AltaRica 3 Based Models for ISO 26262 Automotive Safety Mechanisms

Abraham Cherfi[1, 2], Antoine Rauzy[3], and Michel Leeman[2]

[1] LIX - Ecole Polytechnique, route de Saclay, 91128 Palaiseau cedex, France
name@lix.polytechnique.fr
[2] GEEDS - Valeo, France
{firstname.lastname}@valeo.com
[3] Chaire Blériot-Fabre - Ecole Centrale de Paris, Grande Voie des Vignes, 92295 Châtenay-Malabry, France
{firstname.lastname}@ecp.fr

Abstract. Cars embed a steadily increasing number of Electric and Electronic Systems. The ISO 26262 defines a number of constraints, rules and requirements that the development of Automotive E/E Systems must obey in order to guaranty their Functional Safety. One of the means at hand to enhance the safety of these systems is to reinforce them with so-called Safety Mechanisms. The Standard discusses at length how to estimate the contribution of these mechanisms to Functional Safety. These calculations rely however on Fault Tree models or ad-hoc formulas that are hard to check for completeness and validity. In this article, we propose generic AltaRica 3 for Electric and Electronic Systems protected by first and second order safety mechanisms. These models are of a great help to clarify the behavior of these systems as well as to determine the domain of validity of simpler models such the above mentioned Fault Trees or ad-hoc formulas.

Keywords: Automotive Functional Safety, ISO 26262, Safety Mechanisms, AltaRica, Markov Models.

1 Introduction

Cars embed a steadily increasing number of Electric and Electronic Systems. In order to guaranty their Functional Safety, the ISO 26262 standard [1] was published in November 2011. This standard defines a number of constraints and rules that the development of automotive Electric and Electronic Systems must obey. One of the means at hand to enhance the safety of Electric and Electronic Systems is to reinforce them with so-called Safety Mechanisms. Safety Mechanisms are various types of devices that typically prevent spurious usages of the system, or warn the driver when something wrong happens.

The ISO 26262 standard discusses at length the use of these Safety Mechanisms and how to estimate their contribution to functional safety. To do so, it relies essentially on Fault Tree models or ad-hoc formula. Such models or formulas are indeed of

F. Ortmeier and A. Rauzy (Eds.): IMBSA 2014, LNCS 8822, pp. 123–136, 2014.

interest for practitioners, but they are only approximations. Without a more explicit representation of failure scenarios to serve as a reference, it is hard to check them for completeness, and understand their domain of validity and to ensure their accuracy. Explicit models have been proposed by several authors for Safety Instrumented System described in the mother IEC 61508 Standard [2] (see e.g. [3, 4]). In the case of the ISO 26262 standard, at least to our knowledge, this work has not been done yet.

The remainder of this article will be organized as follows:

First, in order to introduce the different types of safety mechanisms, we will present in Section 2 two typical examples of critical automotive systems. The first example will be based on the control of the Electric Motor Inverter with its detection based safety mechanisms, and the second example will be based on an Electric Seats Control Unit with its inhibition based safety mechanism. This will help us explains their behavior, and their influence on the systems implementing them.

Next to this, in Section 3, we will present our previously established Markov models [7] for the representation of the automotive safety mechanisms behaviors that will help us to explain the various parameter that must be taken into account.

After that, in Section 4, we will present the corresponding AltaRica 3 models for the previously defined examples. This will also help us to introduce more general models for the representation of these safety mechanisms. To finish, we discuss in the same section, the reachability graphs of the two models presented above, and compare them to the unfolded Markov models.

2 Two Typical Examples of Safety Mechanisms

In this section, we present two representative examples of automotive systems embedding safety mechanisms.

2.1 Vehicle Management Unit for Inversion

We shall first consider the case of a Vehicle Management Unit (VMU). In an electric vehicle, a VMU is responsible for commanding the electric motor inverter, among other functions. A VMU consists of a microcontroller which, given certain inputs (gas and brake pedal positions), sends a torque set-point to the inverter that in turn commands the electric motor (traction and regenerative braking), as illustrated in Figure 1. Such a VMU is a critical function: if the microcontroller gets stuck in a loop and continuously sends a command higher (or lower) than expected, it could lead to unintended vehicle acceleration or braking.

In order to prevent such hazards, a watchdog is added which is in charge of bringing the system to a safe state in case the microcontroller is detected to be stuck. The watchdog is an electronic component that is used to detect and recover from microcontroller malfunctions. The microcontroller refreshes regularly the watchdog in order to prevent him from timing out. If it gets stuck in a loop, the watchdog cannot be reset, so the watchdog times out and sends a reboot order to the microcontroller. Such a watchdog is a first order safety mechanism based on error detection.

