

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

Software Architecture Design and Documentation

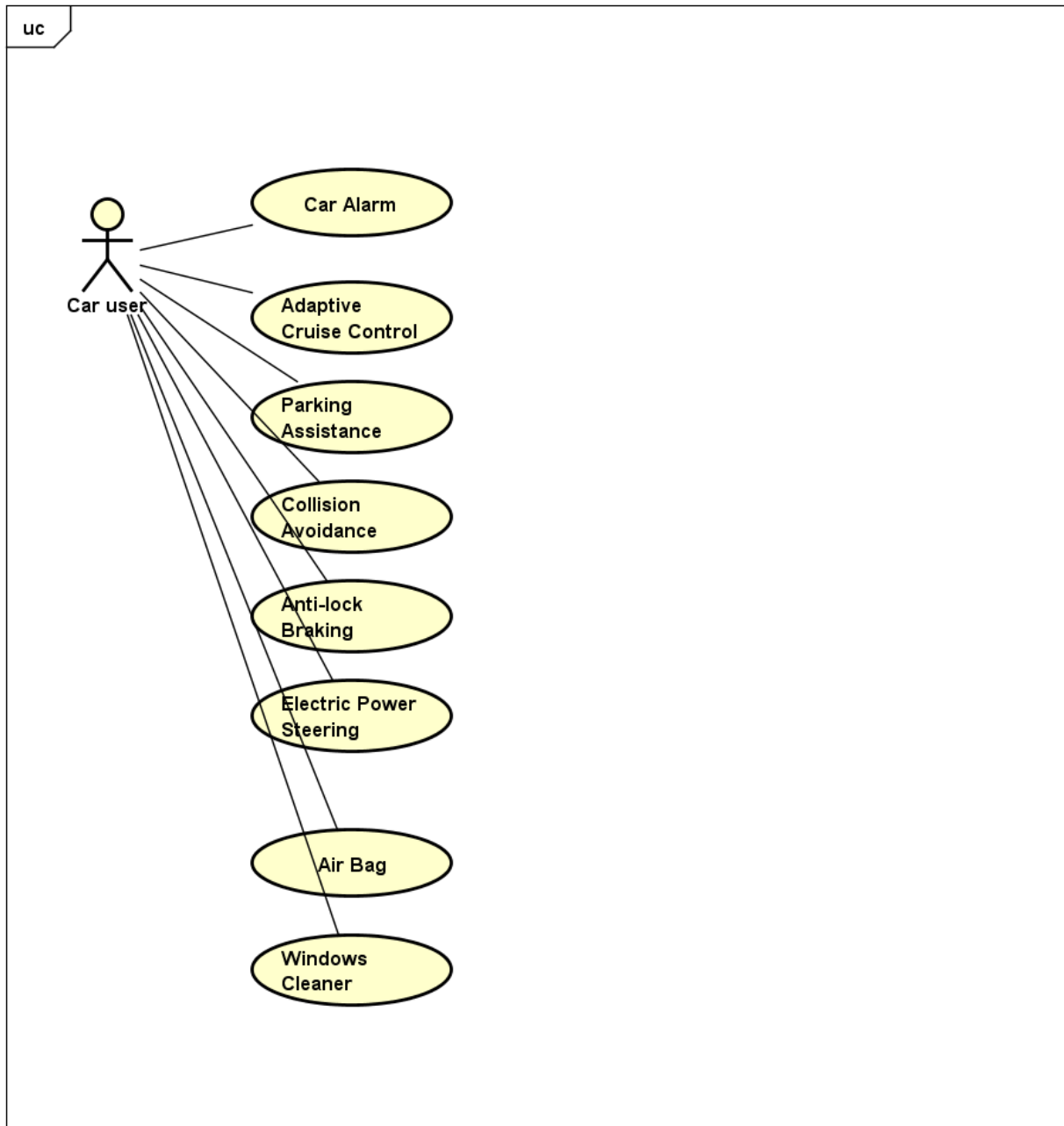
Vehicle control system

Albi Dode
ade15002@student.mdh.se

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

SAD Revision History

Date	Version	Description	Author
2016-02-05	1.1	Starting the writing	Albi Dode
2016-02-29	1.2	Finalize	Albi Dode

Figure 1. Use cases

powered by Astah



Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

Table 1. Functions and Sensors/Actuators relation

Function name	Explain	Sensor;Actuators
Car alarm	Beep if car is being stolen	Motion detection, radio frequencies;siren lock,light
Adaptive Cruise Control	Keep a speed in certain distance	Speed,distance,reflectors,laser;anti-lock,speed control, throttle
Parking Assistente	When parking, when a distance is met it starts beeping	Camera,distance;brake,siren,electric steering,speed boost
Collision avoidance	Avoids collision	GPS,distance,camera;steering,adaptive cruise control
Anti-lock	No lock in brakes	Distance,speed;brake, valves
Electronic Steering	Electric motor help when low speed for car navigation	Position sensor; Hydraulic or electric actuators
Air Bag	Avoid injures	Collision avoidance;valve
Window cleaner	When rain stars, automatically clean it	Rain sensor;windscreen motion

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

Assignment 2.1

There are different views and for each view different stakeholders have different interest. Also, to be pointed out is the fact that depending on the project or company, stakeholders may vary in number.

Starting with **project manager**, whose interest towards schedule and resources is bigger than showing interfaces for example. Yet, constraints, environment, interaction arises the interest. So, in general these views will be read carefully:

- Module – as context diagram
- Deployment+Decomposition+Uses - as a layered view
- Work assignment – organization purposes


Development team member, as name suggests, is driven by the given architecture as the whole work process depends on that.

- Decomposition, uses, layered, generalization view – as they need to know what modules are and general relationship
- Concurrency – interaction of components
- Context diagram, interface specification, variability guide – knowing the environment and boundaries.

Testers are the ones that see architecture as a black box. Still, they need information about integration of classes.

- Specifications, uses view are important to them.
- Implementation view – to locate the needed files for the module
- Deployment view – as it addresses issues of installation and performance
- Context diagram, interface specification, variability guide – guide on how to test

Maintainers are the ones that will deal with future changes. So, somehow, they are future developers plus some info more about decomposition view as it will allow them to find the place where to do the change and also a uses view and layered view would be in help as the effects of the changes must be predicted not only by just seeing the design rationale. In short, they are interested in everything.

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

Application builder is the one who resides in the product line and shapes the aspects of the system by making adaptations.

- Implementation & decomposition/uses view- here the needed artifacts and configuration management and the assembly can be seen.

