ENSURING HIGH QUALITY IN SPECIFICATIONS FOR AUTOMOTIVE EMBEDDED CONTROL SYSTEMS

Eva Rakotomalala,[1] Jean-Pierre Elloy,[2] Pierre Molinaro, [2] Bernard Bavoux,[1] and Didier Jampi[1]

[1] *PSA Peugeot Citroen, 2 route de Gisy 78943 Velizy-Villacoublay cedex, France;*
[2] *IRCCYN Ecole Centrale Nantes, 1, rue de la Noe 44000 Nantes, France*

Abstract: Achieving confidence in safety, and robustness of complex systems is a key issue for an automotive manufacturer. Specifications are the first and crucial stage of the engineering process. The aim is to provide a high level of quality assurance for the specifications of systems incorporing several reused parts. This paper presents a method using an external modeling of the function supervision. We propose a framework based on modes analysis and a formal operation which allows to combine automaton descriptions of modes by adding logic commutations between modes.

Keywords: Formal specification, Embedded Systems, Safety, Verification

1. INTRODUCTION AND CONTRIBUTIONS

The automotive industry is quickly increasing the complexity of its new vehicle designs. Adding functionality or coupling between existing functions makes it more difficult to have high confidence in specifications definition and completeness, then in implementation correctness. The dysfunctionning of an electronic functionality which can occur during operational use of vehicle, can have 3 off-line origins and 3 in operation origins. Off-line, errors which cause dysfunctionning can be due to: 1- incomplete specifications which do not describe all the properties of safety and liveness, 2- incomplete checks which do not test the specification in every situation, 3- implementation errors. In operation, the dysfunctionning can be due to: 4- errors caused by failures in the device (operating system, protocol, hardware), 5- the reception by the functionality of sequences of incoherent signal data or events (generally emitted by other inconsistent or failing functionalities) , 6- the reception by the functionality of sequences of data or events, of which occurrences were not considered in the specifications.

- The treatment of dysfunctionning due to situation 1 is related to system engineering, which steps establish all the requirements that the functionality to be developed must meet. These requirements are built according to the sequences of signals which the functionality can receive from its environment. These sequences are supposed all known. If the environment changes, if the functionality is reused in another environment or if other functions (inter-system) generate sequences of additional input signals, the requirements can be incomplete.

- The treatment of dysfunctionning due to situations 2 and 3 is related to techniques which principles are from now on well-known: model based engineering, validation of these models by model checking (Alur et al., 1993) or theorem proving (Boyer and Moore, 1984), and finally automatic generation of code from this validated model. One limit of these techniques is their complexity in term of algorithmic, consequently, these techniques are currently especially applied to the critical cores of applications. Complete application can thus present incoherence of functioning which are not detected (non-exhaustive checking).

- The situations of type 4 are the subject of currently academic work (wrappers) which aim is to detect and to confine the errors of the device by the installation of observers which, on line, check the coherence of the sequences of signals emitted by the device. The detected errors are transmitted to the application programs.

- The object of this paper is to propose a method which identifies from design phases of conception, all the situations which can produce dysfunctionning of type 5 and 6. The method solicits the designer then so that it proposes alternatives of reaction to these situations which are not considered in the specifications. The treatment of the cases of dysfunctionning 5 and 6 are of importance in automotive application, because they can occur either because of residual defects at the end of dysfunctionnings 1, 2 or 3, or because of the incompletely controlled re-use of functionalities in different versions of vehicles, or during the construction of new inter-systems services which emit new sequences of signals to the functionalities they are associated with.

The proposed method is intended to be applied upstream of the design phase of new functionality or during the analysis of the re-use, in a new environment, as illustrated in the figure 1.

It is about a formal method because:

- it is based on a modeling of the inputs and outputs of the functionality. In this modeling, the inputs and outputs are coded by Boolean signals.

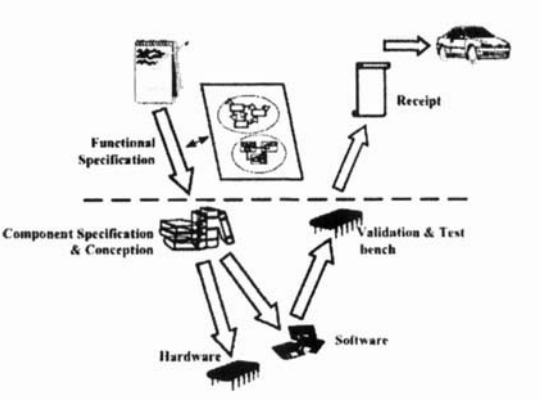

Figure 1. Domain of applicability of the method.