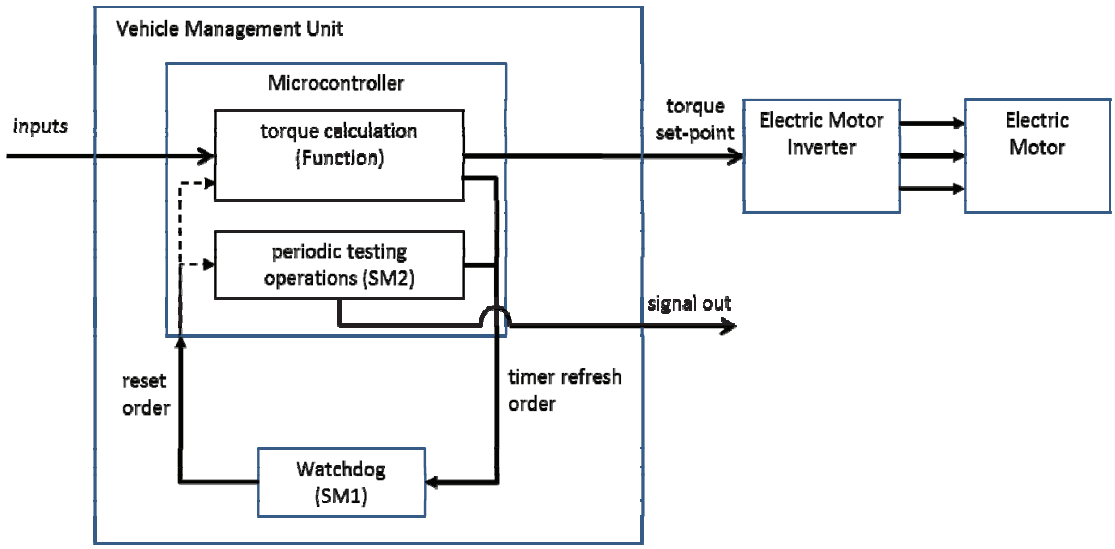

Fig. 1. Simplified functional representation of the Vehicle Management Unit for Inversion

As a physical component, the watchdog may fail (although the reliability of the watchdog is much higher than the one of the microcontroller). Also, the watchdog is able to detect only certain kind of errors of the microcontroller: typically, it is not able to detect memory corruption problems.

In order to ensure that the watchdog is working, the microcontroller tests the watchdog at each vehicle start (when the ignition is set on). The role of this second order mechanism is to warn the driver in a case of a problem with the watchdog. It may itself fail and is itself not able to catch all of the problems of the watchdog.

As the torque calculation function and the second order safety mechanism function are never executed in parallel, their failures are considered as independent (and are independent from watchdog failures).

The above example is representative of safety mechanisms based on error detection as embedded for instance in electric steering column controller, electric braking, several types of microcontrollers protected with watchdogs and more generally command-control systems.

2.2 Electric Driver Seats Control

Another type of safety mechanism is used in Electric Driver Seat Controls (EDSC). An EDSC allows the driver to tune his seat position. A spurious tuning action while the vehicle is running (over a certain speed, e.g. 10km/h) can indeed cause an accident, for instance because the driver is no longer able to reach the brake pedal or because he gets suddenly pushed onto the steering wheel.

In order to prevent this from happening, the system embeds a mechanism in charge of turning off the power supply of the EDSC when the vehicle is running. This first order mechanism is therefore based on inhibition. As previously, it is in general completed with a second order one in charge of testing it at each vehicle start (obviously, it cannot be tested while the vehicle is running).

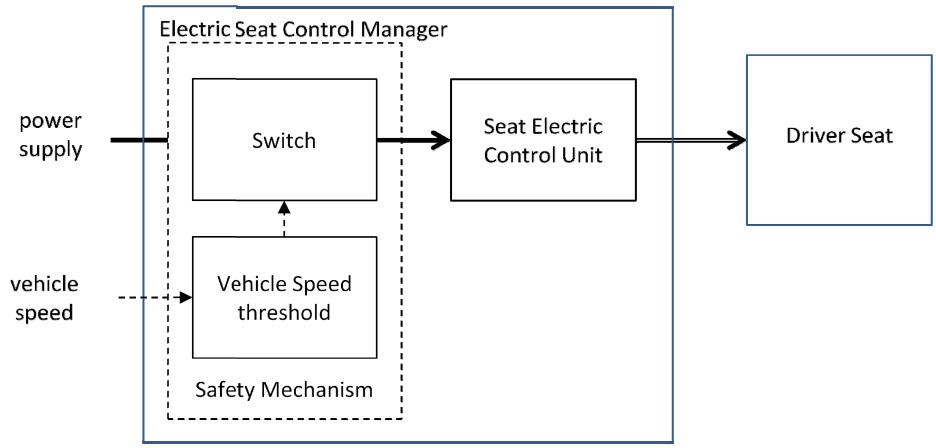

Fig. 2. Functional representation of an Electric Driver Seat Control

The above mechanism is representative of safety mechanisms based on inhibition, as embedded for instance in Electric Steering Column Lock, Automatic Doors opening systems and more generally all systems that must be inhibited when the speed of the vehicle gets above a given threshold.

