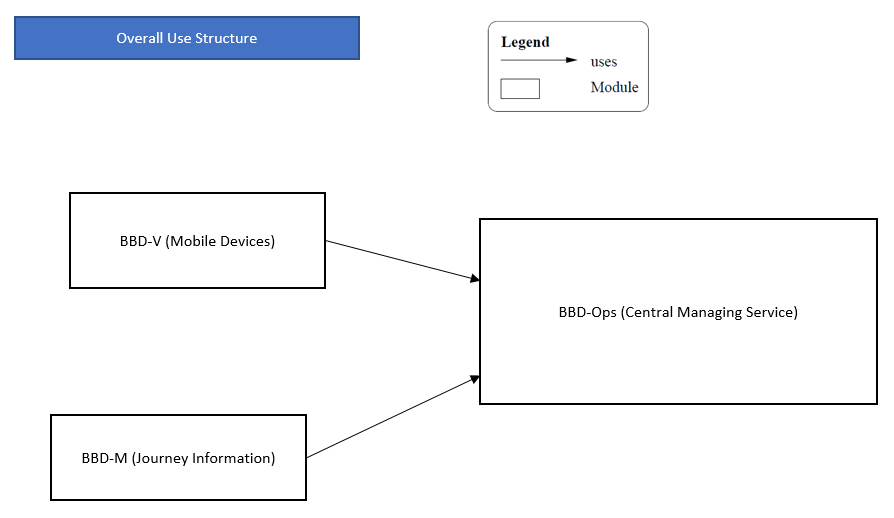
SE325 Assignment 2 Report

840454023, elee353

## Architecture Section

## Module structures

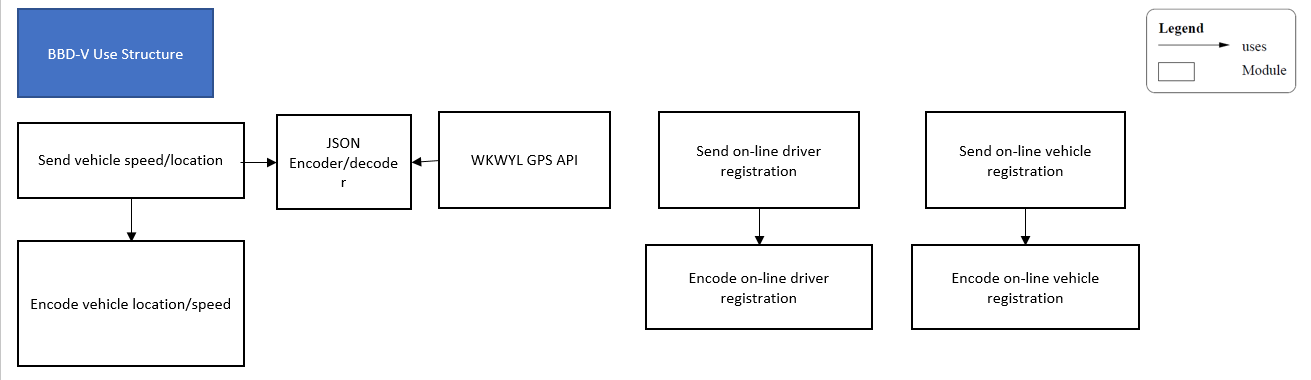
## Overall:



