Eindhoven University of Technology

2IWA0

Automotive Software Engineering

Responsible lecturer: prof. dr. ir. J.F. Groote

Car Lock System Casestudy

Authors:	*Student Numbers:*
Michiel N. de Hoop	1021791
Mario Tilocca	1034263

January 18, 2019

Contents

1 Interactions

The interactions that are relevant to the car lock system are identified and divided into internal actions and external actions. The internal actions are the actions that are performed by the car system itself, and the external actions are the actions that happen or are performed by people or systems other than the car system itself. Table 1 shows the internal actions with their meaning and the external actions with their meaning is shown by Table 2.

Table 1: Table of all internal actions with their meaning.

Internal actions	Meaning
SlowDown	The car is slowing down as a result of pressing the brake.
SpeedUp	The car is speeding up as a result of pressing the throttle.
SendSpeed	The new speed is sent to and synchronized with the controller.
DetectCrash	A crash is detected and this information is sent by the environment and received by the controller.
InitiateAirbag	The airbag deployment is initiated by the controller.
LockSignal	The signal for locking the car is sent by the key and received by the controller.
UnlockSignal	The signal for unlocking the car is sent by the key and received by the controller.
UseHandleFrontInsideSignal	The signal that the inside handle of the front door is pulled, is sent by the front door and received by the controller.
UseHandleFrontOutsideSignal	The signal that the outside handle of the front door is pulled, is sent by the front door and received by the controller.
UseHandleRearInsideSignal	The signal that the inside handle of the rear door is pulled, is sent by the rear door and received by the controller.
UseHandleRearOutsideSignal	The outside handle of the rear door is pulled, is sent by the rear door and received by the controller.
LockCar	All car doors are locked.
UnlockCar	All car doors are unlocked.

Table 2: Table of all external actions with their meaning.

External actions	Meaning
Throttle	The throttle of the car is pressed.
Brake	The brake of the car is pressed.
Crash	A crash of the car is occurred.
DeployAirbag	The airbag of the car is physically deployed.
PressLockButton	The lock button on the key is pressed.
PressUnlockButton	The unlock button on the key is pressed.
UseHandleFrontInside	The inside handle of the front door is pulled.
UseHandleFrontOutside	The outside handle of the front door is pulled.
UseHandleRearInside	The inside handle of the rear door is pulled.
UseHandleRearOutside	The outside handle of the rear door is pulled.
OpenCarDoorFront	The door at the front of the car is opened.
OpenCarDoorRear	The door at the rear of the car is opened.
PutChildSafetyLockOn	The child safety lock is put on.
PutChildSafetyLockOff	The child safety lock is put off.

2 Structure

In this section the compact architecture of the structure of the system is described. The responsibilities of each process and component are explained and the diagram that visualizes the system, is elucidated. The flowchart of the architecture of the system is shown in Fig. 1

In the flowchart the actions of the system are represented by the oval shapes and the components of the system are represented by the squares. The actions that have no incoming arrow and only an outgoing arrow ending up at a component are external actions that are mostly done by humans or by things that cannot be controlled such as Crash. These actions are Throttle, Brake, Crash, UseHandleFrontInside, UseHandleFrontOutside, UseHandleRearInside, UseHandleRearOutside, PressLockButton, PressUnlockButton, PutChildSafetyLockOn, and PutChildSafetyLockOff.

Also, the actions that only have an incoming arrow originating from a component are external actions. These actions are DeployAirbag, OpenCarDoorFront, and OpenCarDoorRear.

Furthermore, the actions that have an incoming arrow and an outgoing arrow are the internal actions and connect the components with each other. These internal actions, however, stand for two combined actions, namely a sent signal and a received signal. For example, SendSpeed is the combined action of the action SendSpeedS, which represents that the signal is sent by the environment, and the action SendSpeedR, which represents that the signal is received by the controller. This pattern is also used for the actions UseHandleFrontInsideSignal, UseHandleFrontOutsideSignal, UseHandleRearInside, UseHandleRearOutside, UnlockSignal, LockSignal, InitiateAirbag, DetectCrash.

