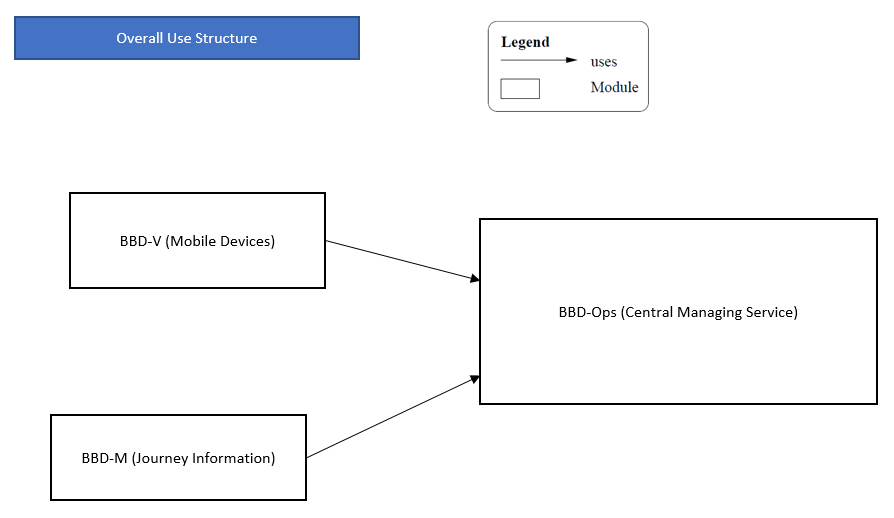
SE325 Assignment 2 Report

840454023, elee353

### Architecture Section

### Module structures

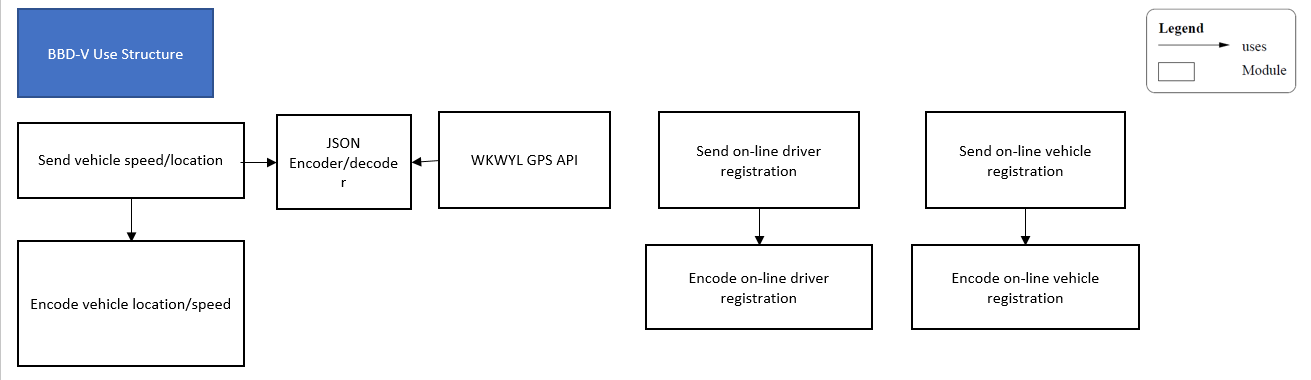
## Overall:



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.