2.3 Discussion

The implementation of the safety mechanisms presented in this section is a practical way to enhance the automotive systems safety without expensive physical redundancy. The majority of automotive first order safety mechanisms can be actually categorized in either of the two categories presented above:

- Most of them are based on error detection. The idea is to switch the system into a safe state when an error is detected. These safety mechanisms are usually made of two elements: the detection device and the actuation device.
- Some of them inhibit the system they protect when the vehicle is in a state where the failure of the system is potentially dangerous.

As a failure of the first order safety mechanism has in general no direct influence on the system it controls, it can hardly be perceived by the driver. A second order safety mechanism is thus often added in order to check periodically the availability of the first one, typically when the engine is turned on or the vehicle starts to move. The role of such a second order mechanism is to warn the driver.

3 Generic Markov Models

To have a clear understanding of the behavior of Electric and Electronic Systems in presence of failures (including those of safety mechanisms), the best method is probably to design state/transition models for these systems. As Markovian hypotheses can be verified or are at least approximated for calculation purposes, these models can be turned into Markov chains in a straightforward way.

In this section, we shall propose Markov chains for systems of each of the two above categories. These Markov chains are generic in the sense that one has just to

adjust values of parameters (such as failure rates, coverage rates…) to assess the safety of a particular system. Markov chains presented hereafter can be subsequently embedded into larger Markov models or approximated either by means of Fault Tree constructs or by ad-hoc formulas. They serve as a reference.

3.1 Case of a Hardware Block Protected by a First Order Safety Mechanism Based on Detection

Let us consider first the case of a Hardware block HB protected by a first order safety mechanism SM1 based on error detection. The generic Markov chain for this system is given in Figure 3.

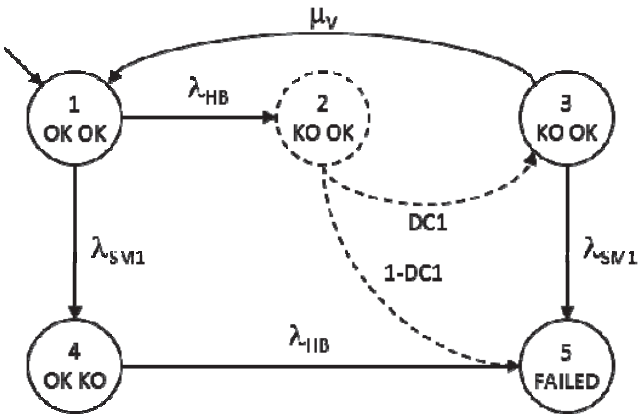

Fig. 3. Generic Markov Chain for a Hardware Block protected by a first order Safety Mechanism based on error detection

Such a system fails in a dangerous state if both the hardware block and the safety mechanism fail, no matter in which order. Therefore, the Markov chain encodes basically three failure scenarios.

In the initial state (1), both the hardware block and the safety mechanism are working properly. The failure rates λ_{HB} for the hardware block and λ_{SM1} are assumed to be constant over the time (no ageing effect). If the hardware block fails first, the system goes to state 2, where the safety mechanism detects or not this failure instantaneously. As a graphical convention, we denote instantaneous states and their outgoing probabilities by dashed lines, as on the figure. The probability not to detect the failure is 1-DC1, where DC1 stands for the diagnostic coverage of the safety mechanism. In the state (2), if the failure of the hard block is not detected the system goes to the failure state (5) (first failure scenario). Otherwise, it goes to the safe state (3). In this state, the mean time before the vehicle is taken to the garage is T_M, i.e. the repair rate of the hardware block is $\mu_V = 1/T_M$. Now, if the safety mechanism fails before the vehicle is repaired, then the system goes to the failure state (5) (second failure scenario). Otherwise it goes back to the initial state (1).

Finally, if, in the initial state, the safety mechanism fails before the hardware block fails, then the system goes to state (4). In this state, we have nothing to do but to wait until the hardware block fails to go into the failure state (5) (third failure scenario).

Note that since there is no mean to detect a failure of the safety mechanism, there is no mean to repair it neither. Moreover, we assume that neither the hardware block nor the safety mechanism are inspected during periodic maintenances of the vehicle. This hypothesis is realistic, although pessimistic.