Coding depends on the studied case: either these signals are real if they are logical sizes (frequent case of the functionalities cockpit), or coding models the presence (the update) and the absence of data, or coding is the binary conversion of enumerated data, etc...

- it is based on a model of the specifications of the studied functionality. This model, of Finite State Machine (FSM) type, is a graph of states which describes only the dependences between the Boolean values of the inputs and the outputs of the functionality without describing how these output signals are calculated. In our approach, a finite state machine is modeling the entity connected to sensors and actuators. The input of the sensors (signal processing) and the control command of the actuators (automation algorithms) are not directly involved in this study. They are associated to the inputs from sensor and to the outputs towards the actuators

- it analyzes and fulfills the properties of completeness (Leveson, 1995) and consistency (Heimdahl and Leveson, 1996) of the model, which are two key properties of automaton specification with high level of quality, independently of the checking of the requirements of the functionality.

A specification is "complete" if there is a state transition or behavior specified for any input or set of inputs that may occur. Consistency requires that there is no more than one state transition for any input or set of inputs that may occur. In this context, the proposed method is not in competition with the techniques of model checking or theorem proving. Applied in upstream phases of these techniques, the proposed method ensure that these techniques will carry out an exhaustive validation of the function. In addition, the model built by

the method is independent of that of the model checking. Our model describes what the function must carry out whereas the model checking is based on a model which describes how it is carried. In this paper, we propose an approach based on the notion of modal decomposition, upon which the global behavior of the control system is described. A formal operation is then introduced to combine and to define legal modes commutations with respect of properties of consistency and completeness of the resulting automaton. Perspectives to increase robustness of the system against inconsistent inputs are finally proposed.

2. BASIC CONCEPT

The detailed study of real cases showed us that it is very useful to obtain the external model of a functionality to be developed by following a modal step. This step consists in breaking up the behavior of each component implied in the functionality into its various operating modes, each mode is representative of the activity of the component under specific operating conditions. The question of modal decomposition is not recent, it was already the subject of many scientific as well as technical works (Degani and Kirlik, 1995) (Jonhson, 1990). Modes differ between them by their functional components and the services which they provide (Elloy, 2002). The mode term designs nominal mode as well as failure mode or downgraded mode. In the literature, there is no general method for decomposing the system in its modes. At most, some guides exist, which are often limited to an applied domain (Moreno and Peulot, 1997). The method we propose is not based on any particular technique of decomposition of modes: all are applicable provided that modes define a mutually exclusive set of system behaviors. It is essential however that the initial system is broken up into the greatest number of elementary components as possible and that the behavior of each one of these components is broken up into the greatest number of possible modes. These two conditions make the method we propose the most efficient: i) it reduces the initial problem, which can be of a great complexity, to a set of independent subproblems, each one of reasonable complexity, ii) it automatically builds the relations between all the elementary models, in order to guarantee the coherence and the consistency of the operation of the total system.

3. METHOD STEPS

The proposed approach consists of the following major steps, as illustrated on the figure 2.

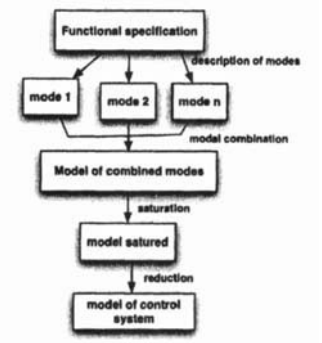

Figure 2. Method steps

3.1 Description of modes

During this first step, modes are identified and described independently from the specification requirements. The proposed approach use a formal modeling language for describing each mode. The formalism of modeling used is based on the general model of Finite State Machine (FSM). It is well recognized in both the research literature and in practice, that state machine model is a convenient way for describing the behavior of model-based system (Sherry and Cuard, 1995). A machine modeling a system can be thought of as having a set of states. The behavior of the system can be described by the possible transitions from one state to one another (Arnold, 1992) (Gill, 1962). This type of modeling is particularly adapted to the specification of control logic system (Harel, 1987). There are two basic types of state machines: Moore state machines (Moore, 1956) and Mealy state machines (Mealy, 1955). They differ only in how they compute their output signals. For our approach, we use an extension of Moore machine so that the state of the machine is defined by the current values of inputs and the current values of outputs