## BBD-V

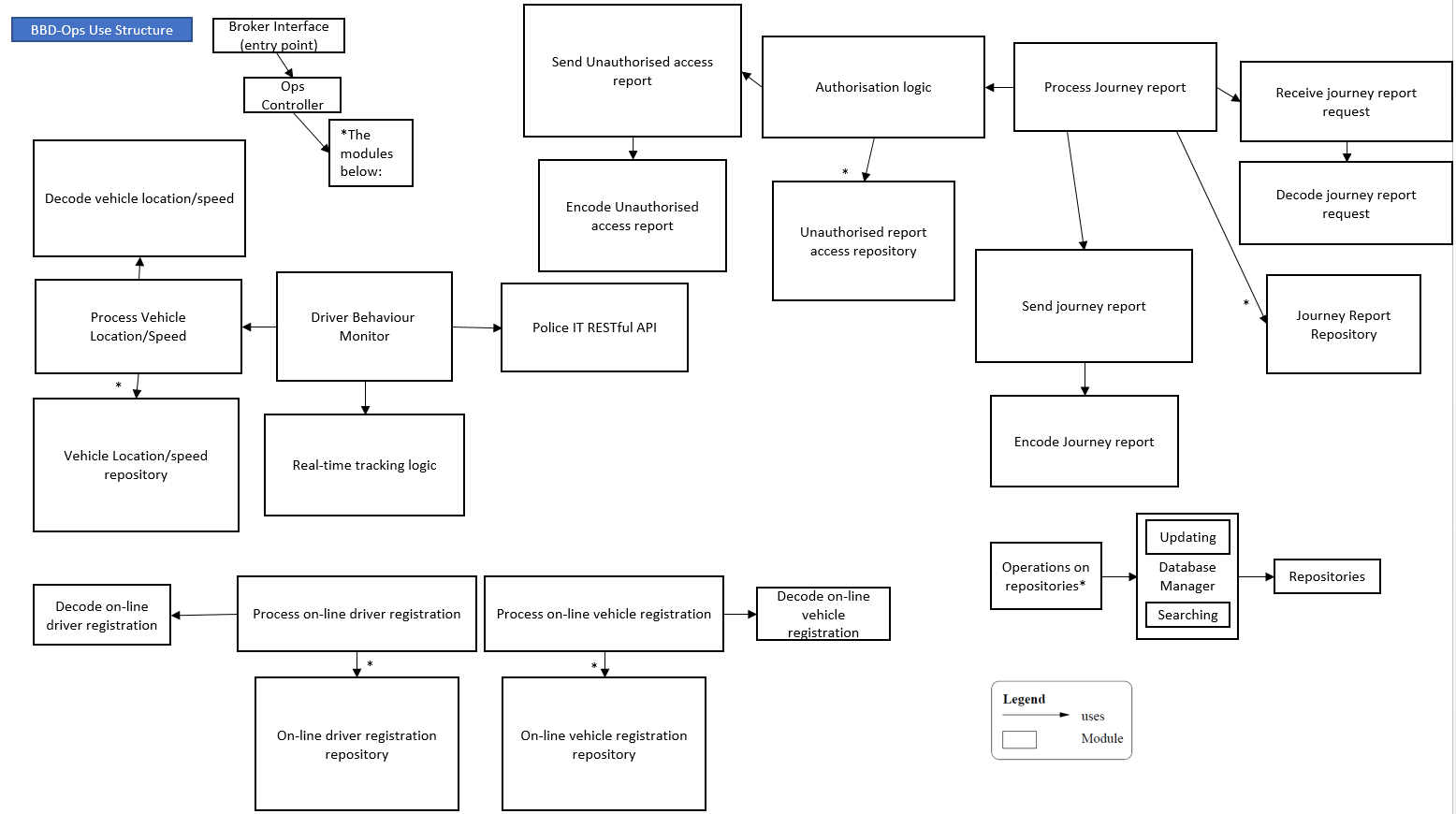


The ‘send vehicle speed/location’ module uses the ‘JSON encoder’ and the WKWYL GPS API to receive real-time details about the speed limit and road-layout for the specified GPS location. It then uses the API included in OS to send the packaged data. The ‘send vehicle speed/location’ module also uses the ‘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 BBD-Ops contains a broker interface to be used by external clients. Requests or data passed in are then filtered by the Ops controller to decide appropriate modules to be used.

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.

It is worth noting that it is implied that all repository operations are through a database manager. It contains submodules for handling operations such as searching and updating efficiently.

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 uses 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 accessor reports.