Customer is the one who pays for the software. Generally their bigger interest is in the progress and costs and in interoperability of the system. For these reasons:

- Work assignment view – how the time was divided on the team members
- Deployment view – to get a general idea of the structure

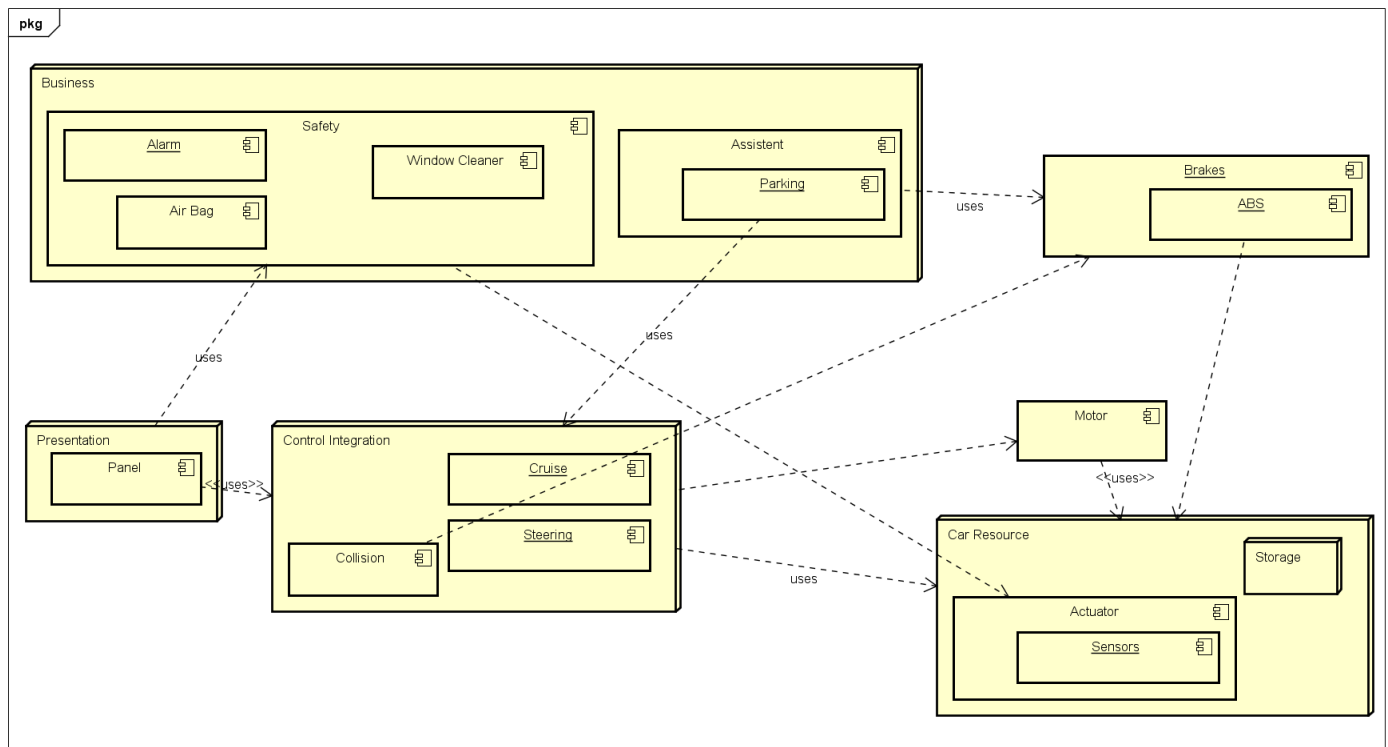
End user are not much interested in the architecture rather than its functionality. Sometimes they are interested on how to gain the maximum of the system so, they might need:

- Deployment view – for functionality understanding
- Sequence diagrams – how the right usage of the software should be


Time spent on the assignment: Total worked hours for this assignment are those of a half normal working day. After the stakeholders were identified, each of the views were connected to a certain stakeholder while giving a why for that.

Assignment 2.2

- **Primary presentation**



powered by Astah

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

- **Element catalogue**

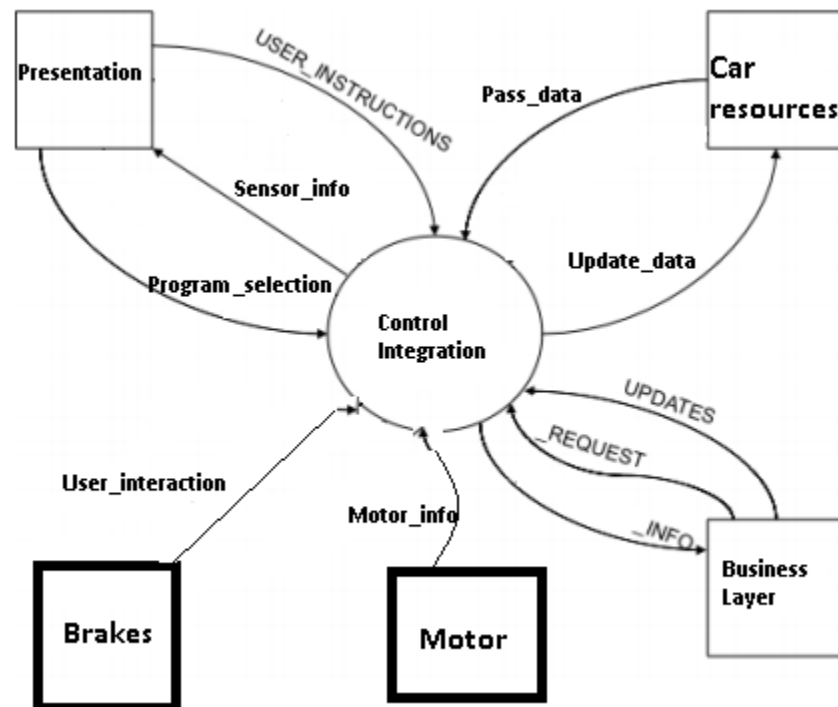
Name Property	Responsibility	Implementation	Relation	Elements
Business	<p>Should hold helping functionalities that every model must have.</p> <p>Alarm- Notify in time</p> <p>Parking – Help for close distance</p> <p>Air Bag – Work in time</p> <p>Cleaner –Easy the view</p>	<p>Alarm – lock + sensor interface. When the sensor trigger is activated, the alarm sounds.</p> <p>Parking – sensor and distance from camera input. The alarm can be modified by user.</p> <p>Air bag – hit sensor</p> <p>Cleaner – sensor triggered</p>	<p>Most of the classes make usage of the sensor and have distance as input. Beside cleaner, others can exchange information among them.</p>	<p>Alarm</p> <p>Parking</p> <p>Air Bag</p> <p>Window Cleaner</p>
Presentation	Panel – As a substitute for the term GUI in a car.	Panel – Appropriately calls the getters from all the other class instances to show info.	Relies on other classes instances.	Panel
Control integration	Cruise & Steering – Helps in driving and keep the right distance	Cruise – its parameters are sensor and speed and are evoked in an appropriate method	Sensor related. Also motor and brakes instances are present	Cruise Steering Collision avoidance