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

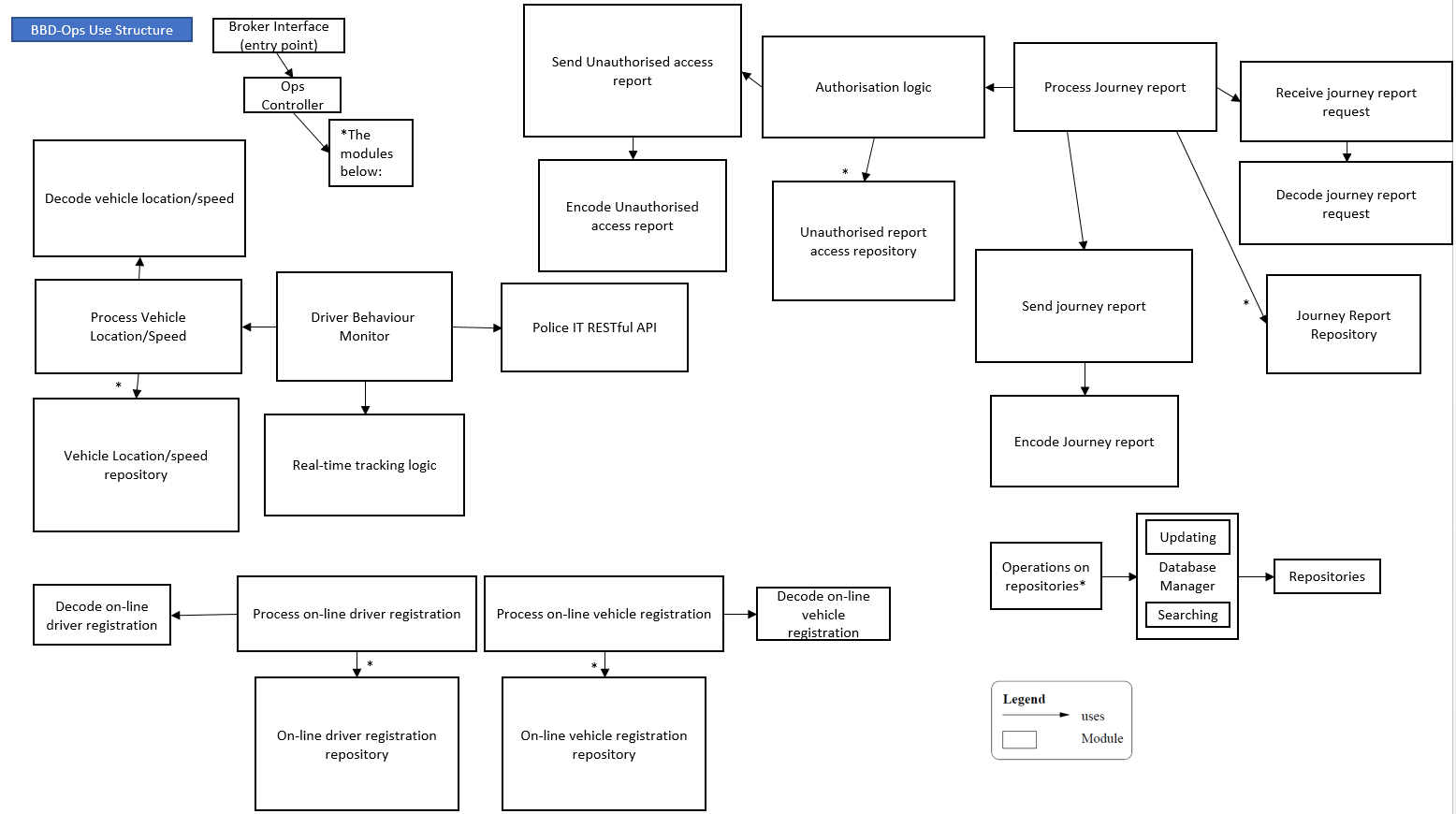


The ‘send vehicle speed/location’ module uses the ‘JSON encoder’ and the WKWYL GPS API to receive real-time details about the speed limit and road-layout for the specified GPS location. The ‘send vehicle speed/location’ module also uses the ‘encode vehicle location/speed’ module to package the speed/location data before sending. It then uses the network API included in OS to send the packaged data.

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 erratic 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.

