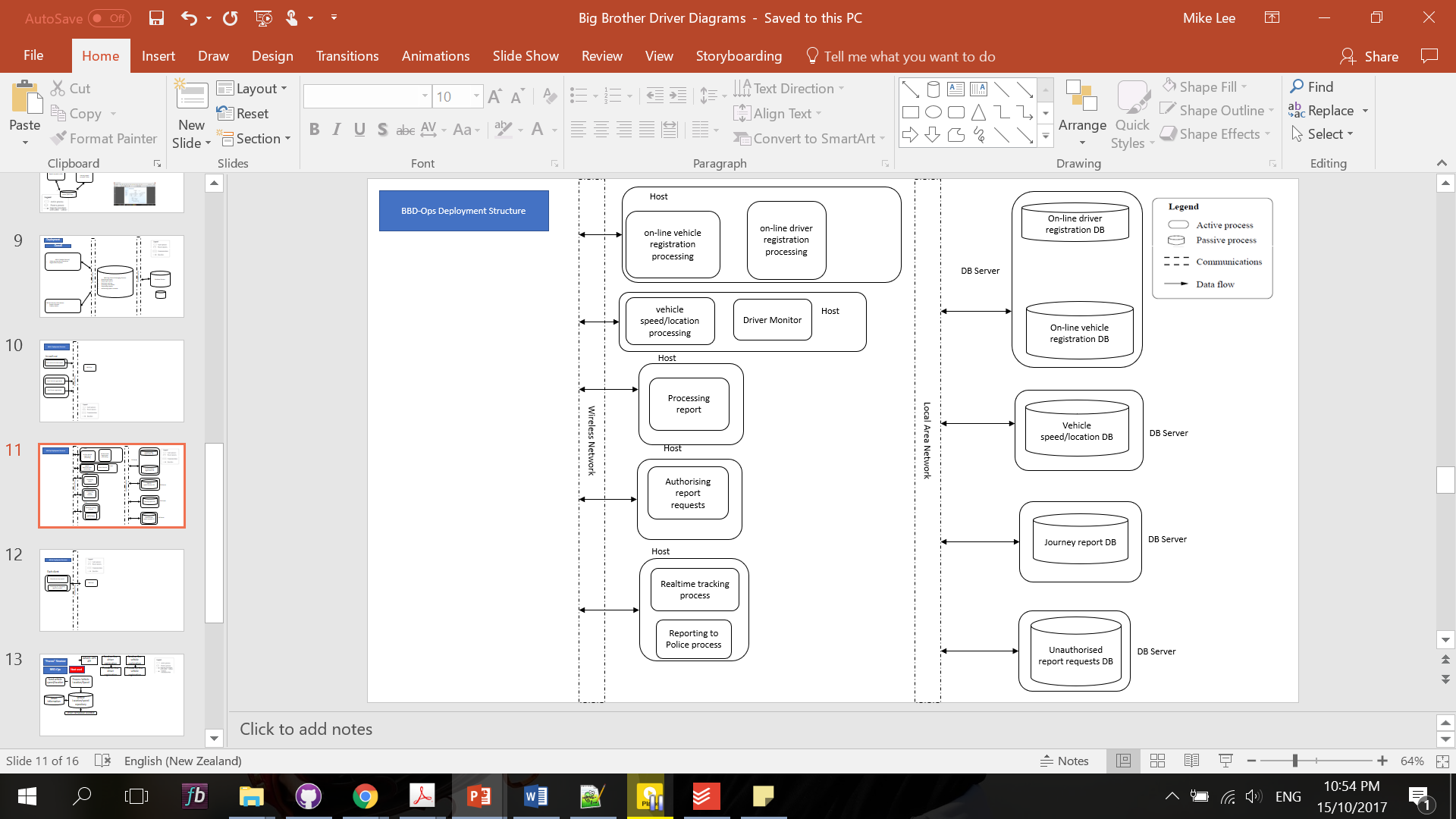


In each car, at least one ‘send speed/location’ process runs on a mobile device.

‘Send driver and vehicle registration’ processes can be run on multiple devices.

These processes communicate to BBD-Ops via wireless mobile network.

## BBD-Ops



BBD-Ops receives data and requests via wireless mobile network.

The various processes are hosted under separate hosts.