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

		Steering & Collision– It can be an extension of cruise		
Car resource	Sensor – Gets information and elaborates it appropriately Storage – Logs the actions	Sensor – getters/setters Storage – Relies from sensor and stores the info	Crucial for the system	Sensor Storage
Brakes	Stops or used to keep distance	It can be integrated with cruise control+abs.	Used with cruise and parking	-
Motor	Used to accelerate	Instantiated by user interaction	Cruise instance	-

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

- **Context Diagram**



Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

- **Variability Guide**

As this is a module view we focus on the various versions of parametrization

Business

It is named business as here same basic functionalities of a car reside. Parameters here are predefined or user through the panel can modify only some of them. If variations in functionality may happen in future, they can be in accordance of not depending too much on other modules functionalities.

Control integration

If new functionalities would be required to be added, a redeployment of the system may be required, as here every other module has a relation with it.

Presentation

Here the driver cannot do anything that an admin can do, but still as it is related to the control integration, he/she can modify some parameters from that module. Like the user can select from the menu like the distance that the parking assistant should work and so on with small details. This may require also some changes in the storage.

Storage


All system messages can be stored in the storage system, and there may exist different versions which are self-updated/deleted according to the time or outside interaction. Changing the file where this information is stored: structure or etc. can mean even the redeployment of the system. (This case may happen due to errors not forecasted in the tests, or new requirements).

Brakes

As they reside as an independent module, changes here are welcomed, like changing the parameters value etc., and also a future implementation could be making a further dependency on sensors while parking.

Motor

Again as motor is independent, its functionality is the one that makes the car run. As tale, a motor can be further enhanced or etc., as being a functionality and independent it gives the possibility to further implementations.

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2


- **Rationale**

This view was chosen as we think it eases the implementation of the vehicle control system. As for a process the code behind resides in the controller. Some pre-installed files in the storage can make the system run even without user input, and still it makes the mapping of classes and events.

While designing we kept in mind the low coupling between sensors and others. Also, we tried to separate sensors from their data usage. This makes possible that the modification that can be made from the sensors outputs can be different and independent from each other. In this way future integrations can be easily done.

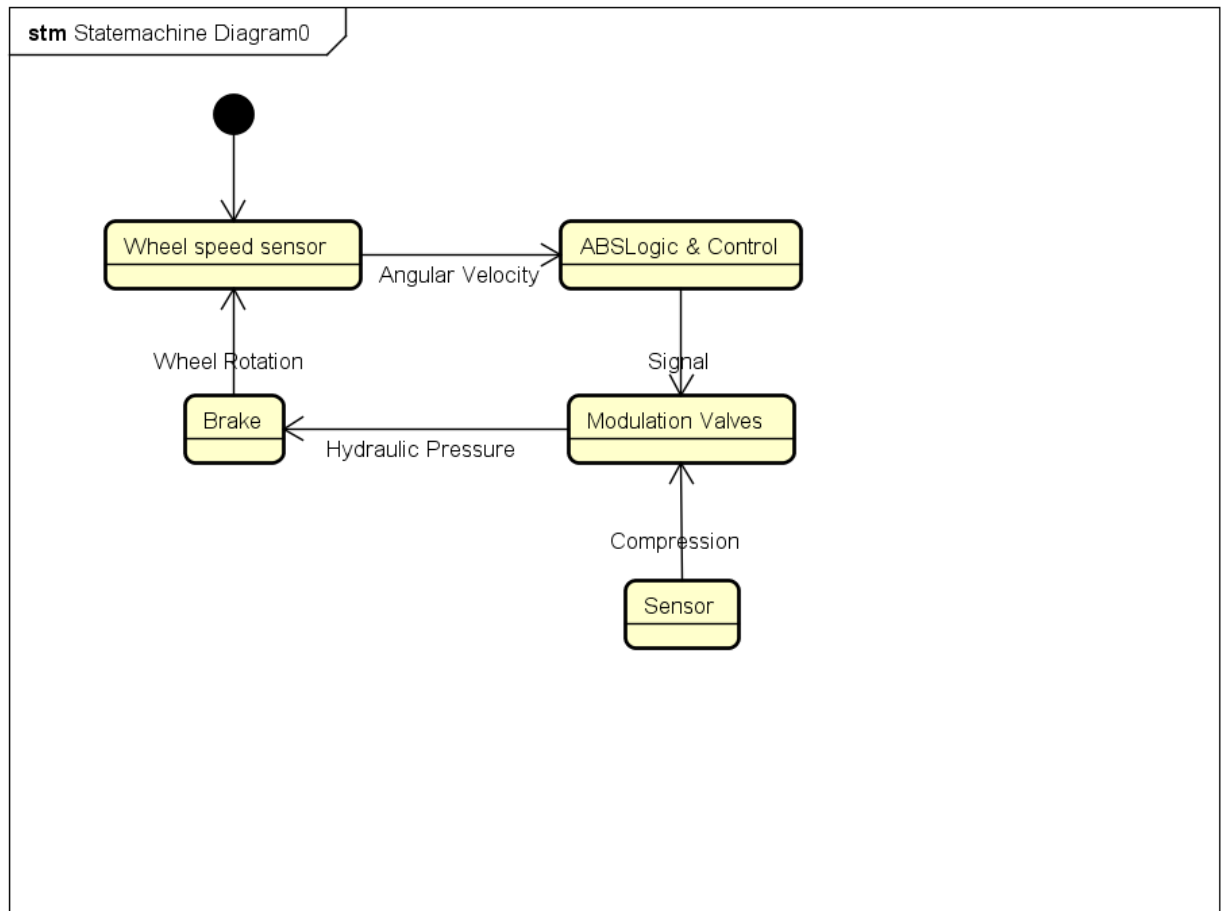
Also some rejected alternatives happened as assumptions about the user interaction were needed, and so the design needed to be updated.

Time spent on the assignment: As understanding this assignment was essential even for the understanding of the other assignment, 2 days were enough to make it. Most of the time was spent on trying to understand the variability guide and its connection to this view.

