Estimated workload: 3 days full time (2 students)

Background

You are the software architect at a major car manufacturer where your job is to design and document the software architecture for the complete vehicle. In a modern car a number of electronic control units (ECUs) are connected through different networks. An ECU is an embedded computing system with a processor, memory modules, I/O connectors to sensors and actuators, one or more network interfaces, etc.

Today, it is not uncommon to find more than 80 ECUs in a vehicle. This is partly the result of the idea that every function should have its own ECU. However with an exponential growth of functionality controlled by electronics and software this is no longer possible. Your company has decided to develop a new generation of electronic control systems where all software will run on a single central computing system.

In the first car to use this new type of control system, a limited number of functions will be controlled by software, including those listed below.

Car alarm. A siren sounds if someone tries to do a forced entry to the vehicle.

Adaptive Cruise Control. The adaptive cruise control adapts the speed so that the car keeps a fixed distance to the vehicle in front.

Collision Avoidance. This function actively takes over steering and breaking from the driver in order to avoid a collision.

Parking assistance. Parking assistance guides the driver when parking the vehicle. It starts to beep if the car get to close to obstacles and prevents collision by breaking if necessary.

Anti-lock Braking. This function prevents the brakes from locking. When a wheel locks due to heavy breaking from the driver the system will take over and release the brake to avoid locking.

Electric Power Steering. This function assists the driver in steering the car with the help of an electric motor. The power of the electric motor is automatically adjusted such that more assistance is given when the car moves at a low speed.

These functions require that the computing system interacts with different driver controls (such as the steering wheel and the brake pedal) as well as sensors and actuators. (A sensor is a device that measures some physical quantity, such as the vehicle's speed or distance to an obstacle, and delivers an electrical signal to the computing system; an actuator receives an electrical signal from the computing systems and produces a physical effect in the vehicle, such as controlling the position of the front wheels or the brakes.)

The software architecture of the system should be designed in such a way that it can be used as the basis for future vehicles where more functions can be added and some functions left out. Consequently, the set of external devices that the software needs to handle is expected to vary between different vehicles as well. Furthermore, the basic computing platform, including the processor and operating system, is expected to vary between car models. In all cases, it can be assumed that the software will be implemented in an object-oriented programming language and will run on top of a real-time operating system with support for multi-tasking.

Your task

Your assignment here is to design and document a software architecture for the vehicle control system described above. The system is to be used in a car where the software will control the six functions listed above plus at least two more functions that you decide. You must also decide what sensors and actuators are needed in the vehicle in order for the control system to perform these functions.

When designing your architecture you should keep in mind that more than one function can use the same sensors and actuators. For example, the steering can be controlled by both collision avoidance and the electric power steering, while the brakes can be controlled by the adaptive cruise control, parking assistance, anti-lock braking and collision avoidance. The functions should be independent, such that designers of future vehicles can choose to include any combination of functions. For example, there should be no dependency such that if you include the automatic parking you must also include the collision avoidance.

Laboration 2.1

Think about who the different stakeholders of the envisioned system are. Identify at least five key stakeholders and describe their interest in the software architecture. Include a description of which of the following "standard" architectural views each of these stakeholders may be interested in and why (see Tables 1.1 and 3.1 of [BCK12]):

-decomposition view

-uses view

-concurrency view

-implementation view

-work assignment view

State approximately how much time you have spent on this assignment.

Laboration 2.2

Design and document a combined decomposition/uses view of the architecture. The view should be at such a level of detail that each software element represents something that can be left out if a future vehicle is to be made without a certain function and/or without a certain sensor or actuator. Document the view using the template in Section 18.6 of [BCK12]:

The primary representation shall include a graphical overview of elements and relations, e.g. as one or more UML diagrams or as a more informal diagram along with an explanation of the notation used.

The element catalogue must include a complete list of elements with descriptions, as well as descriptions of different types of relations. Take particular care to document any elements and relations that you have chosen not to show explicitly in the primary view. For simplicity, you can leave out Sections 2.C and 2.D of the documentation package.

The context diagram shall be a graphical description of the software system's relations to external entities such as human actors and hardware components. Use an informal but informative notation.

The variability guide must describe how your architecture, from the perspective of this particular view, is intended to support variations in functionality and hardware in future vehicle models.

Finally, remember to document the rationale of your design (why you have chosen to design the view as you have) and any assumptions you have made. Include references to any architectural tactics and patterns/styles from literature that you have used.

Again, state approximately how much time you have spent on the assignment.

Laboration 2.3

Design and document a concurrency view of the architecture, using the same template as above. Again, you can leave out Section 2.C, but provide a description of the runtime behaviour of each element (either textual or graphical, e.g. using UML state chart diagrams). It is not necessary to provide a context diagram for this view, but include a variability guide and rationale.

Recall that this is an example of a components-and-connectors view, comprising dynamic (i.e. run-time) elements and relations, while in Assignment 2.2 you designed a module view, comprising static (i.e. development-time) elements and relations. For some additional inspiration on how to document views it may be useful to read [Kruchten95] and [FCGSS07], which include a component-and-connectors view called the process view and a module view called the development view.

State approximately how much time you spent on this assignment.

Reporting the assignment

Create a subfolder called "Lab 2" in your Subversion repository containing the following:

A document (Word or PDF) with your solution to the assignment (do not forget the estimates of how much time you have spent)

After committing the file to Subversion, you must notify us that you are done by submitting the assignment in Blackboard and write your group number (in the Text Submission field). Each group member needs to submit the assignment via Blackboard individually.