The only other internal actions that are not a combination of two other actions, are LockSignal and UnlockSignal. The reason for this is that these actions are done by the Controller component and are fed back to this same component. In the diagram, these two actions are, therefore, connected with the controller with lines with arrows on both sides.

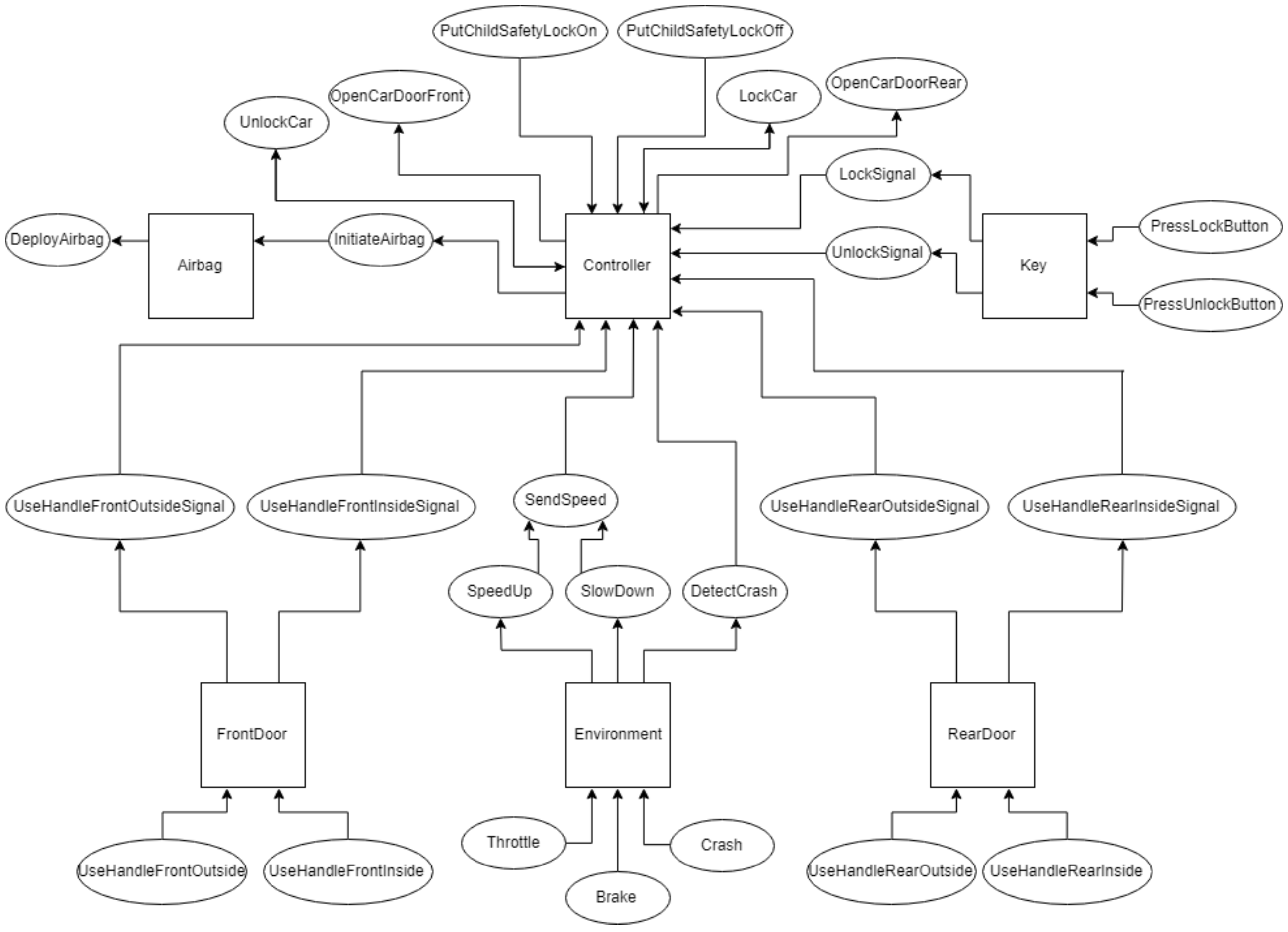

Figure 1: Flowchart of the architecture of the system.