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

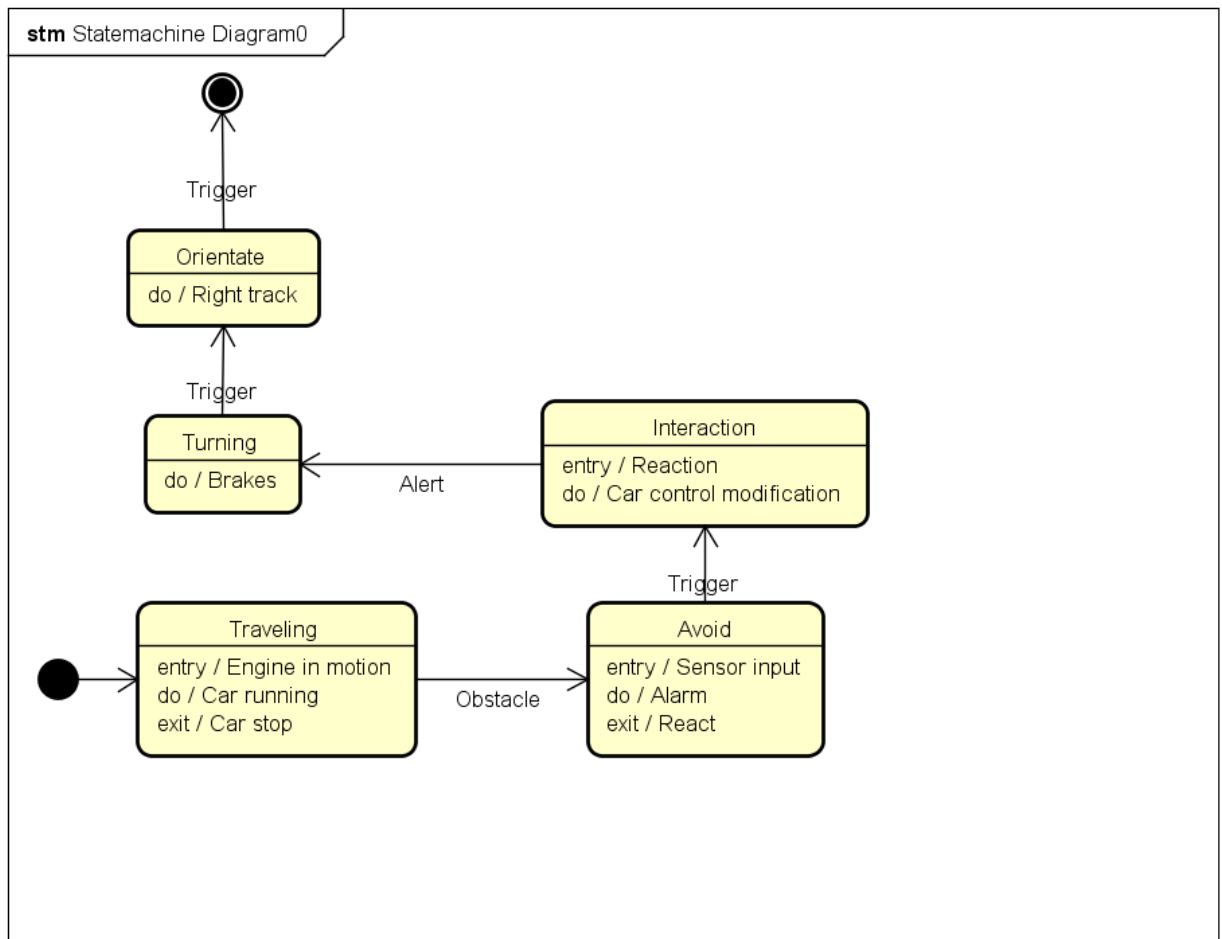
Assignment 2.3

1- Primary presentation



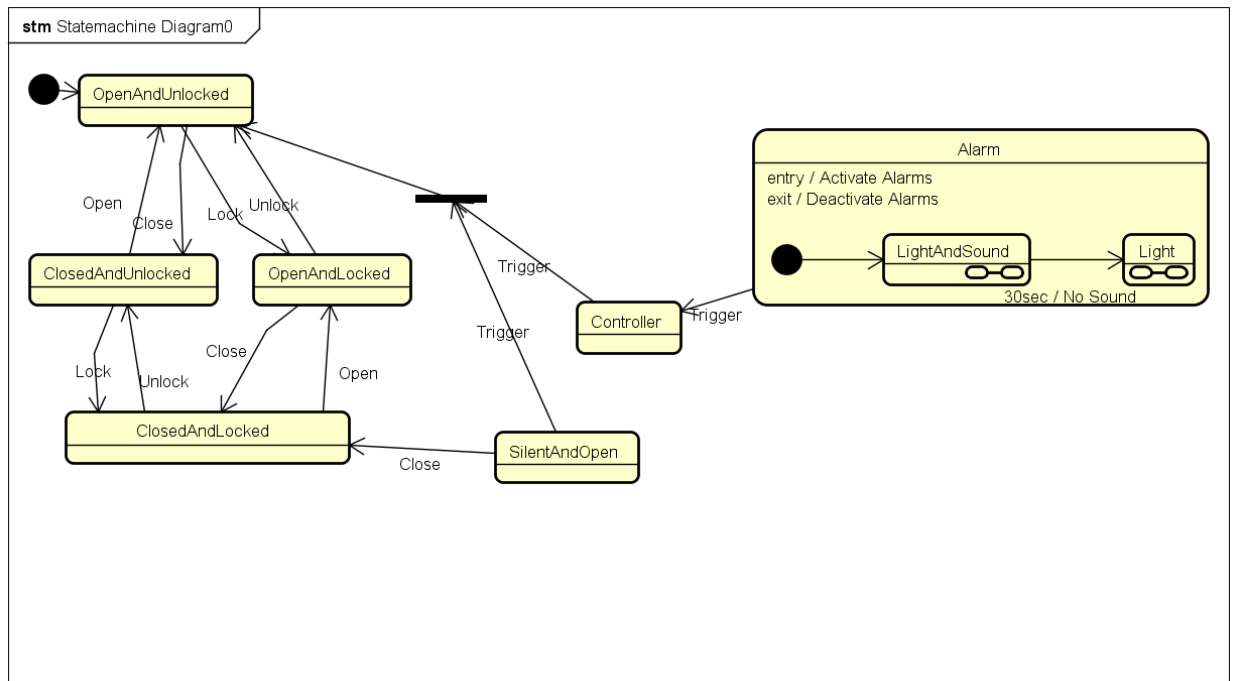
powered by Astah

Figure 1. Anti-lock statechart diagram



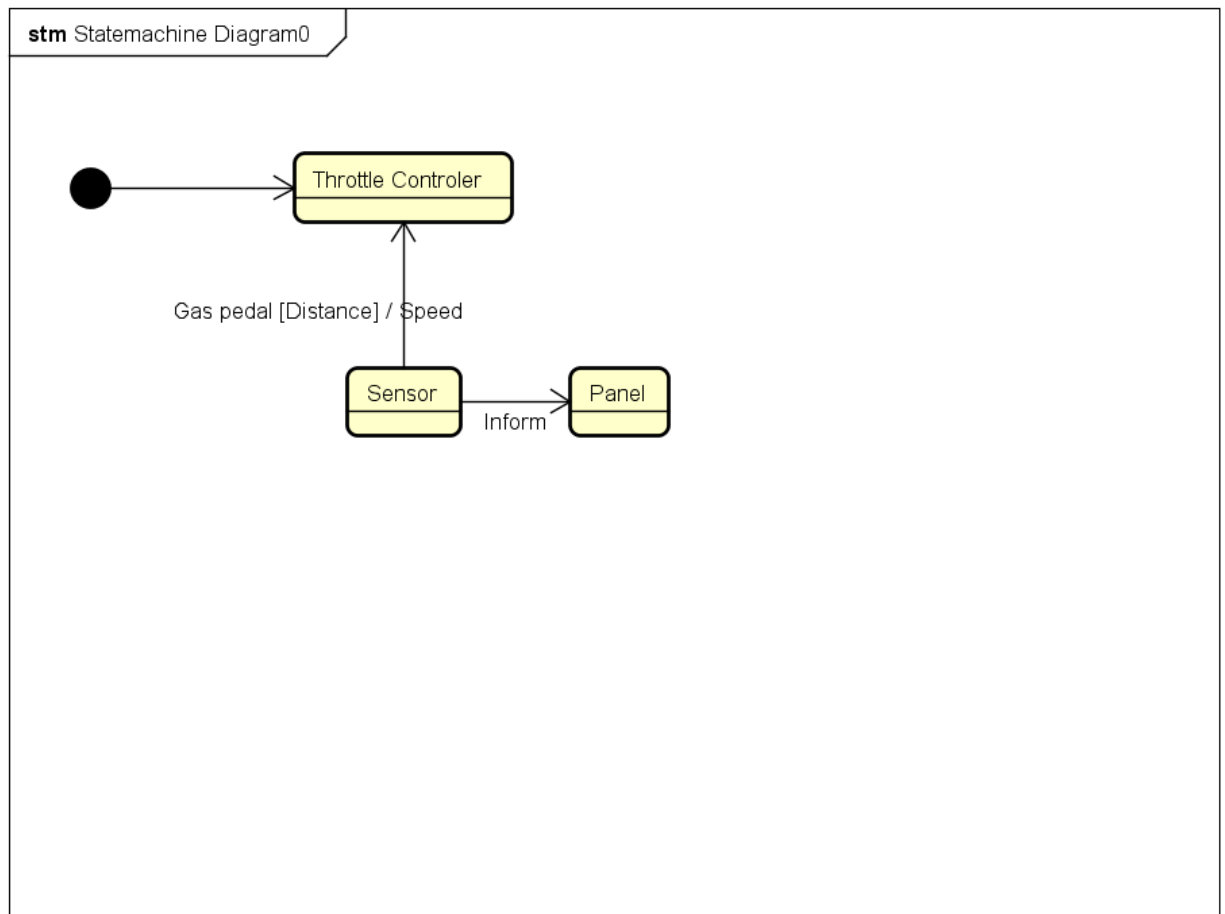
powered by Astah

Figure 2. Collision avoidance statechart diagram



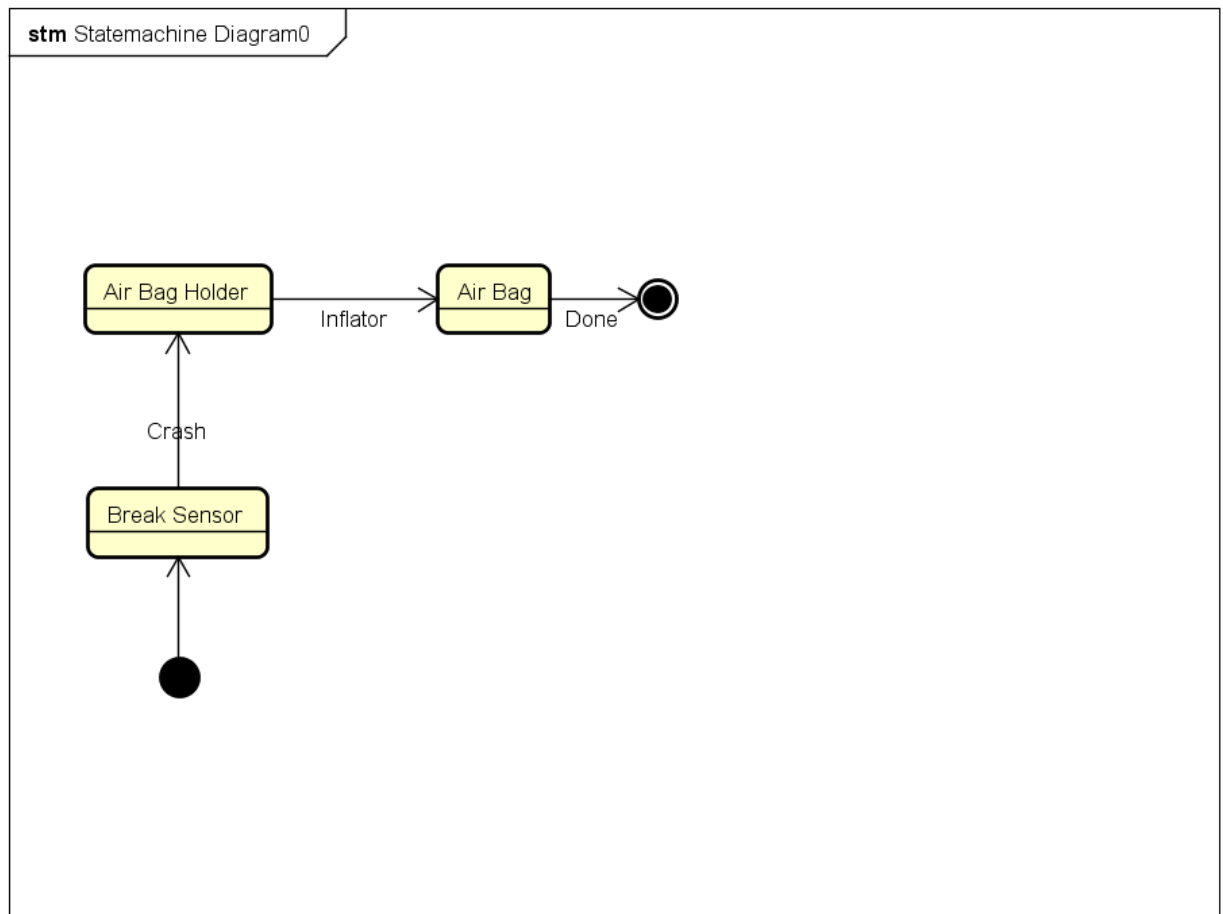
powered by Astah

Figure 3. Alarm statechart diagram



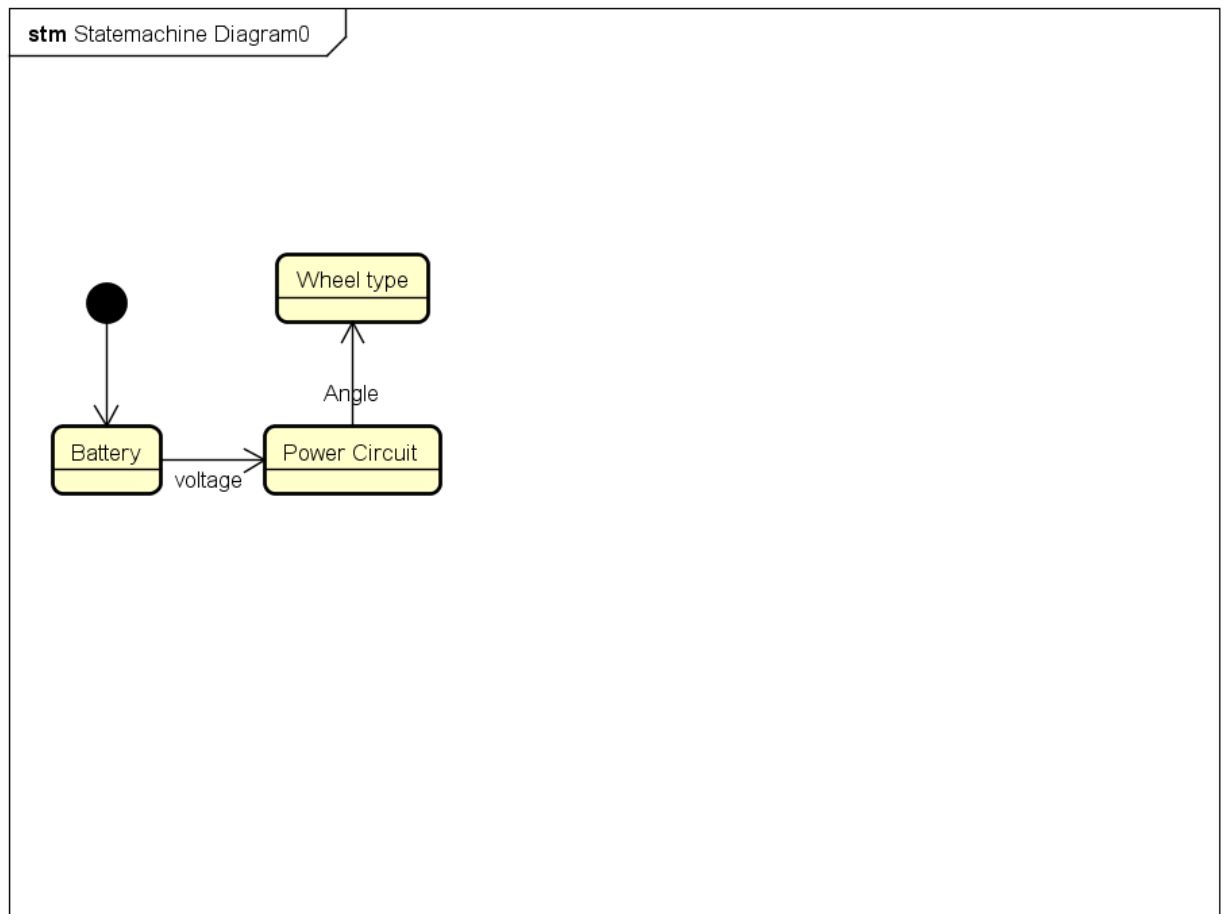
powered by Astah

Figure 4. Cruise control statechart diagram



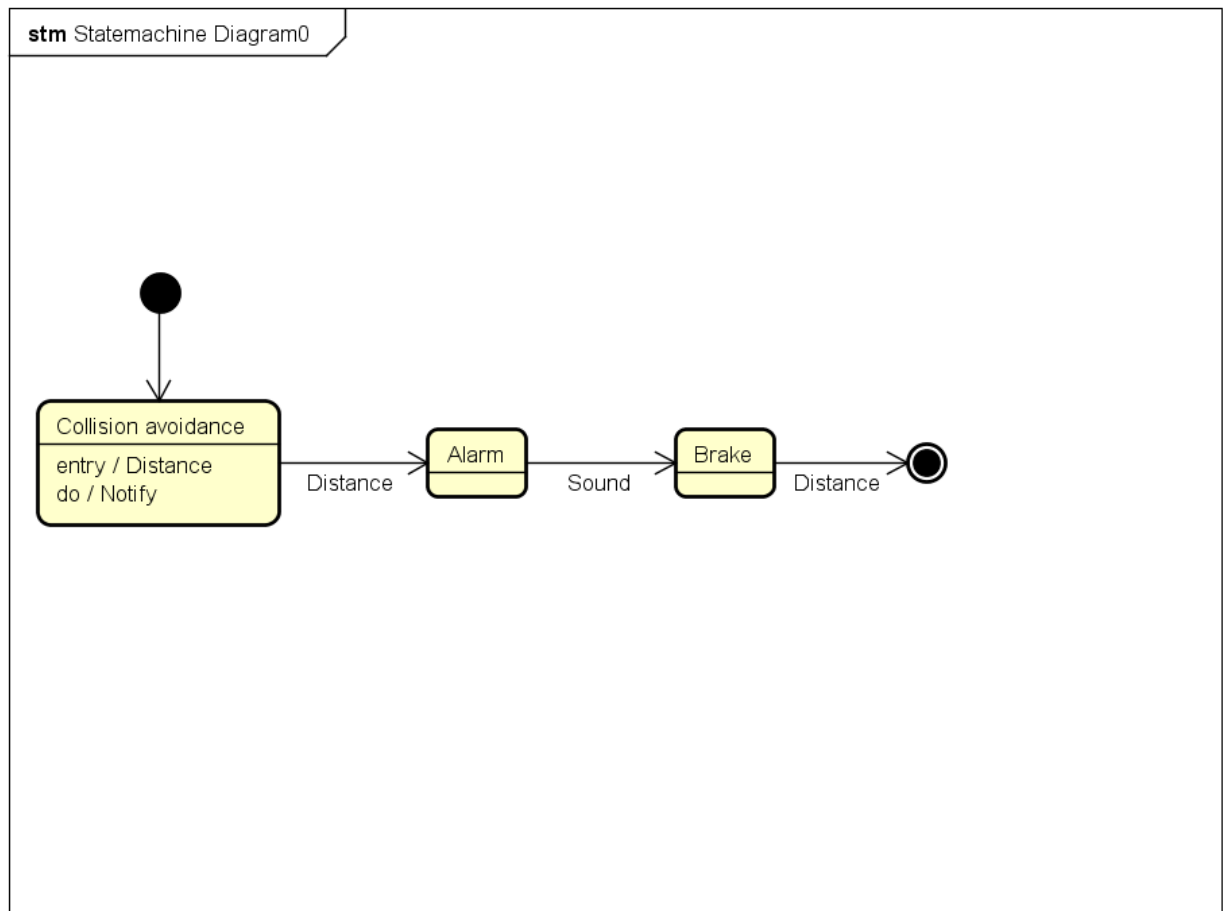
powered by Astah

Figure 5. Air bag statechart diagram



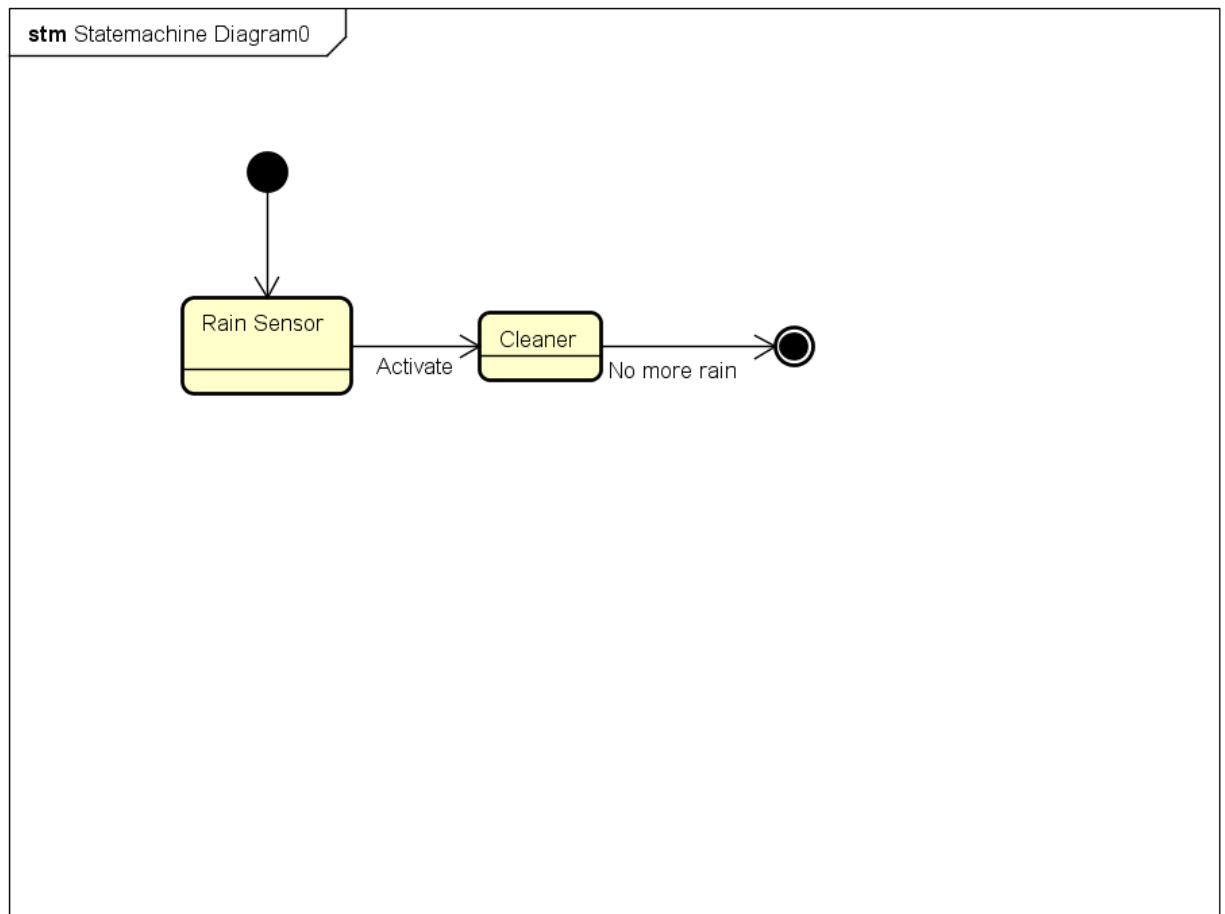
powered by Astah

Figure 6. Steering statechart diagram



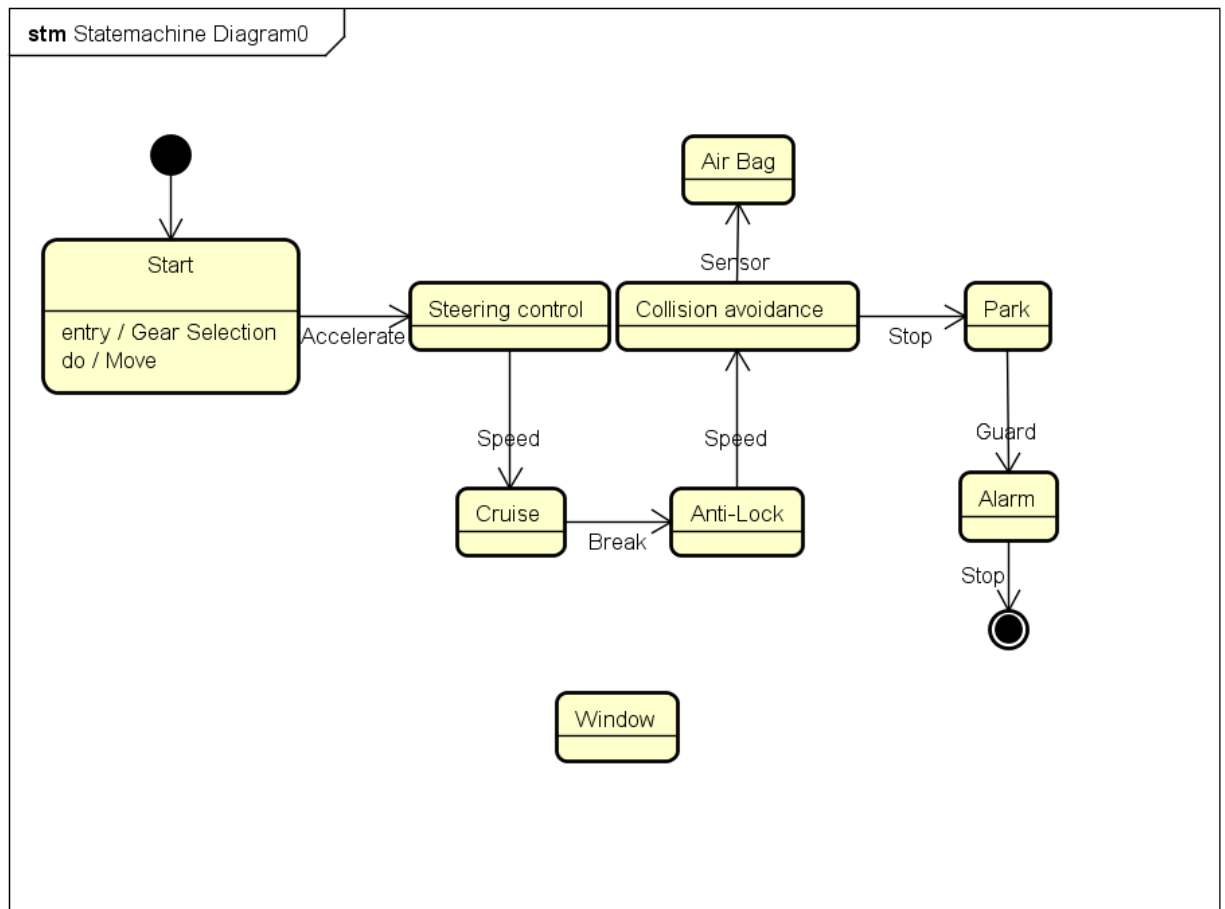
powered by Astah

Figure 7. Parking assistant state chart diagram



powered by Astah

Figure 8. Window cleaner state chart diagram




powered by Astah

Figure 9. General state chart diagram

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

2- Element catalogue

Name Property	Responsibility	Implementation	Relation	Elements
Anti-lock	What is needed in order for a car to brake in the right way.	Needs a controller to hold future wheel changes.	runtime	Wheel speed sensor, ABS logic & control, Modulation Valves ,Brake