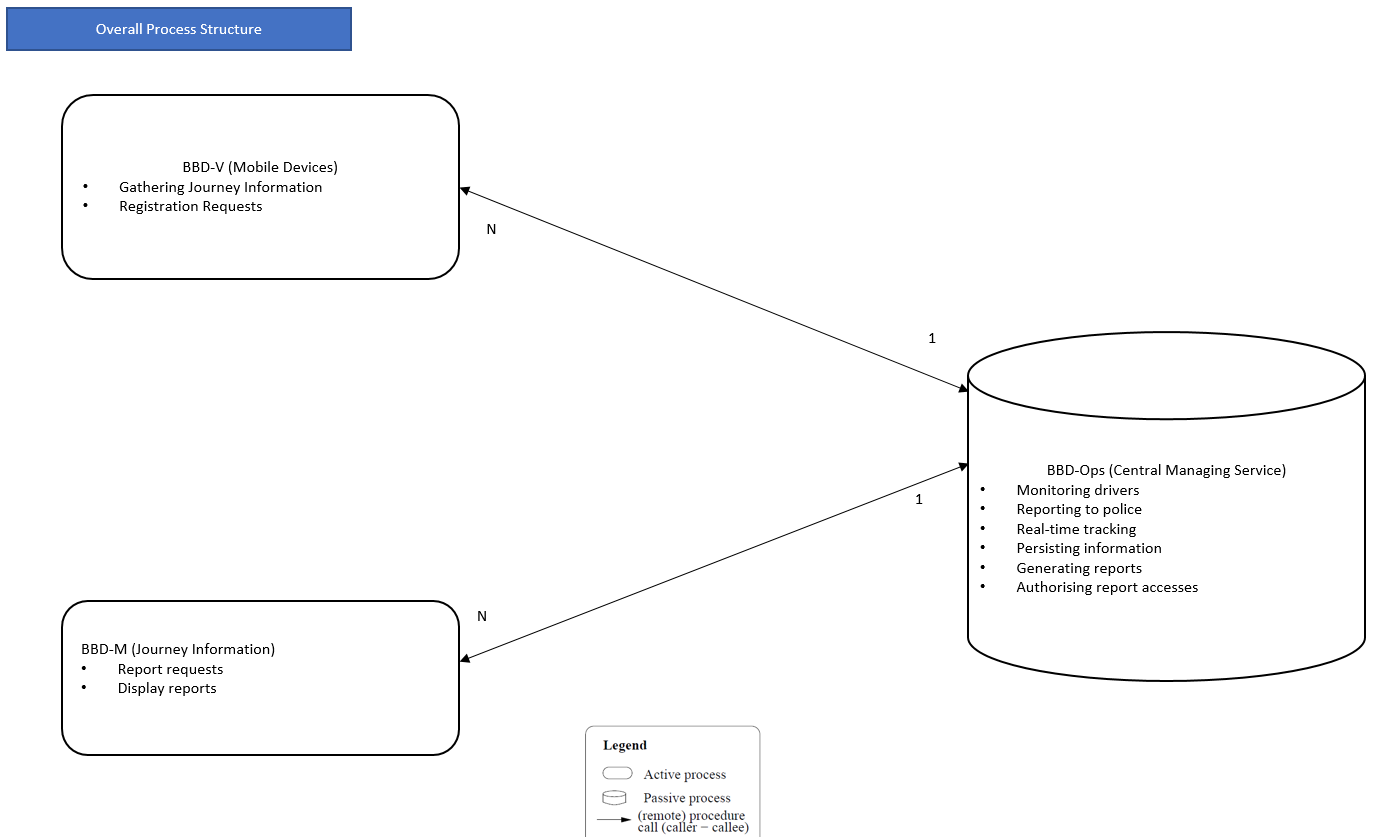
## BBD-M

## 

* 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 to receive and unpackage packets.
* 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:



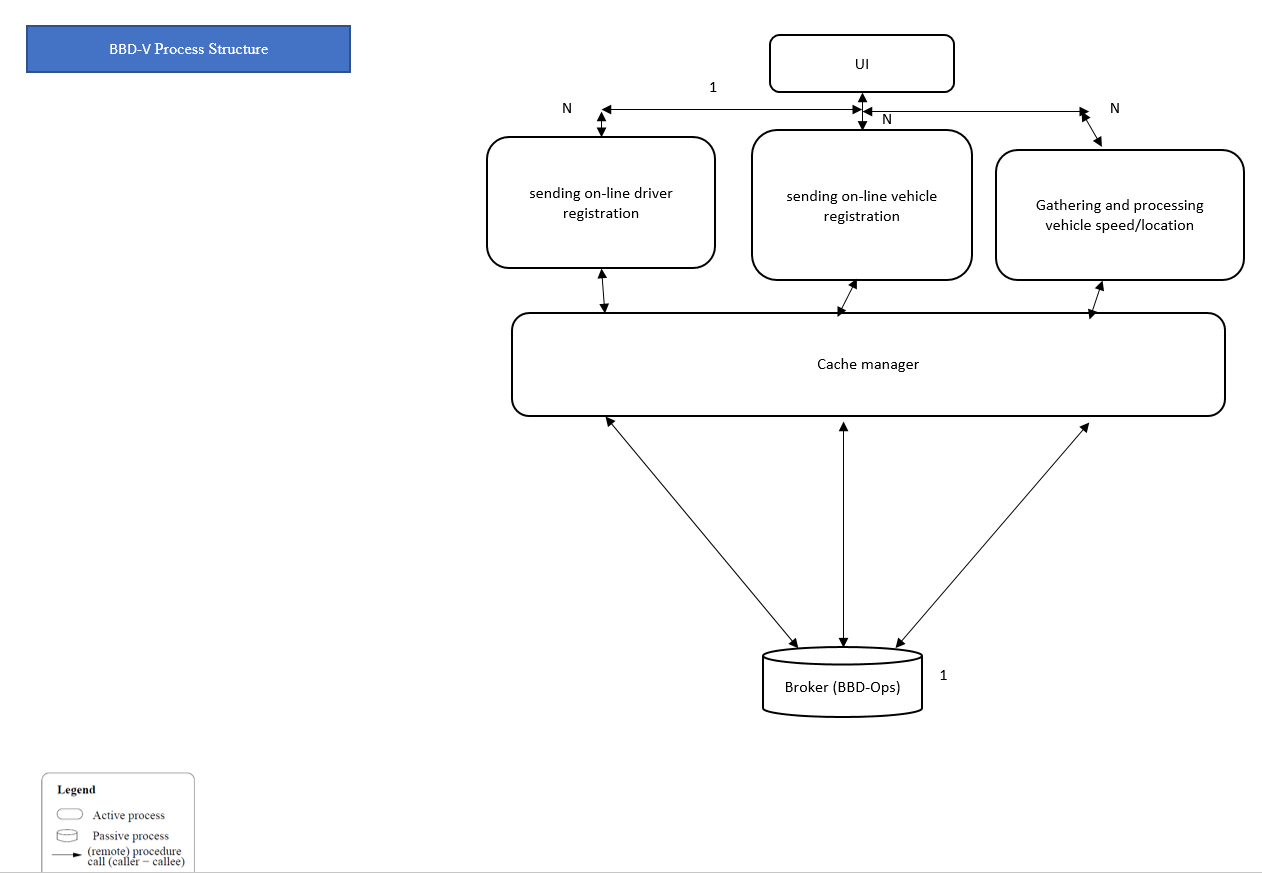
* There are two different types of components: “active” processes and “passive” processes.
* Active processes run independently from other components and can initiate communication similarly to a thread.
* On the other hand, passive processes cannot initiate communication and are regarded as “servers”.
* There are multiple BBD-V and BBD-M clients communicating with the BBD-Ops.
* The BBD-Ops can be duplicated and scaled to handle increased workload.

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.