3.2 Case of a Hardware Block Protected by a First Order Safety Mechanism Based on Detection and a Second Order Safety Mechanism

We shall consider now the case of a hardware block HB protected with a first order safety mechanism SM1 based on error detection which is itself tested by a second order safety mechanism each time the vehicle starts. The generic Markov chain for such a system is given Figure 4.

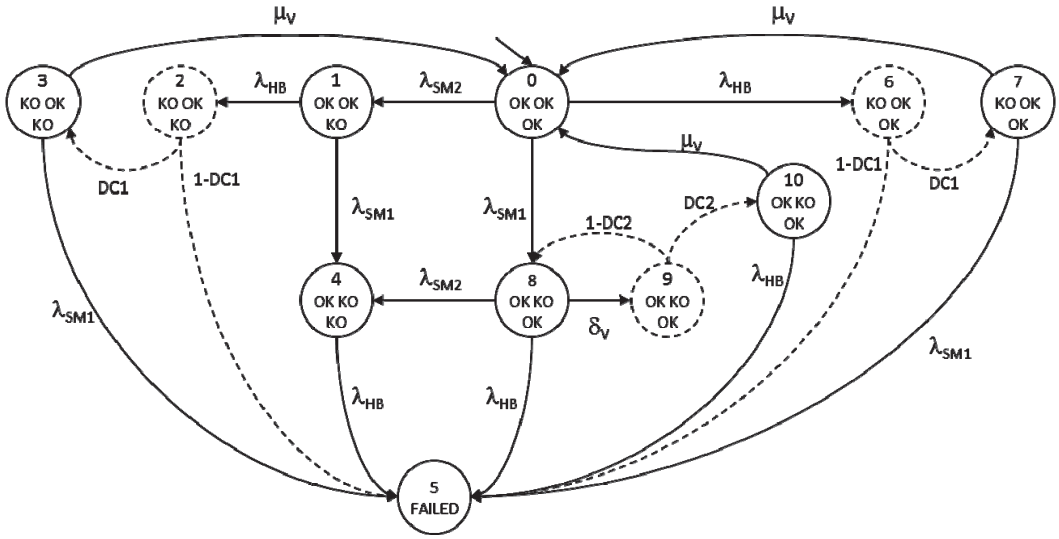

Fig. 4. Generic Markov Chain for a Hardware Block protected by a first order Safety Mechanism based on error detection and a second order Safety Mechanism

This model extends the previous one. The second order mechanism has its own failure rate λ_{SM2} as well as its own diagnostic coverage DC2. Note that it is assumed that when the vehicle is taken to the garage, it is fully repaired and is as good as new after this repair.

In the initial state (0), the hardware block HB and the two safety mechanisms SM1 and SM2 are assumed to work correctly. Now there are three possibilities:

- The second order mechanism fails first. In that case, according to our hypotheses, we are exactly in the same situation as if there was no second order mechanism. So the model obeys the same pattern as previously. We kept actually the same numbering of states 1 to 5 to emphasize this point.
- The hardware block fails first. This situation is also very similar to the previous one, for the second order mechanism plays no specific role in the subsequent scenarios. State 0, 6 and 7 are therefore symmetric to states 1, 2 and 3. The only difference stands in the availability of the second order mechanism.

- The interesting scenarios are therefore those where the first safety mechanism fails first, i.e. the system goes to state 8. We shall now develop these scenarios.

In state 8, we are in the situation where the first order safety mechanism failure is unnoticed. Here again there is a race condition amongst three possibilities:

- The hardware block fails first, before the current journey ends. In that case, the whole system fails (state 5).
- The second order safety mechanism fails first. In that case, we can make the pessimistic assumption that the driver did not notice the warning before this failure. So, we are back to the situation where there is no second order safety mechanism (and the first order one is failed), i.e. to state 4.
- The current journey ends before both the hardware block and the second order mechanism fail (state 9). We can assume that the mean time before the journey ends is Tj so that the transition rate between states 8 and 9 is $\delta_V = 1/T_J$. Now at the next start of the vehicle, the second order mechanism tests the first order one with a probability DC2 of successful detection. If the detection is successful (state 10) then either the driver takes the vehicle to the garage before the hardware block fails (in which case the system goes back to the initial state 0) or the hardware block fails first (in which case the whole system fails, i.e. goes to state 5). If the second order mechanism does not detect the failure of the first order one, then we have to wait for another start of the vehicle to make the test again (so the system goes back to state 8)

It is worth to note that the model described here is quite different from those proposed for Safety Instrumented Systems in references [3, 4]. The difference stands mainly in assumptions about the maintenance policy. As already pointed out, the designer of an automotive Electric and Electronic system has no control on maintenance. So, he has to make pessimistic hypotheses about what the driver will (reasonably) do.