Formally, « a machine M » with n inputs and p outputs, in our approach is defined as:

$M = (E, S, Q, I, F, T)$ where:

E : $\{e_1, \ldots e_n\}$ set of n boolean inputs ($n \geq 1$),
S : $\{s_1, \ldots s_n\}$ set of p boolean outputs ($p \geq 1$),
Q : set of possible states, $card(Q_A) \leq 2^{n+p}$,
I : set of initial states ($I \subseteq 2^{n+p}$)
F : set of terminal states ($F \subseteq 2^{n+p}$)
T : set of transitions ($T \subseteq 2^{n+p} \times 2^{n+p}$)

A state (e, s) is defined by the combination of inputs e and outputs s. An initial state, represented by a circle with an arrow, defines state whereby mode

can be entered. Terminal state, represented by the underlined circle, defines state whereby mode can be left.

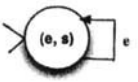

Figure 3. Initial state

Figure 4. Terminal state

A transition $(e,\ s,\ e',\ s')$, is illustrated on the figure 5.

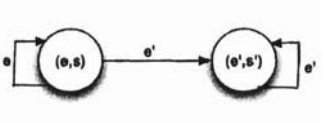

Figure 5. Transition

The formalism of modeling uses a graphic reduction for the representation of states. The notation illustrated on the left of the figure is a graphic reduction of the automaton at the right of the figure 6.

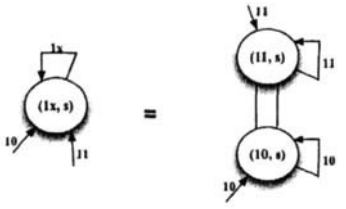

Figure 6. Representation of state

The advantages of this graphic reduction which does not distort the semantics is the possibility to perform complex computation on automaton and the use of tools based on BDD (Binary decision diagram) to verify proprieties, by avoiding exponential growth of states. Checking the resulting system for properties of reachability is possible with almost no limitation due to the use of BDD instead of an explicit automaton.

3.2 Modal combination

This step consists in combination of the modes independently described in order to build the machine representing the complete behavior of the system.

To guarantee determinism of specification, the modal combination of two machines $M_1 = (E_1, S_1, Q_1, I_1, F_1, T_1)$ and $M_2 = (E_2, S_2, Q_2, I_2, F_2, T_2)$ must be preformed under some conditions. First, the modes to be combined must present the same Boolean inputs ($E_1 = E_2$) and outputs ($S_1 = S_2$). Then, the set of modes must be disjoined ($reachable(M_1) \cap reachable(M_2) = \varnothing$).

Formally, *the modal combination* of two machines $M_1 = (E_1, S_1, Q_1, I_1, F_1, T_1)$ and $M_2 = (E_2, S_2, Q_2, I_2, F_2, T_2)$ builds a machine $M = (E, S, Q, I, F, T)$ where:

- $E = E_1 = E_2, \quad S = S_1 = S_2, \quad Q = Q_1 \cup Q_2$
- $I = I_1 \cup I_2, \quad F = F_1 \cup F_2, \quad T = T_1 \cup T_2,$

According to the way in which the designers have identified the modes, two situations can be met:

- All the modes identified are exclusive of other modes. The machines modeling modes do not share any common state, including the initial states.
- Modes are not disjoined. Their accessible parts share common states. A preliminary fusion operation of modes is possible if the common states to the modes present the same properties in terms of states and transitions. If not, a supplementary Boolean output is introduced to differentiate identical states of modes.

To guarantee completeness of specifications, any possible and impossible evolutions of the global system must be considered. The impossible evolutions are those of the occurrence in a given state of a certain combination of inputs which are physically impossible. During the modal combination, the designer will have to introduce additional modes which correspond to the occurring of these events of *impossible* evolution. These additional modes are often called *Default mode or Reply mode*. This step constraints the designer to think explicitly of the occurrences of all the combination inputs and to evaluate their incidence. This constraint is an important factor for the safety of the system.

3.3 Saturation

The saturation is the operation which follows upon the modal combination. Its purpose is to complete the model from modal combination by setting connections between modes, so that the automaton resulting is deterministic, complete and reachable.