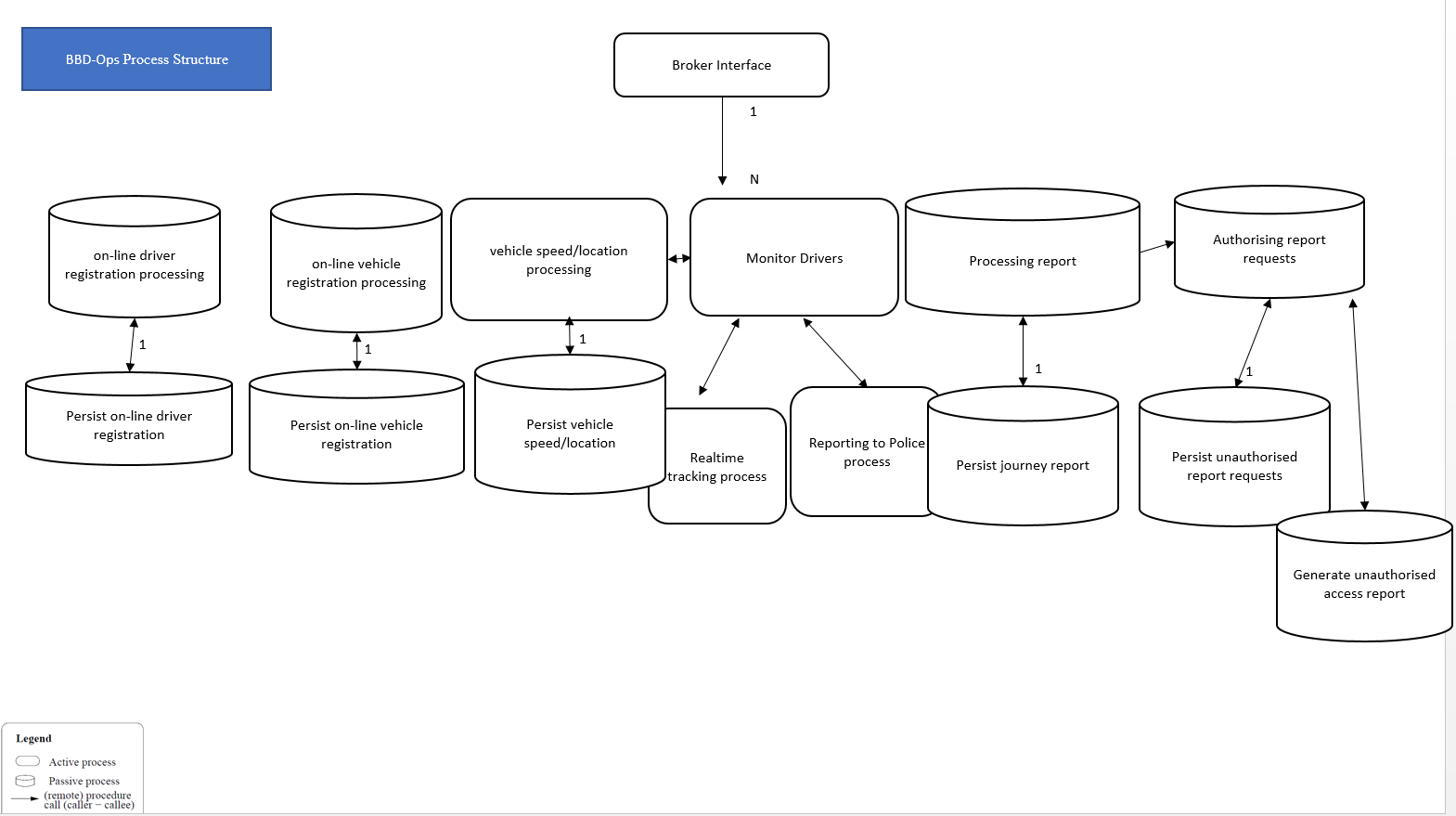
All wireless communication is asynchronous and all other communication is synchronous.

## 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 caching operations are handles by the cache manager. The registration requests and speed/location information are then sent to the broker, which is the BBS-Ops in this context.

## BBD-Ops

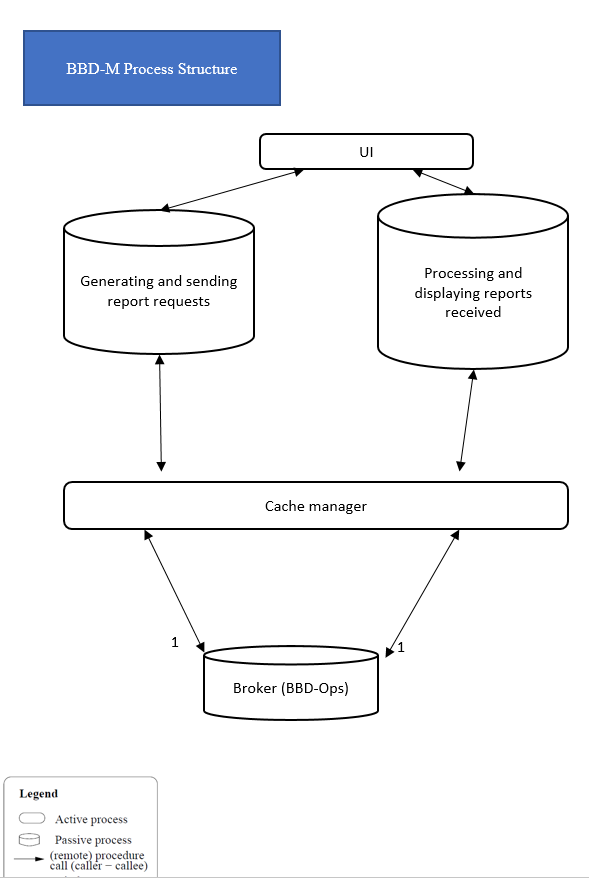


The BBD-Ops is a ‘broker’ and contains a single Broker interface. External clients interact with this broker interface.