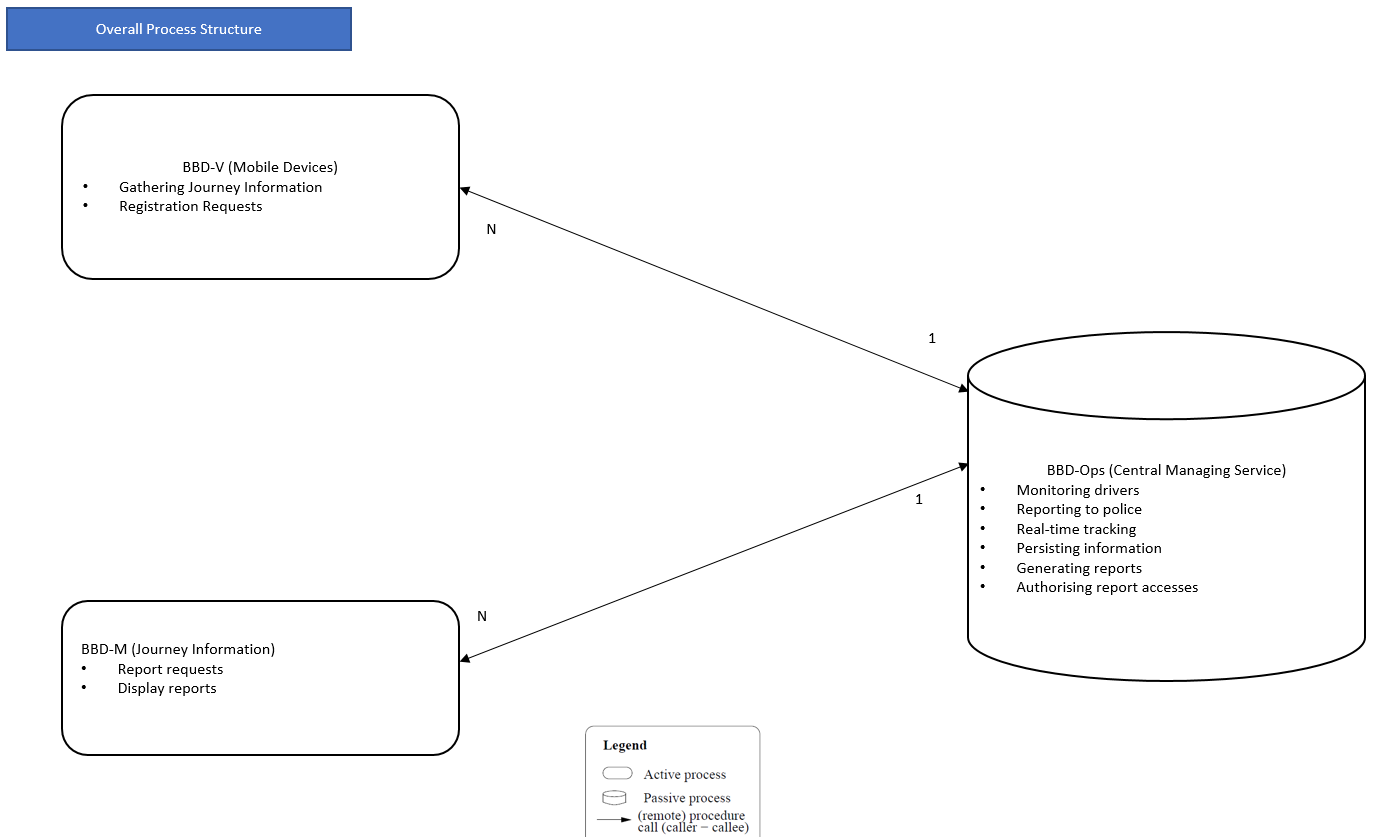
## BBD-M

## 

* 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.

