Hierarchical Modelling and Diagnosis for Embedded Systems

Hervé Ressencourt[1,2] **Louise Travé-Massuyès**[1] **Jérôme Thomas**[2]

[1] LAAS-CNRS
7, Avenue du Colonel Roche
FRANCE-31077 TOULOUSE Cedex 4
hressenc,louise@laas.fr

[2] ACTIA
25, Chemin de Pouvourville
FRANCE-31432 TOULOUSE Cedex 4
jerome.thomas@actia.fr

Abstract

Because of the increasing complexity of engineered systems, abstractions and hierarchies in models are receiving great attention. The behaviour of embedded systems is commonly characterised by hybrid phenomena in which each operational mode is activated by electronic units: it hence involves hardware and software components. The aim of this work is to apply a multimodelling approach on such systems for the diagnosis task. This is illustrated by an example taken from the automotive domain.

1 Introduction

The increasing complexity of engineered systems led the Model-Based Reasoning (MBR) communities to focus their research in reasoning tasks - like diagnosis - based on multiple abstraction level models organised through a hierarchy. Abstractions are useful to reduce the computational complexity of diagnosis reasoning, to account for observations at qualitative levels, and to handle systems whose available knowledge about components is heterogeneous.

Two kinds of hierarchies are commonly used in MBR: structural abstraction [Chittaro and Ranon, 2004] [Mozetic, 1991] [Autio and Reiter, 1998], which aggregates components to describe the system at different levels of detail and functional abstraction, which abstracts the behaviour according to the functional and teleological understanding of the system [Chittaro *et al.*, 1993] [Kitamura *et al.*, 2002]. The main idea of a functional description is to bridge from behavioural to teleological knowledge (knowledge about goals) by exhibiting the functional roles that the structural components may play in the achievement of the function of the whole system.

The objective of this work is to devise a multimodelling cooperation framework for the diagnosis of complex embedded hybrid systems controlled by electronic units. A brief overview of existing approaches on functional modelling is first proposed in section 2. In the third section the limits of these approaches are discussed and an extention of Chittaro's framework [Chittaro *et al.*, 1993] is proposed to model hybrid physical systems including hardware and software components. Then, some perspectives are presented for the off-board diagnosis task of automotive systems based on this framework.

2 Knowledge representation for Model Based Reasoning

Knowledge representation is a key issue in MBR. Luca Chittaro and colleagues [Chittaro *et al.*, 1993] write that choices have to be made, especially about ontologies, epistemological types, representational assumptions and aggregation levels. These choices are mainly directed by the goals of the models (design analysis, diagnosis...) and by the requirements of the reasoning task. It is commonly accepted that knowledge about physical systems can be organised through two axes [Lind, 1982]:

- The *Whole-Part hierarchy* relies on different aggregation levels for a same type of knowledge. An entity of this hierarchy is a part of the upper one. For example, structural abstraction has been used for the diagnosis task [Chittaro and Ranon, 2004] [Mozetic, 1991] [Autio and Reiter, 1998].
- The *Mean-End* or *functional hierarchy* relies on the theological understanding of behavior [Chittaro *et al.*, 1993] [Kitamura *et al.*, 2002].

A functional description hierarchy has to answer three questions: "Why was the system designed?", "What is the system supposed to do to achieve the goal?" and "How must different parts of the system interact in order to realise the functions?" [Modarres and Chehon, 1999].

2.1 Functional abstraction hierarchy

Several works agree on a model hierarchy consisting in a distinction between four epistemological types :

- The *Structural knowledge* is the knowledge about system topology.
- The *Behavioural knowledge* describes the physical laws underlying the behaviour of components composing the system.
- The *Functional knowledge* describes the roles components may play in the process in which they take part. This level is named Base-Function layer in [Kitamura *et al.*, 2002].

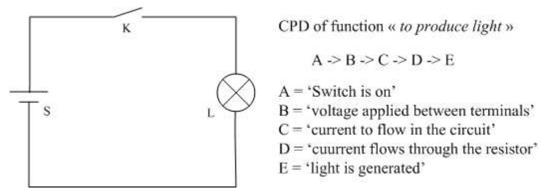
K
S
L
CPD of function « to produce light »
A -> B -> C -> D -> E
A = 'Switch is on'
B = 'voltage applied between terminals'
C = 'current to flow in the circuit'
D = 'cuurrent flows through the resistor'
E = 'light is generated'

Figure 1: CPD example for an electrical circuit whose function is *to produce light*

- The *Teleological knowledge* describes the goals of the system intended by its designer. This level is named Meta-Function layer in [Kitamura *et al.*, 2002].