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.

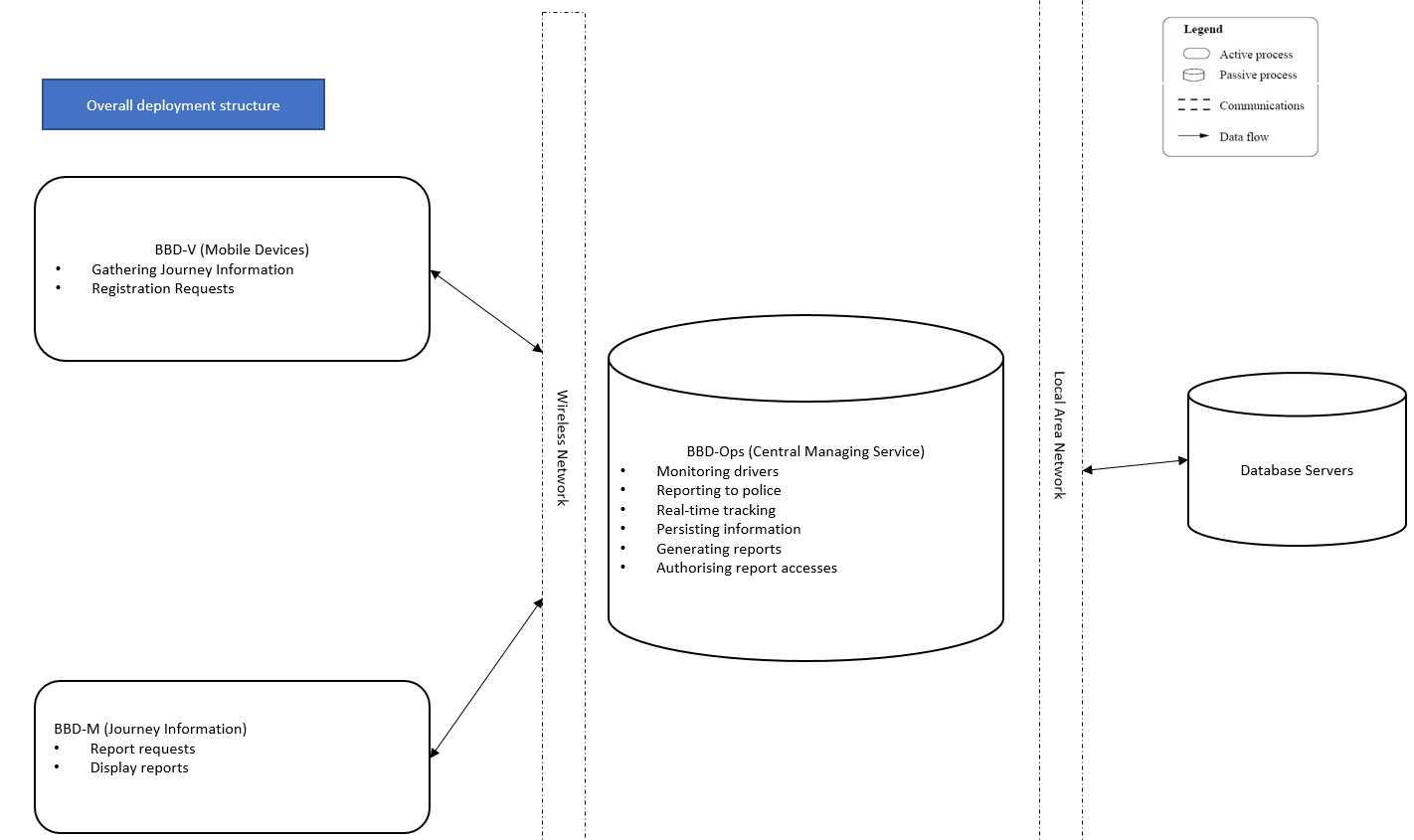
## BBD-M



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

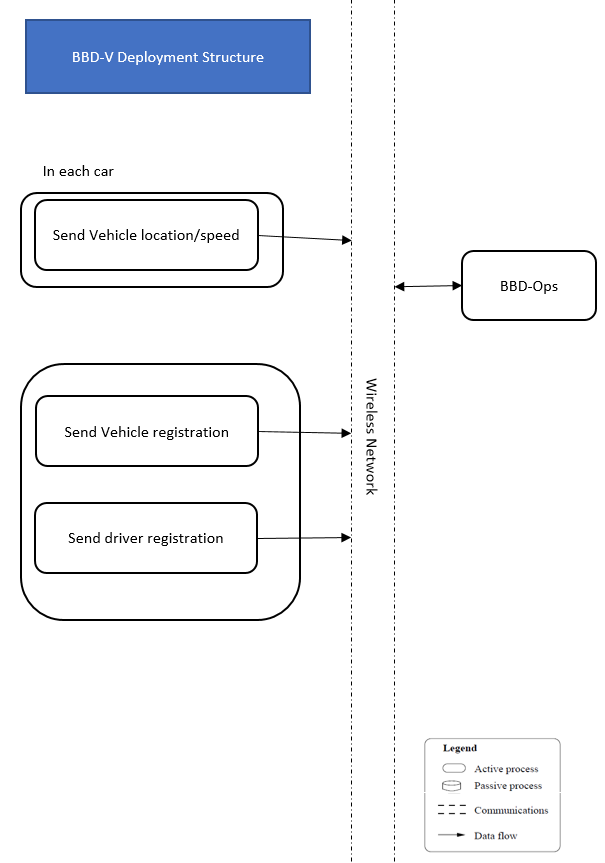
## Overall



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

The BBD-Ops communicate to the database servers using Local Area Networks.