3.3 Case of a Hardware Block Protected by a First Order Safety Mechanism Based on Inhibition and a Second Order Safety Mechanism

We shall now consider the case of a hardware block HB protected with a first order safety mechanism SM1 that inhibits the hardware block functionality, itself periodically tested by a second order safety mechanism SM2. The generic Markov chain for such a system is given in Figure 5. As the reader has immediately noticed, this model is embedded in the previous one. The reason is that if the hardware block fails before the first order safety mechanism, then there is nothing to inhibit and the system is safe (but of course not available).

Note also that there is no detection device and therefore no diagnostic coverage for the first order safety mechanism.

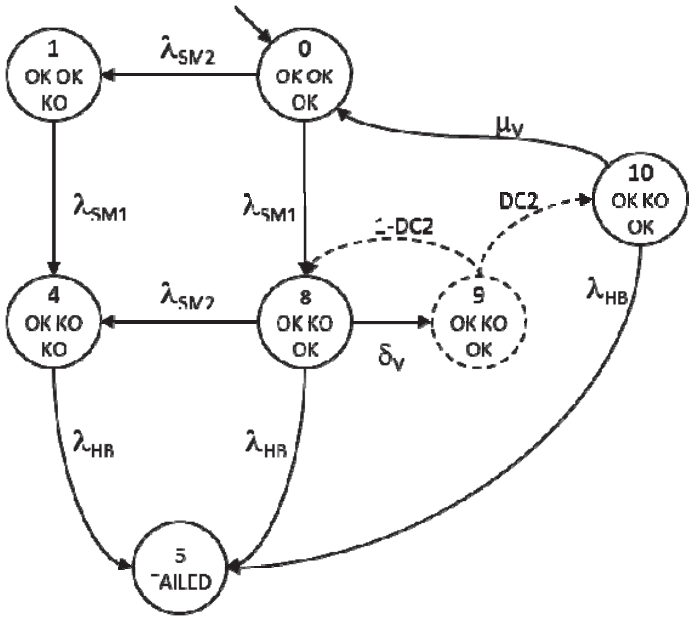

Fig. 5. Generic Markov chain for a Hardware Block protected by a first order Safety Mechanism based on inhibition and a second order Safety Mechanism

4 AltaRica 3.0 Models

In this section we present AltaRica 3 models for the representation of automotive safety mechanisms. One of the main objectives behind this is to be able to define generic classes for the different objects that are handled during the safety analyses and which allow the representation of their dysfunctional behavior.

AltaRica 3 is the latest evolution of event based modeling language "AltaRica" [8]. In this Language, the state of the system is described by means of variables, so, the modification of the system state can only happen when its variables values change. Also, the value of these variables can only change when an event is triggered.

Events can be associated with deterministic or stochastic delays. Models consist of hierarchical components interfaced in a discrete event system: their inputs and outputs can be connected and their transitions can be synchronized.

In the following sub sections, we will use the previously presented examples (Section 2), to explain our implementation of the automotive safety related components.

4.1 AltaRica 3 Models for the Vehicle Management Unit for Inversion

In this subsection, we will describe, element by element how to represent dysfunctional behavior of the VMU for inversion.

So, to begin, each of the considered objects (functional blocks and safety mechanisms) can have two states: working and failed. So, we create the corresponding variable type:

```
domain HardwareStatus{WORKING, FAILED}
```

We also know that each object can be subject to failure when it is working, and is considered as repaired after maintenance. Both maintenance and failure are considered as events, and are represented in our models by transitions with the same name.

With this in mind, we can already obtain the AltaRica model for a generic Hardware Block :

```
class HardwareBlock
  HardwareStatus self (init = WORKING);
  Boolean failed (reset = false);
  Boolean failureDetected (reset = false);
  Boolean safeMode (reset = false);
  event failure (delay = exponential(lambda));
  event maintenance;
  transition
    failure : self == WORKING -> self := FAILED;
    maintenance : true -> self := WORKING;
  assertion
    failed := self == FAILED;
    safeMode := failed and failureDetected;
end
```

We also have three Boolean variables for external communication:

- "failed", that allows to read the state of the hardware block (for examples In Section 2.1, when the refresh orders are not sent to the watch dog, this variable is set to "true").
- "safeMode", that allows to see if the hardware block is in safe mode. As defined in Section 2, the safe mode is engaged when the hardware block has failed ("failed" set to true) and its failure is contained by the first order safety mechanisms ("failureDetected" set to true).
- "failureDetected", is an input variable that allows to indicate if a failure is currently detected by the first order safety mechanism (this correspond to the restart order sent in the example Section 2.1).