Collision Avoidance	Whole system reaction on putting the car in the right track for safety reasons.	Brake and steering needs to be modified by these actions.	runtime	Orientate, turning, Interaction, avoid, traveling
Alarm	The connection of doors, sound, lights shows how to secure the car.	Different states should be taken in consideration.	runtime	Status ,light, sound, controller
Air Bag	Under safety measures, it is a necessity.	Sensor and inflator should exchange information	Run time	Break sensor, air bag holder, air bag
Steering wheel	The connection with power sensors to help the car function	Motor and battery are related	Design time	Battery, power, type, controller
Window cleaner	Applying the rain sensor information for safety	Till no more rain, runtime activated	Runtime	Sensor, cleaner
Cruise control	Throttle control for safety as	Sensor activated	Build time	Controller, Speed sensor, panel


Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

	primary			
--	---------	--	--	--

3- Variability guide

With these state charts diagrams we can see that there are possibilities for different manners to do the same thing. All this depends on the parametrization of the methods. As in this view we in details about the logic of the run time behavior, this will be also our focus: constraints and requirements. In some cases this means more maintenance and parallel versions.

Name/Variation point	Options	Binding time- How it is done
Alarm/personalization	Some user personalization can make variations with including like door lock and different sounds and lights.	Safety critical elements grouped and separated during development (design time). Relation with other modules like ignition system (not in this view) for example may lead to non-predicted behavior.
Cruise/throttle	User can go till a certain speed – constraint control	To be done at build time.
Collision/Distance	User can modify the distance till a certain value – constraint control	To be done at build time.
Parking/alarm	Sound or not	At running time.
Anti-lock	Sensor information about wheel speed and brakes condition will be the main input variables, as this is in accordance with the ABS algorithm. Changes or tests here will be changed according to particular platform and can be treated as timing values (how fast the new system will react). It will be applied to each wheel in parallel.	Design time.
Steering/Wheel type	The integration with the wheels type should be considered in the parameter type. It should be properly correlated with other duties.	Runtime