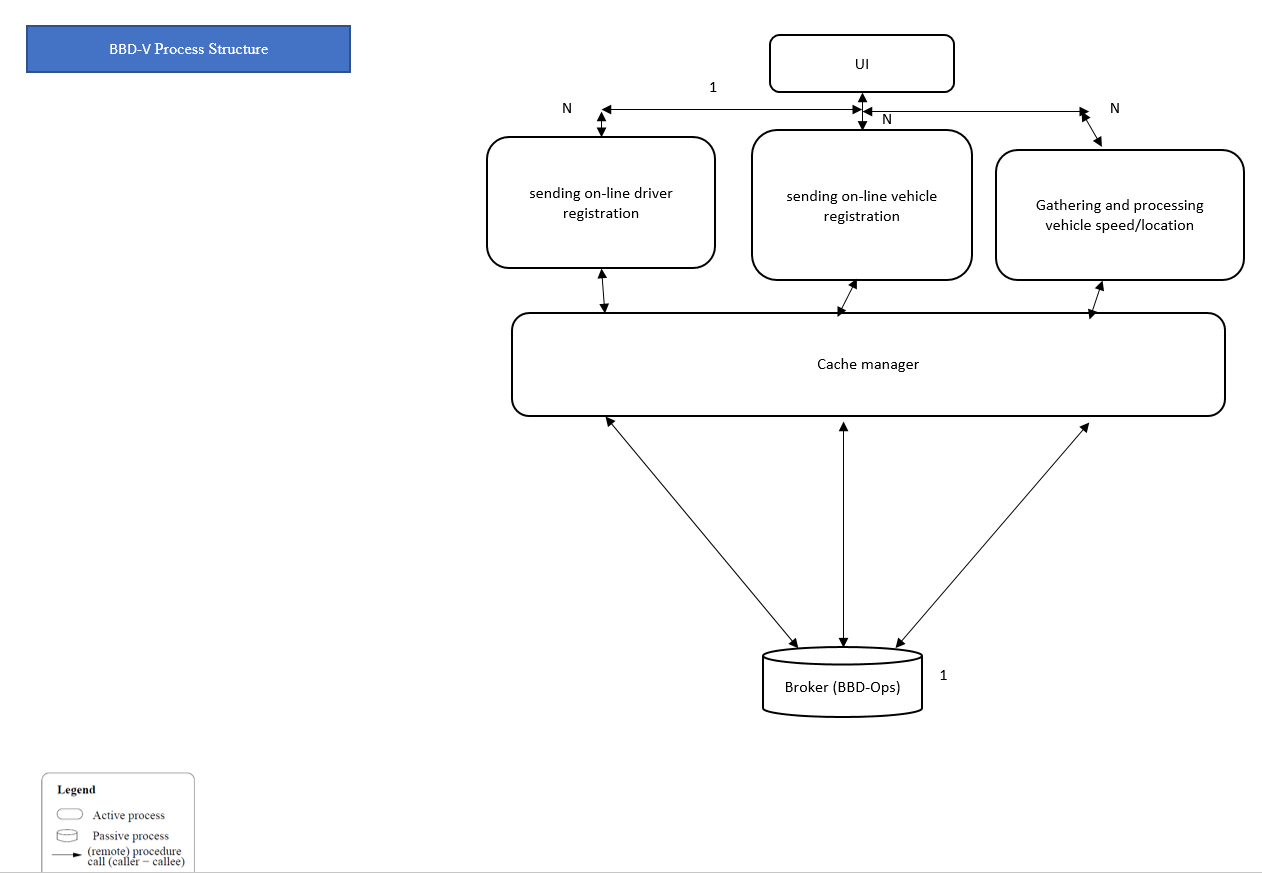
### Component and Connector structures

## Overall:



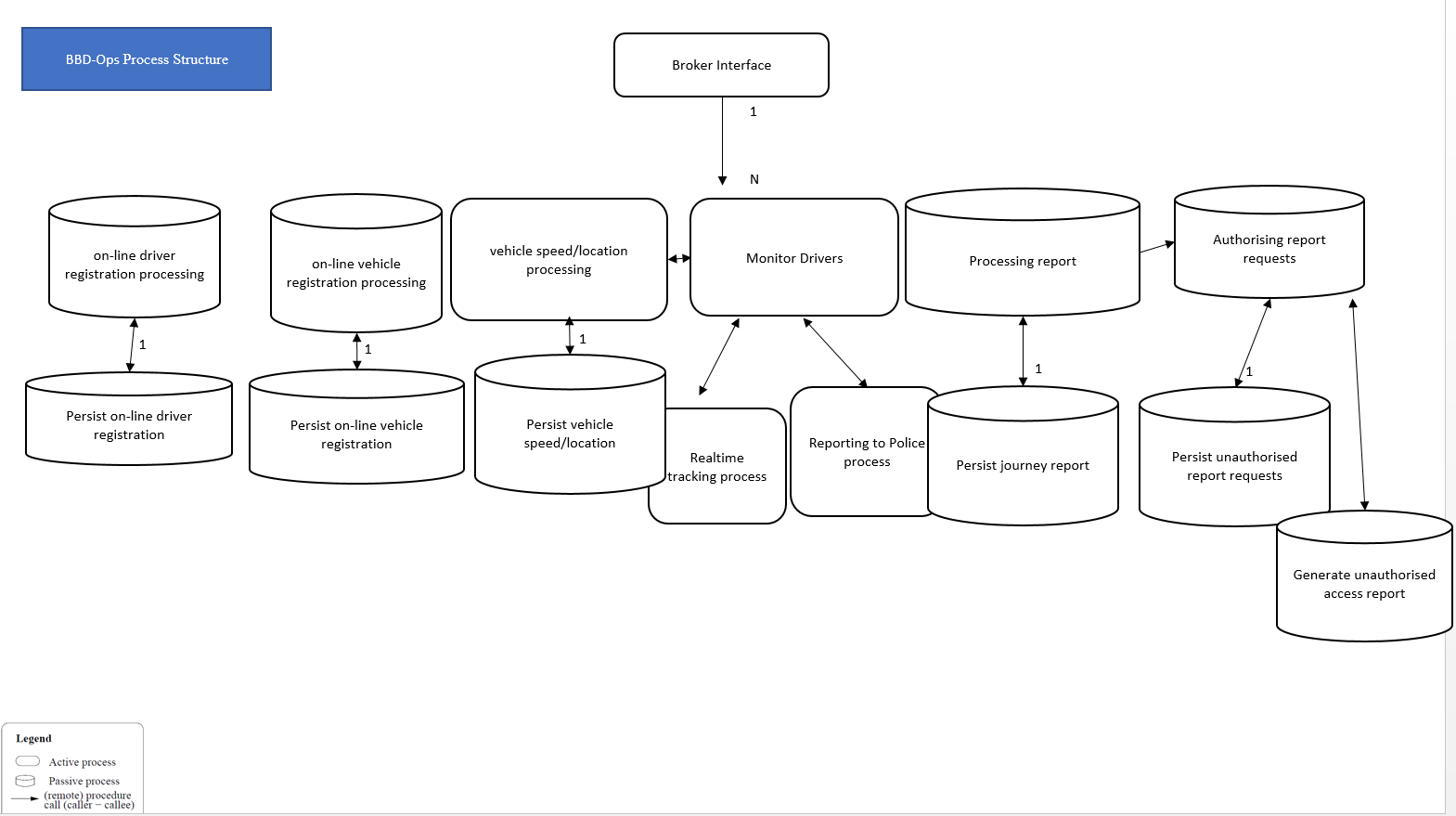
* There are two different types of components: “active” processes and “passive” processes.
* Active processes run independently from other components and can initiate communication similarly to a thread.
* On the other hand, passive processes cannot initiate communication and are regarded as “servers”.
* 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.

## BBD-V



There is 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 handled by the cache manager which control available cache resources on the client side. The registration requests and speed/location information are then sent to the broker, which is the BBS-Ops in this context.