In addition to those, a detection based first order safety mechanism must be able to detect and contain the failure of its associated hardware block.

In order to take into account the diagnostic coverage and its associated probabilities of detection we create two concurrent instantaneous transitions (as only one can be triggered at a time): One for the good detection of the hardware block failure and another for its wrong detection.

```
domain FaultStatus{DETECTED, PROPAGATION, NONE}
class DetectionBasedSafetyMechanism
  FaultStatus faultStatus (init = NONE);
  HardwareStatus self (init = WORKING);
  Boolean failed(reset = false);
  Boolean inputSignal (reset = false);
  Boolean sendSafeModeOrder (reset = false);
  event failure (delay = exponential(lambda));
  event goodDiagnostic(delay = 0, expectation = DC);
  event wrongDiagnostic(delay = 0, expectation = 1 - DC);
  event maintenance;
  transition
```

```
    failure : self == WORKING ->
        {faultStatus := PROPAGATION; self := FAILED;}
    goodDiagnostic: self == WORKING and inputSignal and
        faultStatus == NONE -> faultStatus := DETECTED;
    wrongDiagnostic: self == WORKING and inputSignal and
        faultStatus == NONE -> faultStatus := PROPAGATION;
    maintenance: true -> {faultStatus := NONE;
        self := WORKING;}

  assertion
    failed := self == FAILED;
    sendSafeModeOrder := faultStatus == DETECTED;
end
```

We consider that the first order safety mechanism allows fault propagation in two cases: either the hardware block failure has not been detected (wrong diagnostic), or, the safety mechanism has failed, so it is impossible to stop the hardware block failure propagation.

By following the same logic, we can design the second order mechanism behavior:

```
domain SignalisationStatus {SIGNALED, NONE}
class SecondOrderSafetyMechanism
  FaultStatus faultStatus (init = NONE);
  HardwareStatus self (init = WORKING);
  SignalisationStatus signalisationStatus (init = NONE);
  Boolean inputSignal(reset = false);
  event failure (delay = exponential(lambda));
  event goodDiagnostic(delay = Td, expectation = DC);
  event wrongDiagnostic(delay = Td, expectation = 1 - DC;
event maintenance;
  transition
    failure : self == WORKING -> self := FAILED;
    goodDiagnostic: self == WORKING and
          faultStatus == NONE and inputSignal ->
          signalisationStatus := SIGNALED;
    wrongDiagnostic: self == WORKING and
          faultStatus == NONE and inputSignal -> skip;
    maintenance: true -> {self := WORKING;
          signalisationStatus := NONE;}
end
```

The main difference between this second order safety mechanism and the previously defined first order safety mechanism are the following:

- The second order safety mechanism does not contain faults, it only signals them, and so, we consider that as long as a fault is signaled, it stays in this status even if the second order safety mechanism fails.
- The diagnostic coverage events are periodically timed and not instantaneous as presented in the detection based first order safety mechanism. This represents the fact that the tests that only occurs at the vehicle start.

Now that we have the classes that correspond to each element in our system, we can combine them in order to represent the dysfunctional behavior of our vehicle management unit (VMU).

```
class VMU
  Microcontroller uc;
  DetectionBasedSafetyMechanism watchDog;
  event repairRequestedBySM1 (delay = tTau);
  event repairRequestedBySM2 (delay = tTau);
  observer HardwareStatus status =
      if(uc.torqueCalculationFunction.failed
      and watchDog.faultStatus == PROPAGATION)
      then FAILED
      else  WORKING;
  transition
    repairRequestedBySM1:
            watchDog.faultStatus==DETECTED -> skip; &
            !uc.torqueCalculationFunction.maintenance &
            !watchDog.maintenance &
            !uc.periodicTestingUnit.maintenance;
    repairRequestedBySM2:
          uc.periodicTestingUnit.signalisationStatus ==
          SIGNALED and
          not uc.torqueCalculationFunction.failed -> skip; &
          !uc.torqueCalculationFunction.maintenance &
          !watchDog.maintenance &
          !uc.periodicTestingUnit.maintenance;
    hide  uc.periodicTestingUnit.maintenance,
          watchDog.maintenance,
          uc.torqueCalculationFunction.maintenance;
  assertion
    watchDog.inputSignal :=
        uc.torqueCalculationFunction.failed;
    uc.periodicTestingUnit.inputSignal :=
        watchDog.failed;
    uc.torqueCalculationFunction.failureDetected :=
        watchDog.sendSafeModeOrder;
end
```

As detailed in Section 2.1, the VMU is composed by the microcontroller and a watchdog. The Microcontroller is in charge of two functions: The torque calculation function "torqueCalculationFunction" and the second order safety mechanism "periodicTestingUnit" in charge of testing the watchdog. We consider that the VMU fails only when the torque calculation function fails and when the watchdog can't contain the propagation.