The functional knowledge level aims at bridging the structural and behavioral knowledge on one side and the teleological knowledge on the other side, which respectively rely on two different ontologies :

- The *object-centered ontology*, sometimes named *component ontology*, assumes that the system is made of individual objects with independent context properties and stated in a generic way.
- The *system-centered ontology*, sometimes named *process ontology*, is a context dependent ontology. It assumes that the system involves a set of physical phenomena which are activated/disactivated according to the current context.

Following these lines, [Chittaro *et al.*, 1993] elaborated a proposal called the *multimodelling approach*, which brings solutions to many critical problems like the formalisation of the links between each model, their meaning and the representation language at each level. Notice that [Chittaro *et al.*, 1993] framework matches the one by Kitamura [Kitamura *et al.*, 2002]. One feature of this approach is to implement the functionnal model in three interlinked levels: a model of *functionnal roles*, a role being associated to a single component, a model of *processes* emerging from functional role networks, and a model of *phenomena*.

2.2 Different approaches for functional modelling

Being at the crossroads of a component and a process based ontologies, the functional model necessarily relies on a hybrid ontology. Many approaches have been suggested in the literature for representing functional knowledge. They can be classified into two categories: the state based approaches [Chandrasekaran, 1994] [Price and Snooke, 1998], relying on the abstraction of behaviour states and the flow based approaches, relying on flow models [Lind, 1982] [Chittaro *et al.*, 1993]. Other works have been interested in the use of an ontology for defining the functional concepts [Chittaro *et al.*, 1993] [Kitamura *et al.*, 2002].

The state-based representations

Functions are built from the knowledge of the causal relations existing among system's states. System's states correspond to some instance assigned to the variables describing the system (generally assumed to have discrete value domains). Thus, a directed graph can be defined in which the nodes are predicates about the states of the system and links indicate causal relations. This graph is commonly named Causal Process Description (CPD) [Chandrasekaran, 1994]. All paths in this graph can be interpreted as a function of the system. This framework has been predominantly used for simulation and design analysis tasks [Bell *et al.*, 2005] [Price and Snooke, 1998]. A simple example of a lighting circuit is provided in figure 1 to illustrate the approach.

It should be noticed that such functional models may not be reusable since modelling choices are subjective and may vary from user to user. Nevertheless, one advantage of this approach is that discrete event behaviors can be easily described together with continuous phenomena as long as they are represented at a high level of abstraction.

The flow-based representations

They are based on the concepts of generalised variables of flow and effort [Lind, 1982] [Chittaro *et al.*, 1993] . These concepts were firstly used by Paynter [Paynter, 1961] and the Bond Graph community. In this method, a finite set of functional primitives is defined and the functional description is expressed in terms of these primitives. It should be noticed that the primitives are the same for all flow-based representations. The main advantage of this approach is that a real ontology is defined, on which the functional modelling of any physical system can rely. Functional primitives are linked to structural and behavioral components so that a given system functional model can be generated automatically from the behavioral and structural knowledge. One criticism about this approach is that only physical devices are modelled. No ontology is suggested for components which have discrete events and sequential behaviours.

Despite of the difference between state-based and flow based approaches, they use a common principle : the functional model consists in a causal interpretation of behaviour.

3 Extended multimodelling framework

In this section, we propose a multimodelling framework based on an extention of Chittaro's to include software components implementing control actions, and hence deal with hybrid systems. Our approach stands on:

- adding a mode labelling to the behavioural model that links to the corresponding operating mode,
- introduce new *software* processes and phenomena in the model of processes and phenomena, respectively.

The approach is illustrated through all the section by a case study from the automotive domain presented in section 3.1.

3.1 Case study taken from the automotive domain

The rear wiping system, taken from the automotive domain, has been chosen (see figure 2) to illustrate the multimodelling problem for hybrid and controlled systems. There are three means to activate the rear system on some modern cars: on request of the driver through activation of the steering wheel switching module, on request of the rear washing function, when the screen wiper is activated and the driver engages the reverse mode.

