Software Engineering 3: Vehicle control system	
	Laboration 3

Laboration 3.1

Quality attribute	Attribute Refinement	Scenarios
Performance	Resource utilization, Corrupted data	Resource utilization : Sensors usage Stimulus: Sensor data Environment: Normal system Response: Real time data process Corrupted data: Information not valid Stimulus: Actuators have non-true information Environment: Emergency mode Response: Show error notification
Modifiability	Maintenance cost , Code flexibility	Maintenance cost: Modifying the system Stimulus: Modify functionality Environment: Design time Response: No effect on other non-touched modules Code flexibility : Adding new functionalities Stimulus: Code re-written Environment: Build time Response: Deploys modification
Availability	Available all the time, Recovery	Available all the time: System available when ignition starts. Stimulus: Engine started Environment: Start mode Response: Real time availability Recovery: Fast recovery from errors in sensor Stimulus: Sensor fails Environment: Degraded mode Response: Notify + Log failure

Laboration 3.2

Discussion:

Performance→As timing is everything in such systems, the way in which the dependency between modules and their elements is present in our system might create a bit of overload in the system. So, the response to events might require a high consumption of resources may be at the same time the system is handling other things. Such dependencies may produce latency which is not good. This dependency is not good and also goes in a trade off with availability as system may be not usable for some time.

Modifiability→ When discussing about modifiability, we recall its aspects such as: how easy a component can be "restored/updated/modified" in order to gain better performance for its functions or in extreme cases to adopt to a new environment. So, modifying the code and analyze it in the modifiability point of view, makes us think more in depth about its concerns that it has such as extensibility or functional flexibility. In our case, modifying the code for example: to add new functionalities to the parking assistance (extensibility) would require to turn an existing ability to new usages (functional flexibility) (Bass, Clements, & Kazman, p. 88). In Lab 2, we said that our system respects the low coupling and increases modifiability, so adding/modifying modules should be easy.

So, in this case adding and changes can be considered as positive, as adding new actuators and sensors would be translated into adding a new module. It is a trade off with performance since the code modification may alternate for good the performance considering time measurements after the modifications are done.

Availability→ Critical systems such ours should be operational under good conditions and also to be mentioned in such systems, failure is not permitted. So, the system is said to be under good conditions if modules interact in the way that they were designed and no differently. If we have not good interaction between modules, this means that the system would suffer from not good behavior and jitter may lead to a system crash and a restart may be required. This would be a negative thing for our system, so something needs to be modified. This is why it is a trade off with modifiability. Something needs to be modified after the detection of faults has taken place. The system can be better optimized in order to eliminate points of failure and recover.

The table below is just a summary of the above description.

Quality attributes	Architectural decision as sensitivity point	Effect	Trade point
Performance	Many modules use other modules	Negative System has to prioritize. Overloaded system might be an example. Failure is evident in this case	Performance vs Availability
Modifiability	Code is modified	Positive New module is added	Modifiability vs Performance
Availability	If a module crashes, other modules may suffer as the information would not be exact.	Negative System might crash and might need restart.	Availability vs Modifiability

Lab 3.1- Time spent: 1 working day was necessary for doing this as most of the time was spent on related scenarios.

Lab 3.2 - Time spent: 0.5 working day was necessary as most of the time was spent on reviewing the Lab 2 Designs to understand the sensitivity points.

Bibliography

Bass, L., Clements, P., & Kazman, R. (2012). Chapter 4. In *Software Architecture in Practice.*