## BBD-Ops

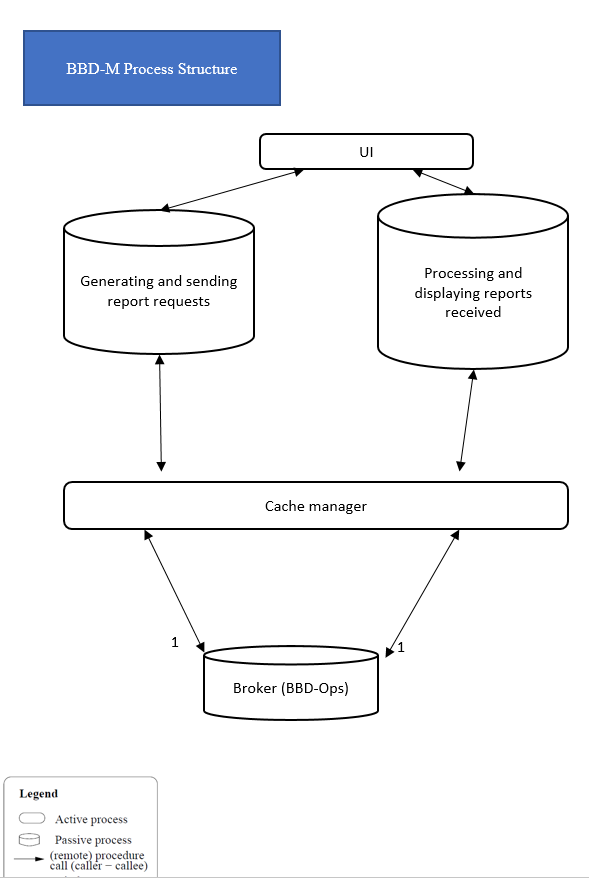


The BBD-Ops is a ‘broker’ and contains a single Broker interface. External clients interact with this broker interface.

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.