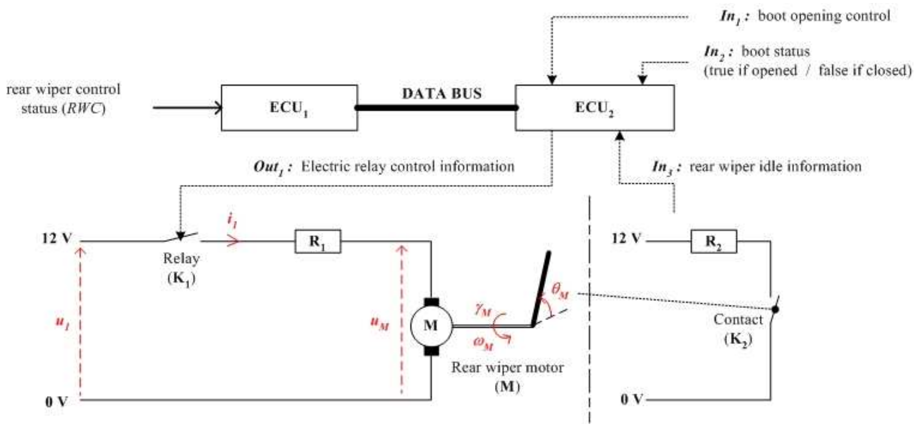
In_1 : boot opening control
In_2 : boot status (true if opened / false if closed)
rear wiper control status (RWC)
ECU_1
DATA BUS
ECU_2
Out_1 : Electric relay control information
In_3 : rear wiper idle information
12 V
i_1
R_1
Relay (K_1)
u_1
u_M
M
γ_M
ω_M
θ_M
Rear wiper motor (M)
0 V
12 V
R_2
Contact (K_2)
0 V

Figure 2: Synoptic diagram of the rear wiper system. The electrical variables u_1, u_M and i_1 represent respectively the voltage applied to the electric circuit pins, the counter electromotive force and the intensity. The mechnical ones ω_M, γ_M and θ_M are respectively the angular velocity of the rotor, the torque and the angular displacement of the rotor.

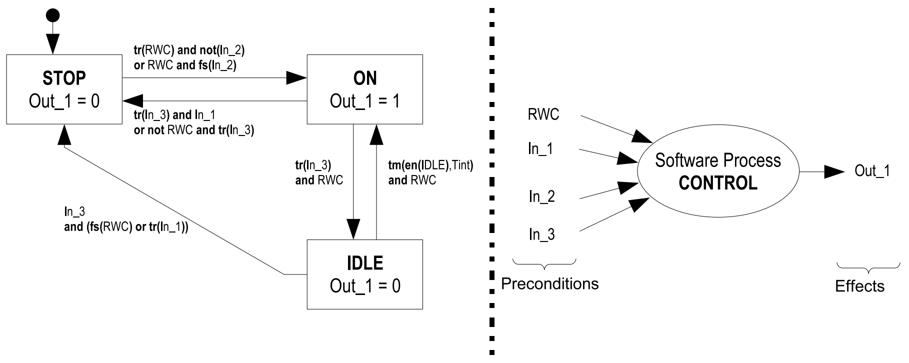
STOP
Out_1 = 0
ON
Out_1 = 1
IDLE
Out_1 = 0
tr(RWC) and not(In_2) or RWC and fs(In_2)
tr(In_3) and In_1 or not RWC and tr(In_3)
tr(In_3) and RWC
tm(en(IDLE),Tint) and RWC
In_3 and (fs(RWC) or tr(In_1))
RWC
In_1
In_2
In_3
Software Process CONTROL
Out_1
Preconditions
Effects

Figure 3: The automaton describing the behaviour of ECU$_2$ (on the left) and its abstraction by a software process named CONTROL. The conditions *tr(e)* and *fs(e)* mean a rising edge and a falling edge on the event *e*, respectively.

The synoptic diagram of this system is given in figure 2. The rear wiper system is composed of two Electronic Control Units (ECU). The role of the first one (ECU$_1$) is to receive the request "rear wiper on" (RCW = 1) selected by the driver on the steering wheel switching module and to transmit this status to the second one (ECU$_2$) through the data bus. ECU$_2$ has to elaborate and send the control signal which closes the electric relay taking into account the user control data and other inputs coming from other functions of the vehicle.

This system is a sum of hardware components (electrical circuits, mechanical devices...) and software components (embedded pieces of software in ECUs, data bus). The behaviour of software components is assumed to be described by state machines[1].

The behaviour of the rear wiper system (figure 3) is characterized by a cycle composed of two activities: one wiping action followed by a waiting state of the wiper at the iddle position $\theta_M = 0$ during a time interval T_{int}. The iddle state is triggered by ECU$_2$ detecting the event associated to $\theta_M = 0$ sent by the switch K$_2$. Such a behaviour is named "*intermitent wiping*" in the automotive domain.