The saturation operation of a machine $M = (E, S, Q, I, F, T)$ with n inputs and p outputs from combination modal consists in computing a machine $M' = (E', S', Q', I', F', T')$, so that:

- from each reachable terminal state of M, the complementary transitions of set of transitions starting from this state is computed. According to the hypothesis that mode is always entered via its initial states, these complementary transitions are added towards each initial states of M.

- for each transition added, an action is associated so that, this action is disjoined of these starting from the state and this action is stable towards the target state. The stability of a transition towards a target state implies the coherence between the input transition and the condition to stay in the state.

Formally, $M' = (E', S', Q', I', F', T')$ where:

- $E' = E,\ S' = S,\ Q' = Q,\ F' = F,\ I' = I,$

- $$T' = T \cup \left\{ \begin{array}{l} \forall\,(e,\ s) \in reachable(M) \\ \text{and } \forall\,(e,\ s) \in F \\ \text{and } \forall\,(e',\ s') \in I \\ \text{and } \forall\, s''\ (e,\ s,\ e',\ s'') \notin T \\ \text{and } e \neq e' \ :\ (e,\ s,\ e',\ s') \end{array} \right\}.$$

Due to this definition of saturation, the determinism and the completeness of the machine resulting from saturation is ensured by construction under these conditions:

- modes are disjoined other them. The intersection of their reachable states is empty.

- the set of initial states inputs forms a *partition* of the set of possible combination values of inputs. That means, they are mutually exclusive and the set of them is collectively exhaustive.

- all the states are terminal. If not, completeness is ensured only for terminal states, for the non - terminal states, the object of the approach is to detect these incompleteness and to inform the designer, so that he proposes alternative strategies for the treatment of these incompletenesses.

An example of construction by saturation is illustrated on the figure 7, which consists of a a manual basic wiper system. It is comprised of the wiper switch, the sensor for detecting the position of the wiper arms and the actuator.

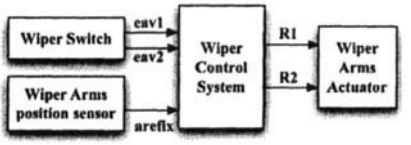

Figure 7. Manual Wiper Function

The boolean inputs of the Wiper Control System are eav1, eav2 and arefix, which combinations correspond to the selected mode. The boolean outputs are R1 and R2, which correspond to the command emitted by actuator to the the wiper arms . From the specification of the function, 5 modes are displayed : mode "stop", mode"speed1", mode"speed2", mode"wait stop", mode "Default switch". The description of each mode using moore machine is given on the figure 8.

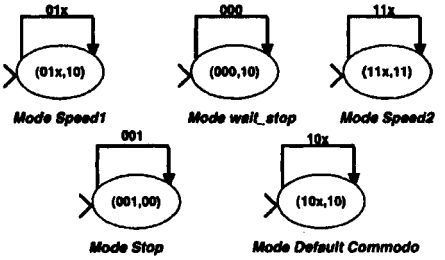

Figure 8. Wiper System modes

The external model of the wiper system after saturation is illustrated on the figure 9. TThe resulting automaton respects properties of completeness and consistency by construction.

4. ROBUSTNESS TO INCONSISTENT INPUTS

A goal for the proposed approach is to assist in writing complete and consistent functional specifications. Safe functional specifications imply that the global system must design to cope with all possible fault conditions. We are focused on fault conditions resulting from input component failures. The obvious solution, is to increase the reliability of the components and to build fault tolerance into system design to take in account these failures. This can be considered a priori in several ways:

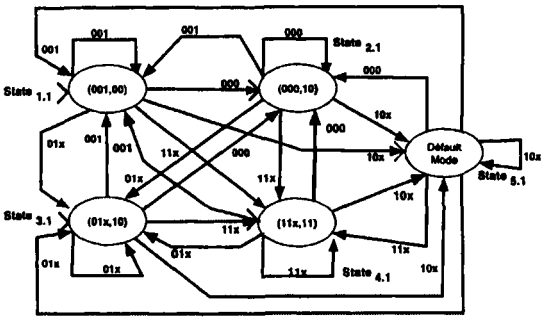

Figure 9. External model after saturation

4.1 Perspective 1: failures are treated by the control system.

A first perspective consists in considering these components failures as an integral part of the operating modes of control system. The automaton modeling the control system after saturation takes into account all possible circumstances, including those resulting from the anomalies of components. Then, the failed transitions are identified and a specific default mode is defined as an additional state of the control system. These failed transitions are then redirect towards this state. This perspective is illustrated on the figure 10.