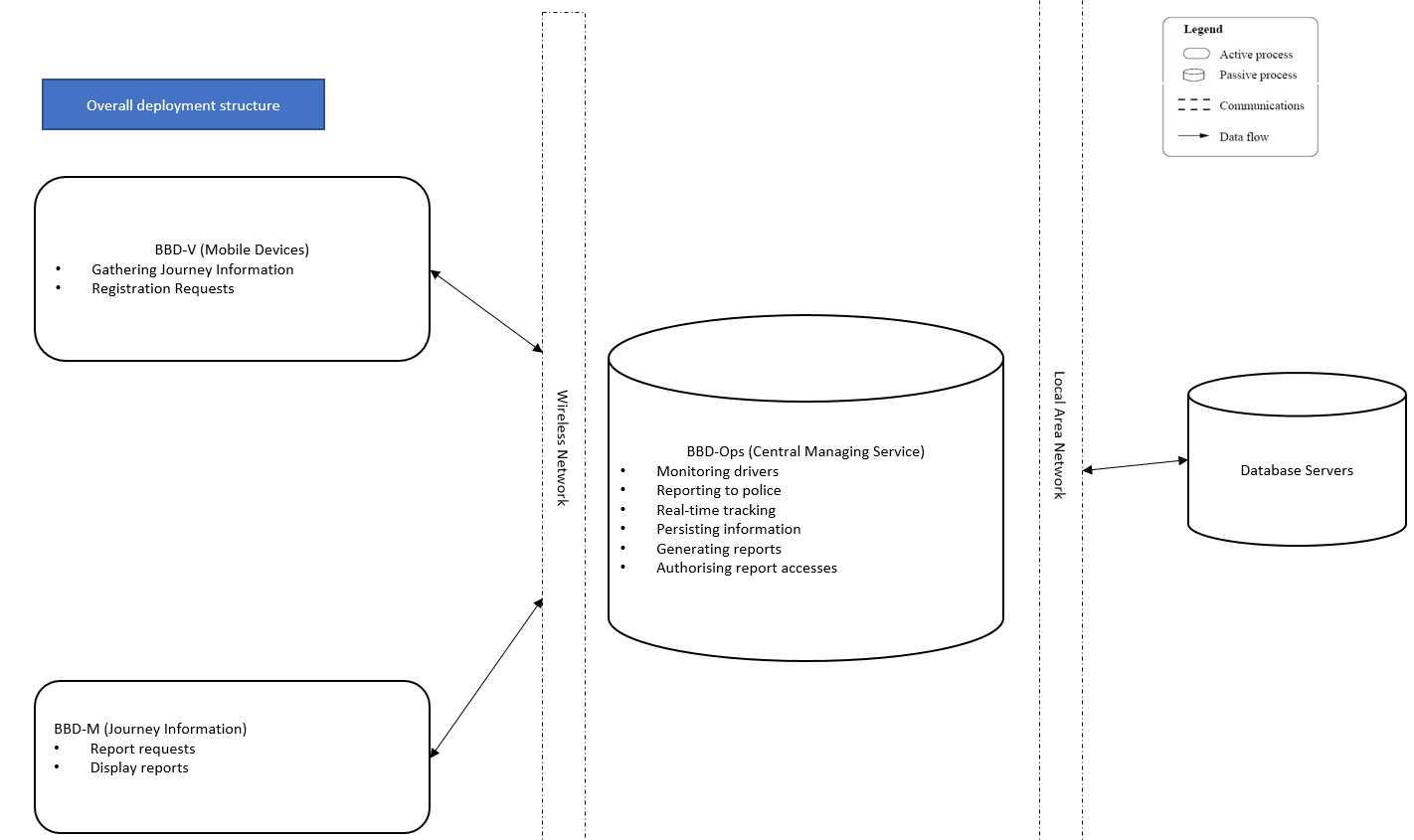
## BBD-M



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

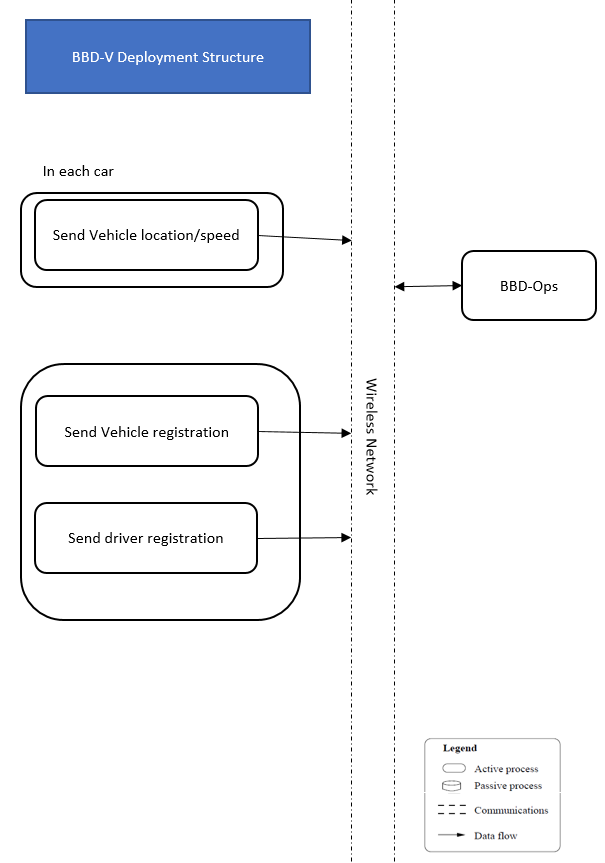
## Overall



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

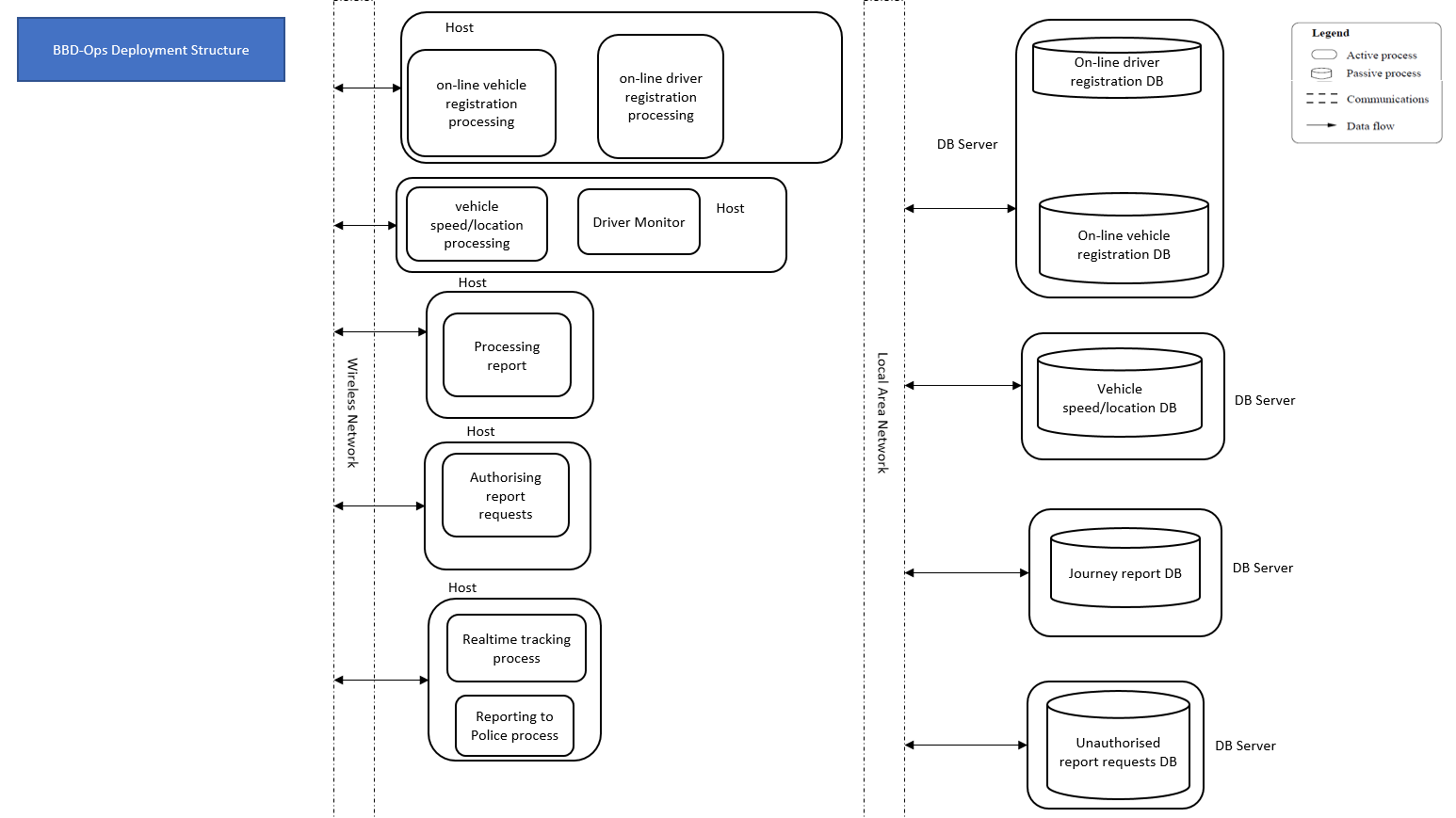


In each car, at least one ‘send speed/location’ process runs on a mobile device.

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

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

## BBD-Ops



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

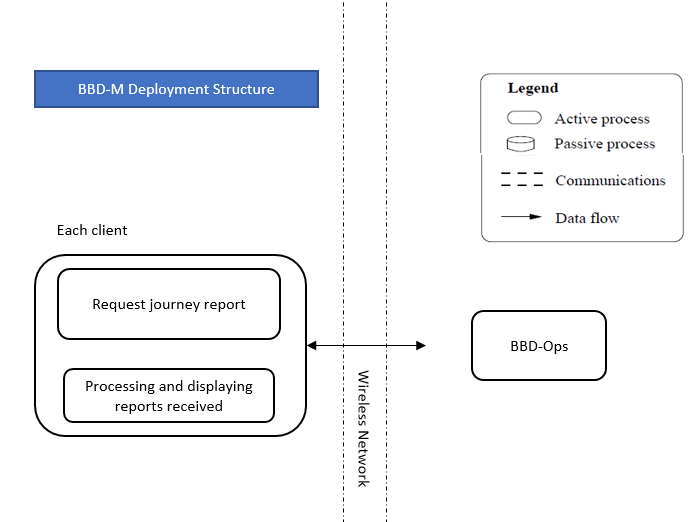
The various processes are hosted on separate hosts.

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. A 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 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.

## Broker

## 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

## Performance Tactics

* **increase computational efficiency**

There are several places where the use of an inefficient algorithm may result in the system not meeting the performance requirements, such as the authorisation algorithm in BBD-Ops. By selecting efficient algorithms and data organizations

* **reduce computational overhead**

Co-locating active and passive processes reduces 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 in which the BBD-V sends out its location and speed to BBD-Ops.

for the drivers that need to be tracked in real-time, the event rate is one per second, but for drivers who are decelerating to stop at red lights, the event rate is one per 2 or 5 seconds.

* **control frequency of sampling**
* Choices have to be made on how often the speed/location data should be recorded. More frequent means more accurate data but it will lead to 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 resources**