## BBD-V

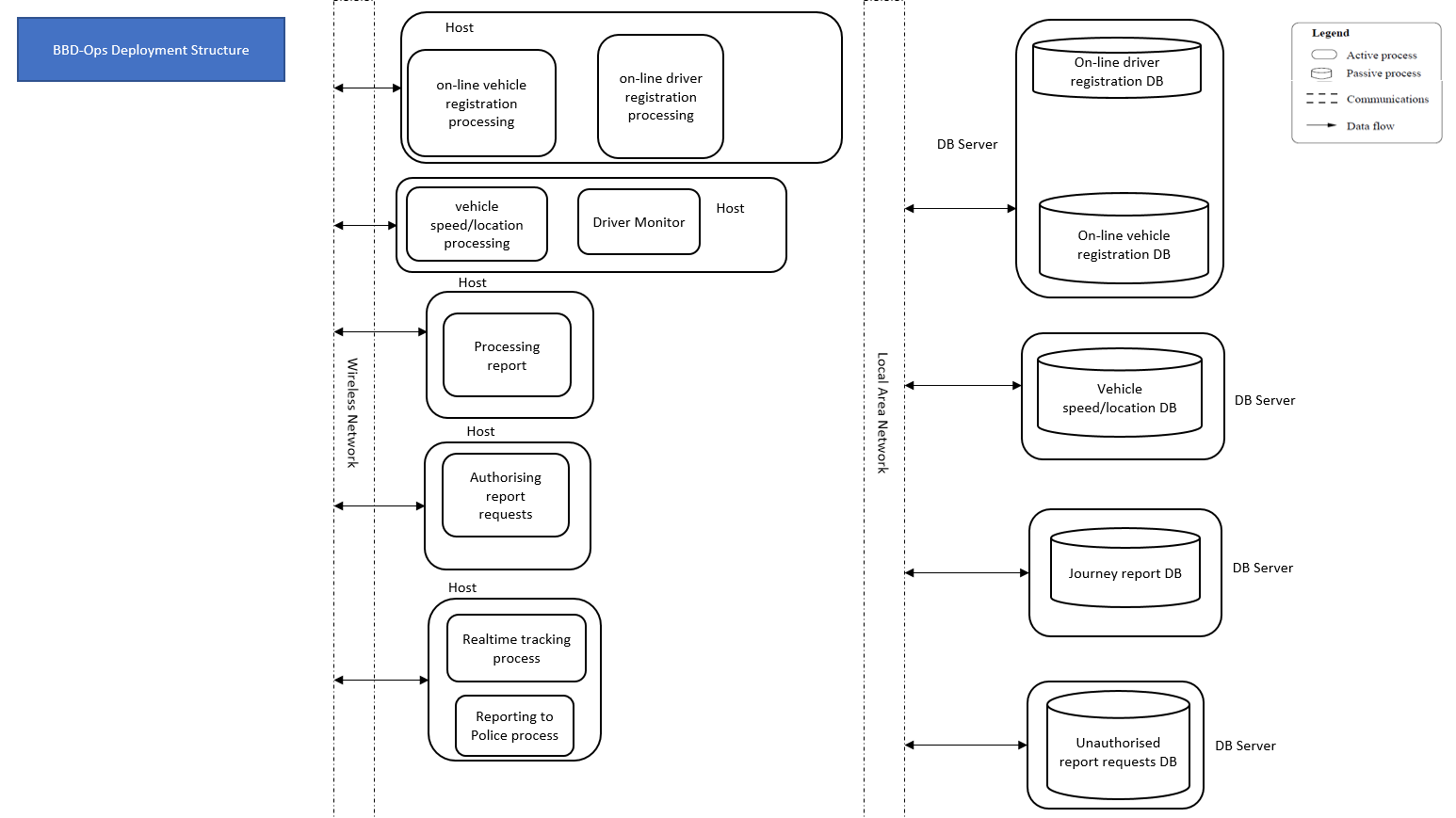


In each car, at least one ‘send speed/location’ process runs on a mobile device.

‘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.

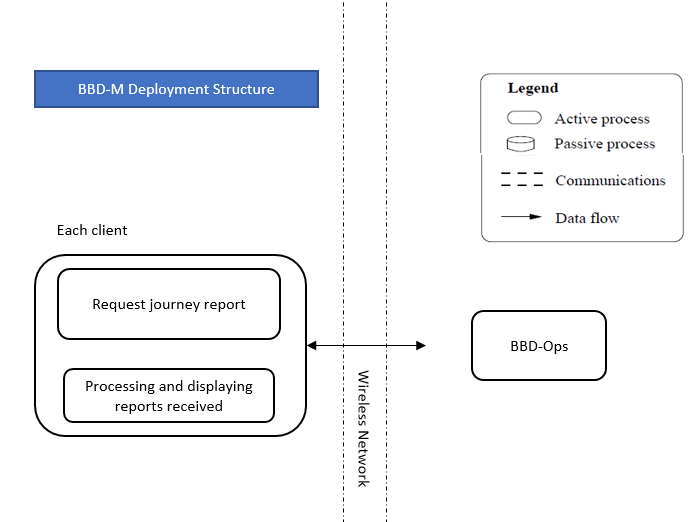
The various processes are hosted under separate hosts.