Also, there are two events that lead to maintenance: either the failure of the torque function is detected by the watchdog or the failure of the watchdog is detected by the second order safety mechanism. This is represented respectively by the transitions "repairRequestedBySM1" and "repairRequestedBySM2".

4.2 AltaRica 3 Models for Electric Driver Seat Control

In this subpart, we present the modifications that are necessary in our previous model in order to be able to represent systems and components with first order safety mechanisms based on inhibition.

The main difference between this model and the previous one is in the type of the first order safety mechanism that is used. Indeed, as described in the section 2.2. This system implements a safety mechanism based on inhibition.

First of all, to realize this implementation, we consider that as long as the first order safety mechanism is active then the associated hardware block is not powered. Thus we change the failure transition of our hardware block by adding this condition:

```
failure : self == WORKING and powered -> self := FAILED;
```

We also must define the safety mechanism based on inhibition:

```
class InhibitionBasedSafetyMechanism
  HardwareStatus self (init = WORKING);
  Boolean failed (reset = false);
  Boolean functionInhibition (init = true);
  event failure (delay = exponential(lambda));
  event maintenance;
  transition
    failure : self == WORKING ->
        functionInhibition := false; self := FAILED;
    maintenance : true ->
        functionInhibition := true; self := WORKING;
  assertion
    failed := self == FAILED;
end
```

As we can see, as long as this safety mechanism is working, it forces the function inhibition by setting its Boolean communication output “functionInhibition” to the value “true”.

For the second order safety mechanism behavior implementation, we use the one that was presented in the previous section.

To finish this implementation, we created the class EDSC that is in charge of combining the previously presented components:

```
class EDSC
  HardwareBlock driverSeatsManager;
  InhibitionBasedSafetyMechanism powerInhibition;
  SecondOrderSafetyMechanism periodicTestingUnit;
  event repairRequestedBySM2 (delay = tTau);
  observer HardwareStatus status =
    if (driverSeatsManager.failed) then FAILED
    else  WORKING;
  transition
    repairRequestedBySM2:
      periodicTestingUnit.signalisationStatus == SIGNALED
      and not driverSeatsManager.failed -> skip; &
      !powerInhibition.maintenance &
      !driverSeatsManager.maintenance &
      !periodicTestingUnit.maintenance;

    hide  periodicTestingUnit.maintenance,
          driverSeatsManager.maintenance,
          powerInhibition.maintenance;
  assertion
```

```
    periodicTestingUnit.inputSignal :=
                powerInhibition.failed;
    driverSeatsManager.powered :=
                not powerInhibition.functionInhibition;
end
```

As we can see, the only condition for the system failure is that the driver seat manager fails. But as the different objects were designed, this event can happen only after the failure of the inhibition based first order safety mechanism.

4.3 Reachability Graphs

Using a stepper, we built the reachability graph for each of the presented AltaRica models.