[1]State-charts are widely used in the automotive industry for software modelling.

3.2 The structural and the behavioural models

The structural model

The structural model describes the topology of the system by using three primitives: the components and their terminals, nodes (to connect together two ore more components) and connections (to describe how components are connected together through nodes).

The behavioural model

The behavioural model describes the internal properties of each component. There are three kinds of behaviour: the continuous behaviours are described by continuous equations (arising from physical laws for instance), the hybrid components are described by hybrid automatons and the software components are described by discrete automatons[2]. In the

[2]In this paper, we restrict ourselves to discrete controllers

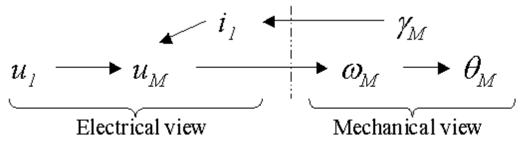

Figure 4: The causal model of the rear wiper motor

case of hybrid components, the continuous behaviour relations are labelled according to the corresponding operating mode. Software components are virtual and defined by the fact that they each implement the control of a hybrid component.

3.3 The functional model

The functional model has to achieve the bridge between the behavioural and the teleological knowledge. So, its aim is to describe how the behaviours of individual components contribute to the achievement of the function intended by the designer. Note that the same component may contribute to the achievement of more than one function when acting in different operating modes. Functional modelling is performed by using three progressive levels of interpretation: the causal model, the model of processes and the model of phenomena. Notice that our functional model replaces the functional role model of [Chittaro *et al.*, 1993] by a causal model, which allows us to exhibit the processes automatically as proposed by [Thétiot *et al.*, 1998].

The causal model

Causality is one of the essential concepts for reasoning about physical systems. It is widely used in qualitative physics to explain and to predict physical systems' behaviours.

Several operational methods have been proposed for the automatic generation of *causal links*, also named *influences*, from the behavioural knowledge. Among the most well known methods are causal ordering algorithms [Iwasaki and Simon, 1994] [Travé-Massuyès and Pons, 1997]and the Bond Graph based method [Thétiot *et al.*, 1998]. By considering the rear wiper electric motor example (figure 2), the behavioural equations when the relay K_1 is closed are:

$$u_1 = U_0 \quad (1)$$
$$u_1 = R_1 \cdot i_1 + u_M \quad (2)$$
$$u_M = k \cdot \omega_M \quad (3)$$
$$\gamma_M = k \cdot i_1 \quad (4)$$
$$\gamma_M = C_r \quad (5)$$
$$\frac{d\theta_M}{dt} = \omega_M \quad (6)$$

In this behavioural description, two physical views are represented. The equations (1) and (2) correspond to an electrical view and the equations (5) and (6) correspond to a mechanical view. The mapping between this views is given by equations (3) and (4). The causal influences between the variables of this device, given in figure 4, form a causal network in which each arrow between two variables x and y ($x \longrightarrow y$) means that "x influences y" or that "the events occuring in x influence the events occuring in y", without specifying these events.

When the behaviour of a physical system is hybrid (with different operating modes), a causal model is computed for each mode and labelled accordingly. In the exemple, the causal model of figure 4 corresponds to the mode "the electric relay is closed" and this mode is activated by the state of the software implemented in the ECU$_2$.

Hence, like for the behavioural model, the software components are represented at the causal model level by the labelling. One reason for this choice is that software components implement control actions determining the operating mode of components according to the events occuring on their inputs. Their input-output mapping is explicitly represented at the level of processes (cf. section *The model of processes*) and above.

The model of processes

The model of processes represents the set of processes that may occur in a system and their relationships. One process is specialised in one physical or software domain. *Physical processes* are mapped to a set of causal influences of the causal model. *Software processes* are each mapped to the *automaton* that determines the mode of one physical component. In figure 3, the software process named *CONTROL* is an abstration of the partial automaton that determines the mode of the electric relay through the output *Out*$_1$ of ECU$_2$. So, a process is represented by a four-tuple <cofunction, precondition, effect> [Chittaro *et al.*, 1993]:

- *cofunction* is a causal network which specifies which causal influences are necessary to enable the occurence of a physical process; or an automaton for a software process.
- *precondition* is a logical predicate which characterises the situation which enables the process to occur.
- *effect* is a logical predicate which characterises the situation during the occurence of the process.