* having many processors is essential but having faster processors is essential as well.
* **maintain multiple copies of either data or computations**
* The cache manager is used in BBD-V and BBD-M to cache data received or before it is sent. It reduces …
* By placing multiple copies of data and server around the 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.
* 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.

## Modifiability Tactics

anticipate expected changes

The BBD-Ops is kept stateless.

If the number of clients increases

hide information (who are the clients, what is the server), runtime registration

broker interface hides internal implementations for the BBD-Ops. This allows encapsulation and

* Split module:

The application tiers are well split into separate modules which minimises the needs for changing implementations when the hardware changes.

So, when new devices are released, only the modules specifically coupled to the devices need to be changed.

The BBD-Ops contains a module that presents the broker interface. It

(encapsulate, use an intermediary, abstract common services)

## Adherence to Defined Protocols

Service registries support standards for description and data schemas. For example, JSON is used to communicate with the GPS API. This also increases cohesion because JSON is a widely used standard.

module to manage change in connection (increase semantic coherence, anticipate expected

changes

•

## Security Tactics

## Audit Trail: logging to be used by admin

An audit trail is a copy of each transaction applied to the data in the system together with identifying information. Audit information can be used to trace the actions of an unauthorised report accessor. It supports non-repudiation and provides evidence that an unauthorised request was made), and support system recovery.