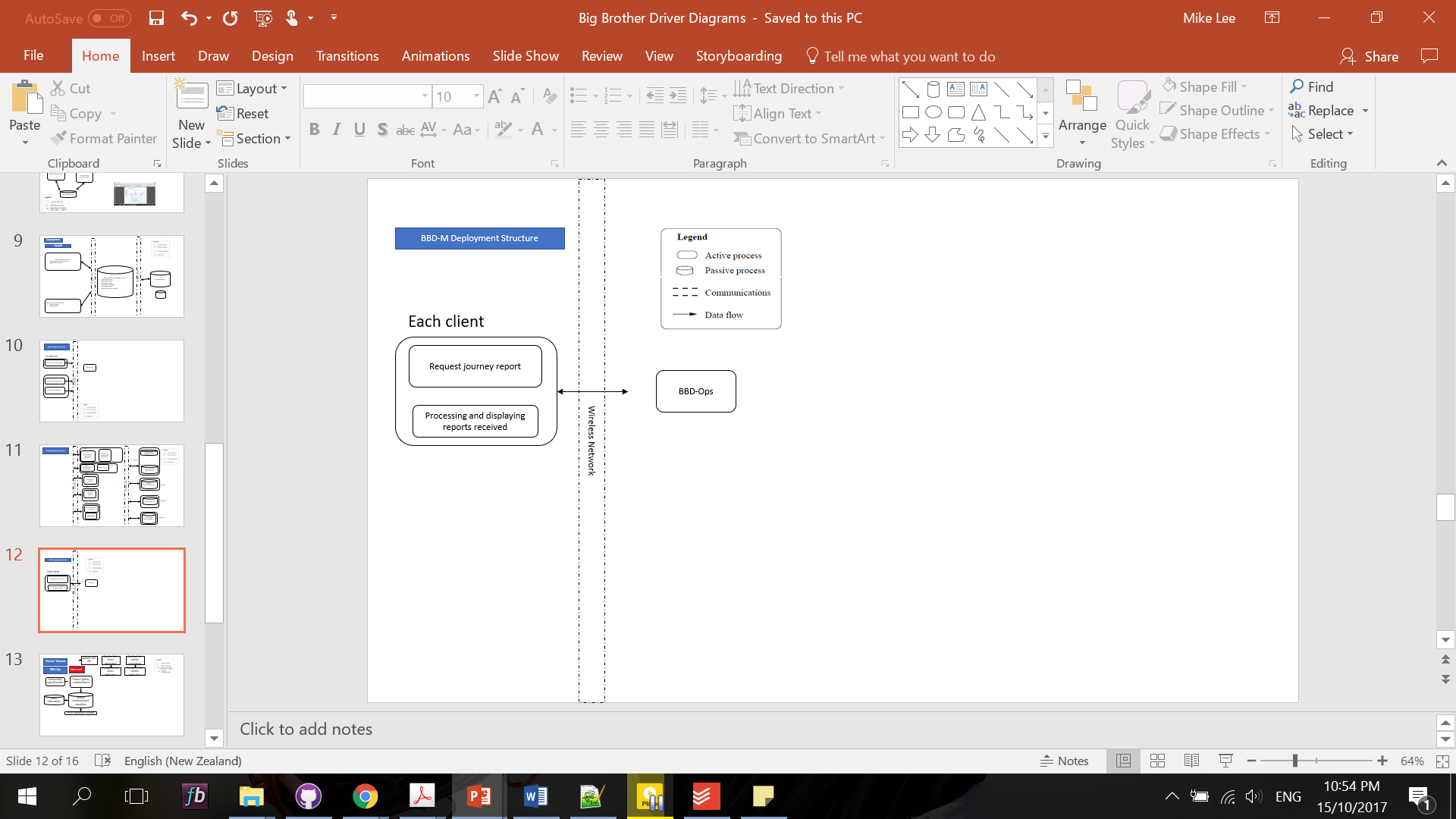
The registration request processes are co-located on one host.

The ‘vehicle speed/location processing’ and driver monitor processes are co-located. This helps to improve latency. Similar strategy is applied to the ‘Realtime tracking process’ and ‘reporting to police’ processes.

BBD-Ops then use Local Area Network to communicate with the database servers. The servers include the following: registration DB, Vehicle speed/location DB, Journey report DB, Unauthorised report requests DB.

## BBD-M

The ‘Request journey report’ and ‘Processing and displaying reports received’ processes run on each mobile client. They communicate wirelessly with the BBD-Ops.



### Patterns

# Client-Server / N-Tier Systems

* The overall software architecture of this system is a Client-Server architecture combined with the Layered architecture.
* The system consists of an application server, a large number of clients, and a database.
* The first tier is a presentation tier which deals with the interaction with the user. The presentation tier is also known as the client. There can be a large of number of clients which can all access the server at the same time.
* The clients here are thin clients. This means they do not contain a lot of application code. The clients process user input, send requests to the server, and show the results of these requests to the user.
* The second tier is an application tier which processes the requests of all clients. It is the actual web application that performs functionalities specific to the BBD application. It does not store the persistent data itself and contacts the database server whenever it needs data instead.
* The application tier in this system is designed to be stateless. This allows server duplication for scalability.
* The third **database tier** contains the database management system that manages all persistent data.

### Tactics

* tactic: caching (client)
* load balancing
* BBV-Ops is made stateless, to enable server replication.
* States are maintained by the use of cookies.
* broker (single module)
* **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** Multiple copies of the **Estimator** process are used,
* one per bus. It is also possible to duplicate the route information or history information.

### 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.