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 ‘Real-time tracking’ 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 to retrieve journey reports.

## 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.

## Pipe-and-Filter

* A lot of the data is transformed serially in the system, for example, the results processed by the ‘vehicle speed/location process’ is used by the driver monitor process.

systems in which data is transformed serially

* supporting functional composition data analysis

### 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**
* **maintain multiple copies of either data or computations**
* Multiple copies of the ‘vehicle speed/location’ and ‘Driver Monitor’ process are used
* It is also possible to duplicate the route information or history information.

**Tactics**

* Tactics for Performance:
* Increase computational efficiency:
* using an efficient algorithm to …
* Reduce computational overhead:
* Co-locate some processes results in reduced overhead of the remote communication. Less communication cost means that the data can be transferred faster to meet the performance requirements.
* Manage event rate:
* deciding on how frequently to update journey information from BBD-V to BBD-Ops for the drivers that need to be tracked in real-time.
* Control frequency of sampling:
* choice to be made on how often the data should be recorded. More frequent means more accurate data but it will lead to high cost and slower performance.
* Introduce concurrency:
* having multiple servers to process incoming data from many drivers. This will allow many journey information from multiple BBD-V drivers to be processed and get it ready for BBD-M at the same time within the required limit.
* Increase available resource: having many processors is essential but having faster processors is also essential as well. It may cost more but it must be done to meet the performance requirements.
* Maintain multiple copies of either data or computations: have multiple copies of data and servers. By placing multiple copies of data and server around country, the computation will be done faster and more effectively as the distance from the mobile device and the server decreases. It also ensures the security of data loss or corruption due to many reasons (i.e. natural disaster).
* Tactic for modifiability:
* Split module:
* the hardware (displays and user interface) and the computations are well split into separate module which minimises the needs for changing computations when the hardware changes. So, when new devices are released, only the user interface should require adjustment which can be done quite quickly, i.e. can be done within 40 hours of development and testing.
* Tactic for security:
* Verify message integrity: confirming that data has not been changed or corrupted to deliver correct information to the related users.
* Authenticate actors: authenticate the actor on the mobile device to be the authorised person by making the user trying to access BBD-M to enter a password that matches.
* Limit access: Access to the journey information is limited by the BBD-Ops who distribute the information to specific BBD-M users. This reduces probability of sending the information to un-authorised people most of the time i.e. BBD-M user does not have access to all the information on the database in BBD-Ops.
* Encrypt data: Make sure the data being sent is encrypted which can only be decrypted by special token or key which is given to authorised users of BBD-M. The BBD-M user with no right would not be able to decrypt data and view it.
* Lock computer: the app is being used on personal mobile devices that will be locked and prevent unauthorised people from using the device and the app.

## Scenarios

### Scenario 1 & Scenario 2

* Assume send once every second (worst case scenario): 512 bytes (maximum size) \* 30 Journeys per month \* 60 seconds \* 20 minutes = 18432000 bytes = 18.4 MB
* For the data to be sent to BBD-Ops, 4 bytes each will be used on latitude, longitude (GPS co-ordinates) and vehicle ID. 1 byte each will be used on speed and acceleration. Therefore, a total of 14 bytes is used.
* In reality, the data packets sent to BBD-Ops are likely to be: 14 bytes \* 30 Journeys per month \* 60 seconds \* 20 minutes = 504000 bytes = 0.504 MB
* A checksum can be appended (4 bytes) if necessary. Further encryption can also be added
* 5 Mib / 512 bytes gives 10240 times to send messages.
* This should be affordable for most mobile phone users with average data plans.
* By reducing sampling rate at certain conditions, such as when the vehicle is waiting at a red light, this data usage can be reduced.
* local area network speed (100Mbps).
* We can use the map information retrieved from the GPS data received to detect if the vehicle is stopping at red lights.

we can reduce either the send rate from the vehicle (manage event rate) or ignore many of the data coming in a relatively short amount of time (control frequency of sampling).

* We can have separate networks (**increase available resources**) and have multiple copies of some of the information, such as the driver and map information (**maintain multiple copies of either data or computations**).
* Caching (**maintain multiple copies of either data or computations**) is used to reduce the network contention.
* There is some overhead involved in grouping the speed/location information and then packaging them up in

messages to be sent to each BBD-Ops.

* Constructing the strings from the speed/location information can take a considerable amount of time for each client.
* The time taken depends on the string concatenation implementation used, but careful implementation when constructing these JSON objects (**increase computational efficiency**) will improve performance.

reduce number of events to process

◦ manage event rate

◦ control frequency of sampling

Performance

reduce resources required for processing stimuli

◦ increase computational efficiency

◦ reduce computational overhead

• reduce number of events to process

◦ manage event rate

◦ control frequency of sampling

• control resources consumed

◦ bound execution times

◦ bound queue sizes

Increase available resources — use multiple communications links, use more processors

introduce concurrency — have multiple servers processing requests

• maintain multiple copies of either data or computations — have multiple

copies of the data about flights, one per server

•If the **repositories** are indexed by driver identifier or vehicle identifier and uses an efficient data structure for searching on the identifier, such as a hash table (choose a more efficient algorithm, increase computational efficiency), then updating the repository with the new locations and speeds should exhibit logarithmic (O(log(N))) performance.