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

Air Bag/Sensor +inflator	Always active	Runtime
---------------------------------	----------------------	----------------

4- Rationale

We use state chart diagram as a way to describe how a system behaves as we can get some information on the dynamic behavior and events. The events predicted to occur in different states are shown as a diagram that has classes and different states in which its objects may be. The representation can be similar to that of a finite state machine as it illustrates how the behavior evolves. We decided to go in this way as it is easier to answer questions such as:


- Forward and reverse engineering.
- What events to use in order to achieve a state
- Object-event relation
- Responding to an event for an object

The other why behind this diagram is to show the changes of states as imposed by events which can either be internal or external factors. Programmers need to know the states of the class objects and their identification makes the implementation easier. As for the assumptions we can say that this diagram has to be seen in two ways: as a sequence and as independent. This is why we included two versions, as the user may use the car as a normal driving vehicle and also just some of its functionalities. The possible states that the classes and their respective events makes that the transitions somehow show the lifecycle of that class.

Runtime behavior

The behavior of a state, bindings, the times a method is called or a variable is accessed can be seen in the run time. Going with this path, we can try to allocate anomalies and it can be a good starting point for testing. We can also think of the presentation panel as an information collector, and according to later future objectives, it interprets them, also based on subcomponents that are important in the decision making it as a controller.

When the classes' instances and methods will be running, they will be coded in such a way that modifying the default behavior should be easy. This can be achieved may be by adding a method that binds parameters (among classes) a dispatcher and a validator.

Software Engineering 3: Vehicle control system	 MÄLARDALEN UNIVERSITY SWEDEN
Sample Software Architecture Documentation	Laboration 2

Time spent on the assignment: 1.5 day was spent as it required more UML work and the template of the assignment 2 has to be seen from a different prospective.