## Justification

### Scenario 1 & Scenario 2

* For the data to be sent to BBD-Ops, latitude, longitude (GPS coordinates) and vehicle ID will be sent in 4-byte long type. 2-byte int type will be used for speed and acceleration. Therefore, a total of 16 bytes is used for each message sent.
* A checksum of 4 bytes can be appended if necessary. Further encryption can also be added. But This will increase the overall package size.
* In the worst-case scenario, we are sending messages to BBD-Ops every second when tracking in real-time. The data packets sent to BBD-Ops are: 16 bytes \* 30 Journeys per month \* 60 seconds \* 20 minutes = 576000 bytes = 0.576 MB
* Therefore, this is within the data limit restriction of 5243000 bytes.
* By reducing sampling rate at certain conditions, such as when the vehicle is waiting at a red light, this data usage can be reduced.
* There are 4000000 road vehicles according to NZTA. Each vehicle is assumed to make on average one journey per day. Say in the worst-case scenario, they all send speed/location packets to BBD-Ops simultaneously. Then BBD-Ops will receive:
* 4000000 vehicles \* 16 bytes = 64000000 bytes
* BBD-Ops uses local area network to communicate with database servers. The 64000000 bytes is within the average local area network speed of 100Mbps and should be sent in 0.64 seconds.
* The average 3G network has a download speed of 800Mbps, the download speed for a packet is therefore 1.6 × 10^-6 seconds.
* The average 3G network has an upload speed of 10Mbps, the upload speed for a packet is therefore 2.0 × 10^-8 seconds.
* 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 can depend 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** run on, as well as efficiency of the algorithms. We can improve the hardware (**increase available resources** and **introduce concurrency**), and may be able to optimise 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

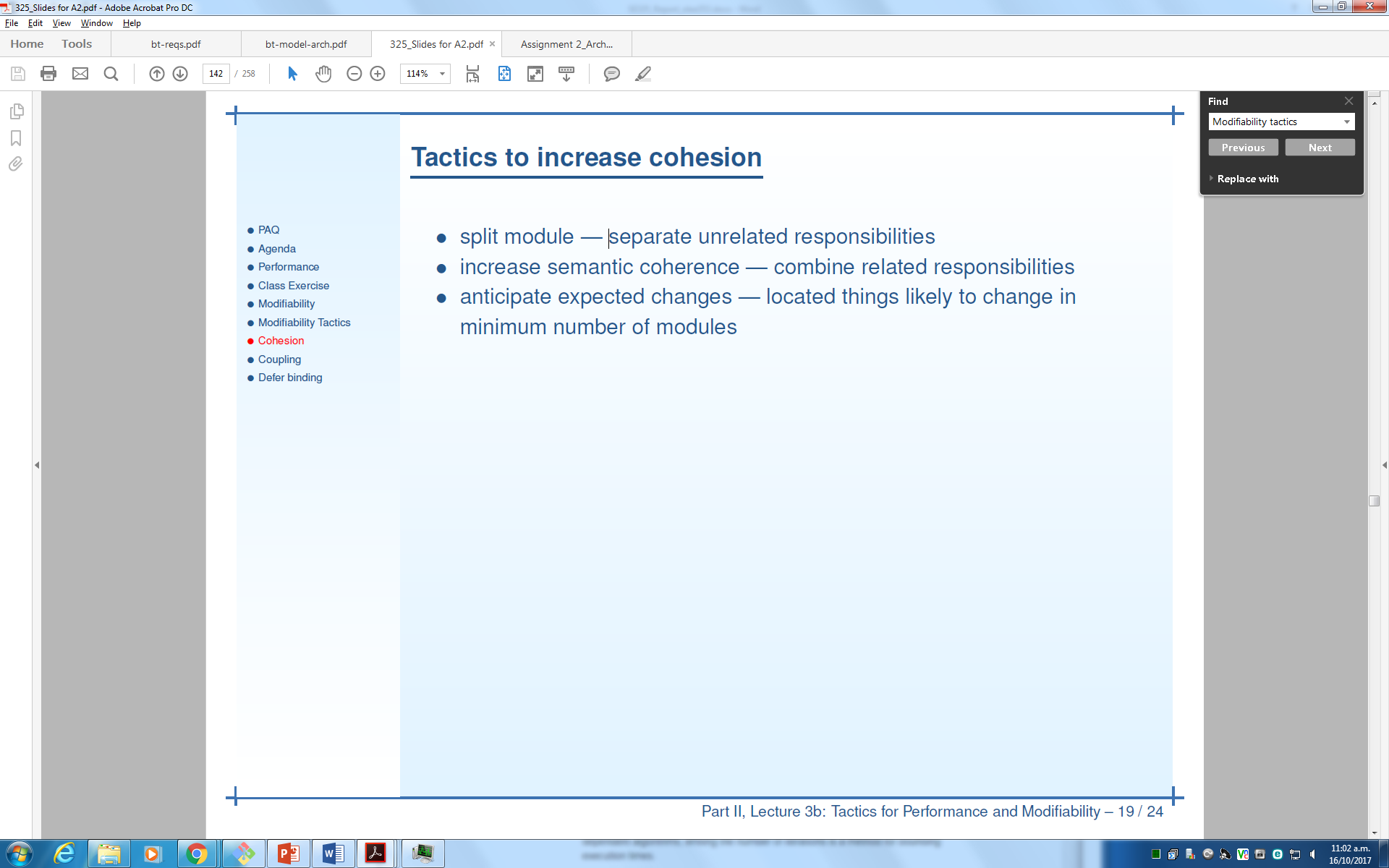
Reduced coupling by separating the BBD-V and BBD-Ops. The BBD-V interacts with BBD-Ops via it Broker interface. This increases semantic coherence and encapsulates BBD-V internal implementations.

• 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

### Calculation

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