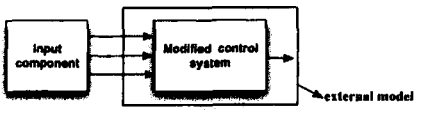

Figure 10. Failures are treated by control system

4.2 Perspective 2: failures are treated by a filter.

In this case, the control system does not have to face the occurrence of these failures and to treat them. The failures are filtered at the output of components, as illustrated on the Figure 11.

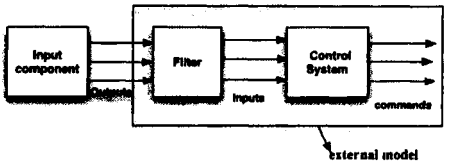

Figure 11. Failures are treated by a filter

4.3 Perspective 3: failures are supervised by parallel monitoring.

A third perspective considers that the failures are detected by a specific monitoring module which relays the control system when failure occurs. The role of this monitoring box is to force the signal command in case of undesired behavior. In absence of failures from input components, the command from the control system is applied. In the opposite case, specific command which is previously established to treat this circumstance is applied.
The role of the monitoring box is that of a detector of failure and a selector. This configuration is illustrated on the figure 12.

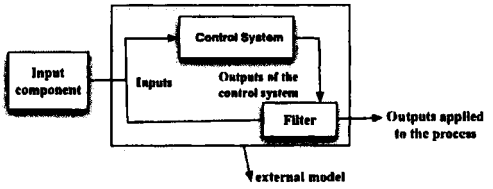

Figure 12. Failures are supervised by parallel monitoring

These three possibilities can be considered. It is up to the designer to choose the one which is suitable for his application.

5. CONCLUSION AND FUTURE WORK

We have outlined an approach for the design of a safe control system by reducing potential specification errors. The system is described according to its "operating mode" and the formal operation of "saturation" creates transitions for commutation between modes with respect of properties of consistency and completeness of the automaton resulting. This work is still in evolution steps. For the future, we plan to investigate the formalization of the whole steps of

the design process. This can be used as a starting point for the definition of formal tools based on BDD representation.

REFERENCES

Alur, R., Courcoibetis, C., and Dill, D. (1993). Model checking in dense real-time. *Information and computation*, 104:2–34.

Arnold, A. (1992). *Systemes de Transitions Finis et Semantique des Processus Communiquants*. Masson.

Boyer, R. and Moore, J. (1984). Proof checking, theorem proving, and program verification. In *Contemporary Mathematics, Automated Theorem Proving : after 25 years*, pages 119–132, Rhode Island. American mathematical society.

Degani, A. and Kirlik, A. (1995). Modes in human-automation interaction : initial observations about modeling approach. In *Proceeding of the IEE International Conference on Systems, Man, and Cybernetics*, pages 3443–3450, Vancouver, Canada. IEE.

Elloy, J.-P. (2002). Modelisation des modes de fonctionnement dans une architecture vehicule en ail. Technical report, AEE.

Gill, A. (1962). *Introduction to the Theory of Finite-State Machines*. Mc Graw-Hill, New York.

Harel, D. (1987). *Statecharts : a visual formalism of complex systems*. Science of Computer programming 8:231-274.

Heimdahl, M. and Leveson, N. G. (1996). Completeness and consistency analysis of state-based requirements. *IEEE Transactions on Software Engineering*.

Jonhson, J. (1990). Modes in non computer devices. *International Journal of Man-Machines Studies*, 32:423–438.

Leveson, N. G. (1995). *Safeware: System Safety and Computers*. Addison Wesley, Reading, Massachusetts.

Mealy, G. H. (1955). Method for synthesizing sequential circuits. *Bell System Tech.*, 34:1045–1079.

Moore, E. F. (1956). Experiments on sequential machines. In *Automata Studies*, pages 129–156, C.E. Shannon and J. McCarthy, editors. Princeton University Press, New Jersey.

Moreno, S. and Peulot, E. (1997). *Le GEMMA : Mode de Marche et d'arrêt*. Educalivre-Collcetion A. Capliez.

Sherry, L. and Cuard, J. (1995). A formalism for the specification of operationally embedded reactive systems. In *14th digital avionics systems conference*, Cambridge MA. AIAA/IEEE.