The Send vehicle speed/location process get speed and location information relevant to the vehicle. A reasonable organisation of the data (**increase computational efficiency**) should allow this to be done within seconds. The location or map information can be cached and pre-processed beforehand.

This off-line pre-processing reduces the computation needed and amount of data required (**increase computational efficiency**).

The actual time taken by the driver monitor and process Vehicle Location/Speed algorithms will depend on the speed of the machine **they** running on, as well as efficiency of the algorithms. We can improve the hardware (**increase available resources** and **introduce concurrency**), and may be able to optomise the algorithm (**increase computational efficiency**).

* We can use the map information retrieved from the GPS data received to detect if the vehicle is stopping at red lights.

we can reduce either the send rate from the vehicle (manage event rate) or ignore many of the data coming in a relatively short amount of time (control frequency of sampling).

### Scenario 3

24

• coupling — reduce the probability that a change in one module will

impact another module

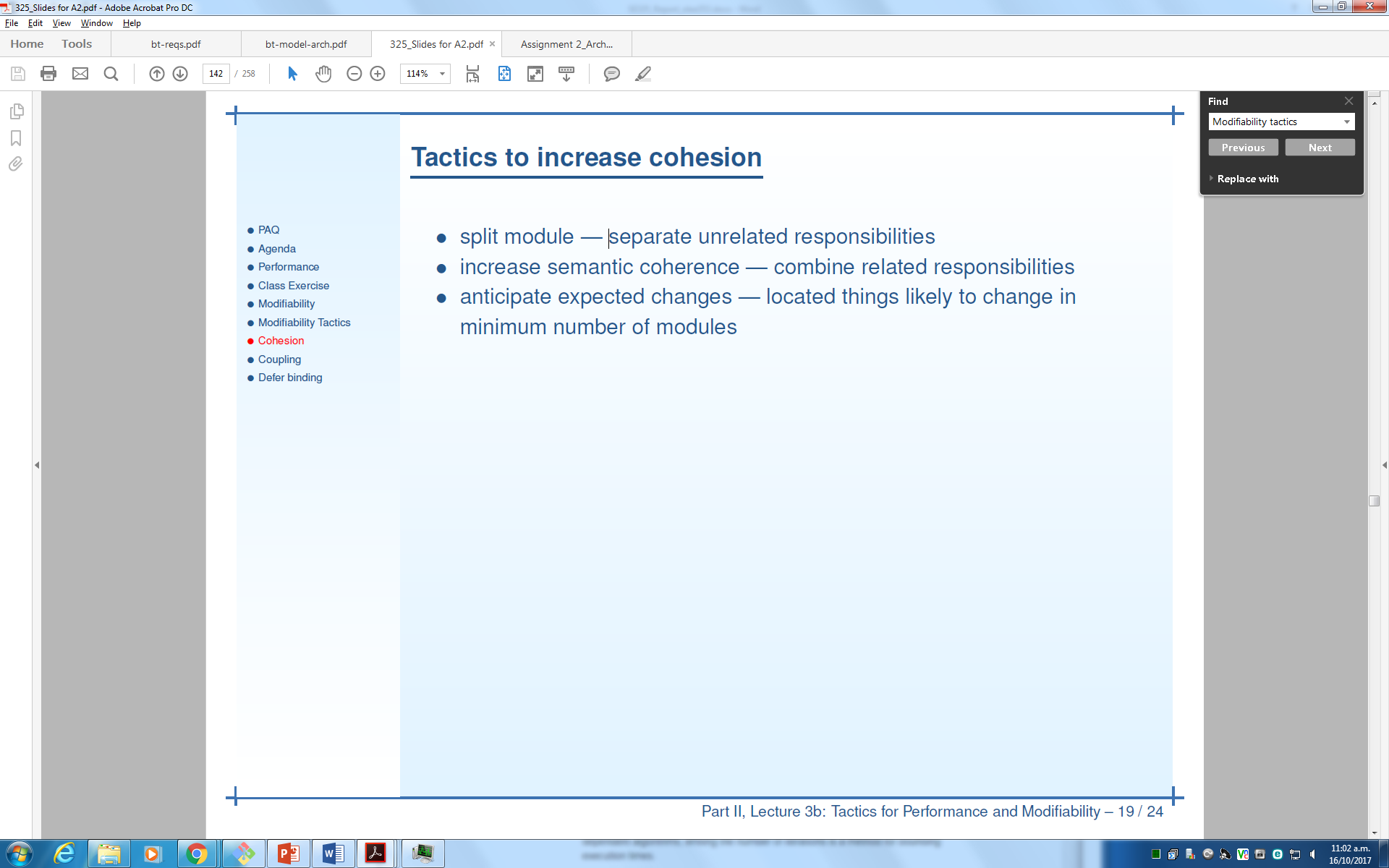
Reduced coupling by separating the BBD-V and BBD-Ops. The BBD-V interacts with BBD-Ops via it Broker interface. This increases semantic coherence and encapsulates BBD-V internal implementations.

• cohesion — maximise the probability that if some part of a module

changes, it all changes

• defer binding time — increase the ability of the computer to manage the

change (binding: determining what a name means)



### 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.