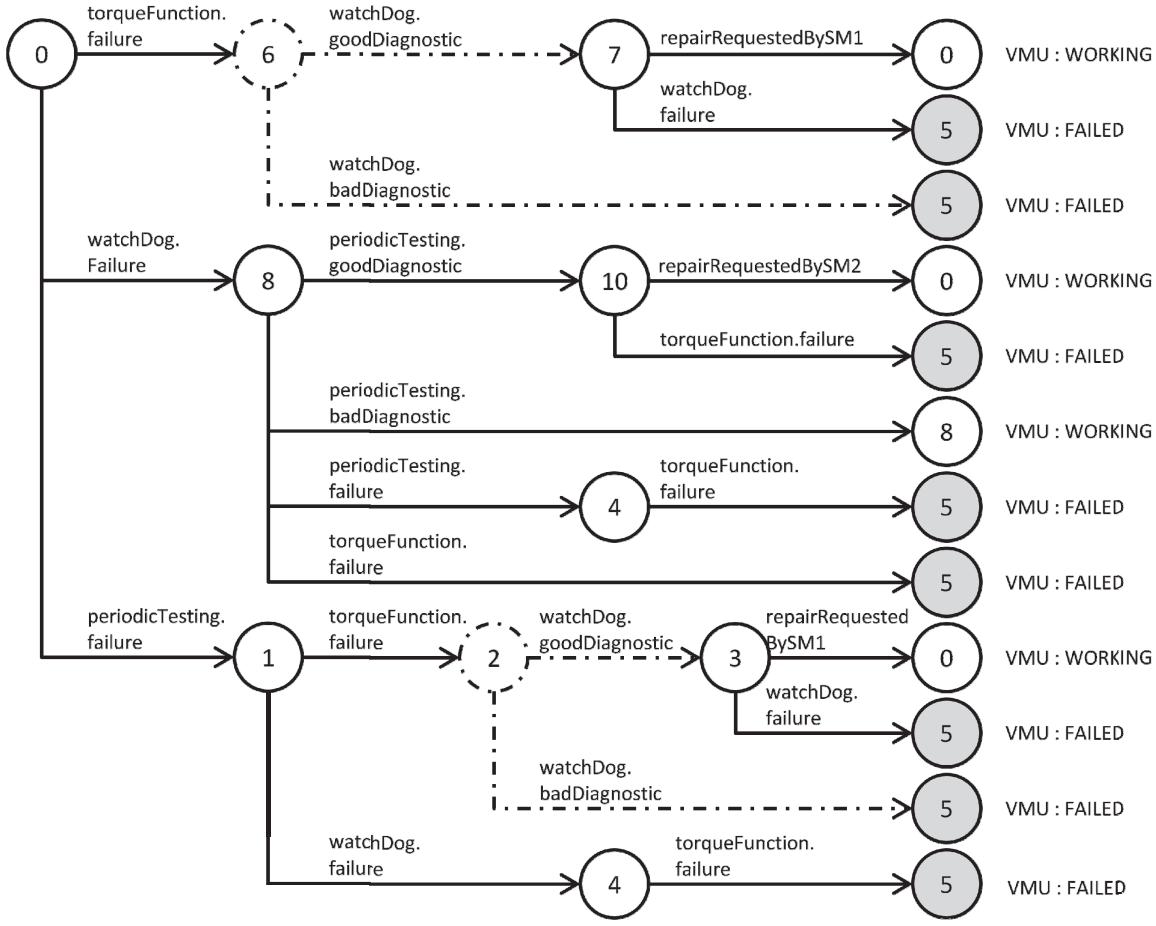

Fig. 6. Reachability graph of the VMU AltaRica 3 Model matched with the unfolded view of the corresponding Markov Model

As we can see in Figure 6, our VMU AltaRica model perfectly matches the Markov model presented in Figure 4. The states numbers in this graphic are the same as the ones presented in the corresponding Markov model. This allows us to say that AltaRica model presented in Section 4.1 provides good implementations of the automotive Safety Mechanisms.

The same work has been done to compare the model proposed in Section 4.2 with the one illustrated in Figure 5. In this case too, the two models perfectly match, allowing us to deduce that they provide good implementations of Automotive Safety Mechanisms.

5 Conclusion

In this article, we proposed AltaRica 3 models for the implementation of the behavior of large classes of automotive Electric and Electronic systems protected by first and possibly second order safety mechanisms. These models are generics in the sense that only few adjustments must be made in order to use them in most of the practical cases.

The modularity of the models that we proposed allows to easily combine them in order to represent large sized systems. Also, given the fact that they can be compiled to Guarded Transition Systems models, the computations on these models can be really fast and accurate (by using the Limited Depth Markov Generation [6]).

References

1. ISO 26262, Road vehicles – Functional safety, Working Group ISO TC22 SC3 (2011)
2. IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems, parts 1–7. Geneva: International Electrotechnical Commission (1998)
3. Innal, F., Dutuit, Y., Rauzy, A., Signoret, J.-P.: New insight into the average probability of failure on demand and the probability of dangerous failure per hour of safety instrumented systems. Journal of Risk and Reliability 224, 75–86 (2010)
4. Jin, H., Lundteigen, M.A., Rausand, M.: Reliability performance of safety instrumented systems: A common approach for both low- and high-demand mode of operation. Reliability Engineering and System Safety 96, 365–373 (2011)
5. Boiteau, M., Dutuit, Y., Rauzy, A., Signoret, J.-P.: The AltaRica Data-Flow Language in Use: Assessment of Production Availability of a MultiStates System. Reliability Engineering and System Safet 91(7), 747–755 (2006)
6. Brameret, P.-A., Rauzy, A., Roussel, J.M.: Preliminary System Safety Analysis with Limited Depth Markov Chain Generation. In: Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013, York, Great Britain (September 2013)
7. Cherfi, A., Rauzy, A., Leeman, M., Meurville, F.: Modeling Automotive Safety Mechanisms: A Markovian Approach, Reliability Engineering and System Safety (accepted in April 2014), doi:http://dx.doi.org/10.1016/j.ress,04.013
8. Prosvirnova, T., Batteux, M., Brameret, P.-A.: The AltaRica 3.0 project for Model-Based Safety Assessment. In: DCDS 2013, York, Great Britain (September 2013)