3 Implemented Actions in MCRL2

As it is shown in Fig.1 the controller acts as the central brain of the system. In addition to the actions which can be observed in Fig.1, three internal states are implemented in the controller: the first one of type *speed* which is used in order to update the controller regarding the current speed of the vehicle, (which can be *stationary*, *slow* or $fast$),the second and third are of type *Bool* and they concern whether the car is locked or unlocked,*locked*, and if the child safety lock is activated or if it is not, *childlocked*.
Whenever the throttle or the brake is pushed there is a change of velocity of the vehicle, if the new speed parameter received by the controller is equal to $fast$, meaning that the vehicle is traveling faster than 30 km/h, the action LockCar takes place in the controller and the *Bool* parameter *locked* is set to true.

In case a Crash happens the signal DetectCrash will be sent to the controller from the environment to the controller, consequently the controller will communicate to the airbags to be deployed within a finite amount of time, while all car doors will be unlocked, *locked* Bool parameter is set to $false$, and the child safety will be permanently disabled in order to allow the rear doors to be opened at any time.
Furthermore whenever the PressLockButton or PressUnlockButton are used the signal to lock or unlock the doors is sent from the key and received by the controller and according to the case the parameter *locked* is either set to *true*, the car is locked, or $false$, the car is unlocked.

Whenever the vehicle is not stationary and not locked, it is always possible to open the front doors of the vehicle after the controller has received the signal from the front door that the handle has been used. Conversely in case the car is stationary and *locked* is set to *true*, if the inside handle in the front is used the signal is sent to the controller which will unlock the car and open the front door is possible, *locked* is no longer *true* but it is changed to $false$; Only in case *locked* is $false$ and the vehicle is stationary in case the outside front handle is used the front door of the car will be opened.

For what concerns the rear doors in case *childlocked*, and *locked* are $false$ and the speed of the vehicle is not stationary, if the inside rear door handle is used the controller will allow the rear door to be opened. Conversely if the child safety is off and the car is stationary and was previously locked, by pulling the inside rear handle the controller will unlock the car and it will allow the rear door to be opened, at the same time the internal state *locked* will be changed to $false$. In case the car was previously unlocked it will be possible to open the rear doors by pulling the rear outside handle, no change needs to be carried out in the three internal states.
Furthermore, whenever the PutChildSafetyLockOn action occurs the internal state *childlocked* will be set to *true*, while in case the PutChildSafetyLockOff happens *childlocked* will be set to $false$.

4 Requirements and Validation

1. After one action is executed you can always do another action afterwards.

   For the aforementioned requirement the following μ calculus formula was used:

$$[true*] < true > true \tag{1}$$

2. After any sequence of actions, if any sequence of actions in which UseHandleFrontOutside does not occur, OpenCarDoorFront can happen.
   After any sequence of actions, if any sequence of actions in which UseHandleRearOutside does not occur, OpenCarDoorRear can happen.

   The following μ calculus formula was used in order to meet this requirement:

$$\begin{aligned}&[true*] <!UseHandleFrontOutside * .OpenCarDoorFront > true\\ \&\&&[true*] <!UseHandleRearOutside * .OpenCarDoorRear > true\end{aligned} \tag{2}$$

3. After any sequence of actions, when PressLockButton happens, and LockCar has not happened yet, then LockCar can happen.
   After any sequence of actions, when PressUnlockButton happens, and UnlockCar has not happened yet, then UnlockCar can happen.

   The following *mu* calculus formula was used to achieve this requirement:

$$\begin{aligned} [true * .PressLockButton.!LockCar*] < true * .LockCar > true \\ \&\&[true * .PressUnlockButton.!UnlockCar*] < true * .UnlockCar > true \end{aligned} \tag{3}$$

4. After any sequence of actions, when SendSpeed(fast) happens and LockCar has not happened yet, then LockCar can happen if any sequence of actions in which PressLockButton does not occur happen.

   This requirement was tested according to:

$$[true * .SendSpeed(fast).!LockCar*] <!PressLockButton * .LockCar > true \tag{4}$$

5. After any sequence of actions, when SendSpeed(stationary) happens and UnlockCar has not happened yet, UnlockCar can happen if UseHandleFrontInside or UseHandleRearInside has happened.

   The following *mu* calculus formula was implemented in order to validat the aforementioned requirement:

$$\begin{aligned} [true * .SendSpeed(stationary).!UnlockCar*] \\ < true * .(UseHandleFrontInside||UseHandleRearInside).UnlockCar > true \end{aligned} \tag{5}$$

6. After any sequence of actions in which Crash does not occur, When PutChildSafetyLockOn happens and PutChildSafetyLockOff, UseHandleRearOutsideSignal, and Crash has not happened yet, then OpenCarDoorRear can never happen.

   In order to validate this requirement, the following *mu* calculus formula is used:

$$\begin{aligned} [!Crash*][PutChildSafetyLockOn \\ .!(PutChildSafetyLockOff||UseHandleRearOutsideSignal||Crash) * . \\ OpenCarDoorRear]false \end{aligned} \tag{6}$$

7. After any sequence of actions, when Crash happens, and DeployAirbag has not happened yet, then DeployAirbag can happen.

   This requirement was translated to *mu* calculus as following:

$$[true * .Crash.!DeployAirbag*] < true * .DeployAirbag > true \tag{7}$$

8. When any sequence of actions in which Crash has not happened, DeployAirbag can not happen.

   The aforementioned requirement was translated into *mu* calculus as following:

$$[!Crash * .DeployAirbag]false \tag{8}$$

9. After any sequence of actions, when Crash happens and after that InitiateAirbag happens, OpenCarDoorFront can happen after UseHandleFrontInside or UseHandleFrontOutside has happened. At the same way OpenCarDoorFront can happen after UseHandleFrontInside or UseHandleFrontOutside has happened.
   After any sequence of actions, when Crash happens and after that InitiateAirbag happens, OpenCarDoorRear can happen after UseHandleRearInside or UseHandleRearOutside has happened. At the same way OpenCarDoorRear can happen after UseHandleRearInside or UseHandleRearOutside has happened.

$$\begin{aligned}&[true * .Crash.InitiateAirbag]\\ &< (UseHandleFrontInside||UseHandleFrontOutside).OpenCarDoorFront > true\\ &\&\&[true * .Crash.InitiateAirbag]\\ &< (UseHandleRearInside||UseHandleRearOutside).OpenCarDoorRear > true\end{aligned} \tag{9}$$

10. After any sequence of actions in which UseHandleFrontInside and UseHandleRearInside does not occur, when LockCar happens and UnlockCar, UseHandleFrontInside and UseHandleRearInside has not happened yet, OpenCarDoorFront and OpenCarDoorRear can not happen.
    After any sequence of actions in which SendSpeed(fast) does not occur, when UnlockCar happens and LockCar and SendSpeed(fast) has not happened yet, OpenCarDoorFront or OpenCarDoorRear can happen.

    This requirement was tested according to:

$$\begin{aligned}&[!(UseHandleFrontInside||UseHandleRearInside)\\ &*.LockCar.!(UnlockCar||UseHandleFrontInside||UseHandleRearInside)\\ &*.(OpenCarDoorFront\&\&OpenCarDoorRear)]false\\ &\&\&[!SendSpeed(fast) * .UnlockCar.!(LockCar||SendSpeed(fast))*]\\ &< true * .(OpenCarDoorFront||OpenCarDoorRear) > true\end{aligned} \tag{10}$$