The organisation of processes of the rear wiper system is given in Fig. 5. There are three kinds of processes according to the different views:

- Pm1, Pm2 and Pm3 are mechanical processes. Pm1 and Pm3 are named "SWITCHING" because they are related to the mechanical actions on the rear wiper control and on the contact K$_1$, respectively.
- Pe1, Pe2 and Pe3 are electrical processes. They are named "TRANSPORTING" like in [Chittaro *et al.*, 1993].
- Ps1, Ps2 and Ps3 are software processes. Ps1, named "STORAGE", is related to the software component ECU$_1$ whose role is to observe the state of the rear wiper control, to store it in a message for being sent to the ECU$_2$ through the data bus. Ps2, named "TRANSPORTING", is related to the proccess occurring in the data bus. Notice that the process "TRANSPORTING a generalised variable" described in [Chittaro *et al.*, 1993] is

extended to “TRANSPORTING a message” in the software view. Then, Ps3 is related to the software process occurring in the ECU$_2$.

The model of phenomena

The last model in the functional knowledge level is the model of phenomena. One phenomenon is described by a four-tuple $<$organisation, precondition, effect$>$ in which *organization* is a process network which defines which processes are necessary and how they must be related together in order to enable the occurence of the phenomenon. So, a phenomenon is an aggregation of processes organised throught causal links.

Two high level phenomena are described for the rear wiper system (Fig. 5):

- The phenomenon PH1, named “WIPING” is related to the wiping action of the system (to the state “ON” of the automaton Fig. 3).
- The phenomenon PH2, named “WAITING” is related to the idle state of the system.

Like in [Chittaro *et al.*, 1993], the *ontological links* allow one to describe the links between phenomena and processes.

3.4 The teleological model

The teleogy of a system is defined as the specification of the functions as they are intended by its designer. This notion is close to the perception of the behaviour by a human user. The definition of function which is be used in this work is:

Definition 1 *A function defines a mapping between a conjunction of conditions on atomic inputs and a given system’s state as it is intended by the designer or perceived by the user as output.*

A function is commonly represented by a triple $<$function pattern, operational conditions, intended behaviour$>$[3]:

- The *function pattern* assigns a name to the function and specifies its arguments which are variables relevent to the definition of the goal. ”To wipe the rear window with a user activation” identifies one function of the system. The angular position θ_M of the wiper is the argument.
- The *preconditions* are the operational conditions which specify what should be provided as input to the system for the achievement of the intended function. There are two operational conditions for the example: ”the rear wiper control is activated” (RWC) and ”the boot is closed” (In_1).
- The *effects* specifies the behavioural intended behaviour the function. For the example, the effect is: ”intermittent wiping” mapped to the variable θ_M.

This representation of function refers to two important notions: on the one hand, it describes a goal in terms of the desired artifact’s behaviours and on the other hand it is linked with the notion of testability. The operational conditions and the intended behaviours have to be clearly defined to enable testing the function achievement. This issue is discussed in the next section for the diagnosis task.

[3]Some authors use the terms preconditions and effects instead of operational conditions and intended behaviour, respectively. We prefer the later because these terms map better the physical system to the human designer/user

3.5 The sequential behaviours

It can be noticed that in Fig. 5 the abstraction of the two states sequences “IDLE” and “ON” up the hierarchy deserves further attention. For this kind of behaviour, classified as “sequential and intermittent behaviour”, preconditions and effects may be expressed by using temporal logic operators [Bell and Snooke, 2004].

4 Off-board multimodel based diagnosis

4.1 The off-board diagnosis issue in the automotive domain

Diagnosis is the process of identifying the cause (fault) of a system’s malfunction by observing the system at various monitoring (test) points. The number of possible causes of dysfunction has increased with the technological advances of automotive systems while reduction in the number of monitoring points results in reduced observability, making increasingly difficult to troubleshoot vehicles.

The different types of observations

The diagnosis task is driven by the available observations. In the automobile domain, observations are of different types ranging from functional symptoms reported by the clients to qualitative observations and physical measurements. The observations can be classified as follows:

- A *functional symptom* relies on a high level observation by providing information about the functions and their failures. More precisely, it refers to a missing intended behaviour of a function. When it is reported by a client to the garage mechanic, it is called “client symptom”.
- *ECU’s data*: when the garage mechanic connects its computer to the diagnosis interface of a car, he can access some input/output variables of the ECUs, useful for the diagnosis task. These variables are of two types: physical or logical quantities or fault codes[4].
- A *physical measurement* is an observation at the behavioural knowledge level.

The test problem

The off-board diagnosis problem, in the automotive domain, is equivalent to a test problem. The diagnosis activity starts with a set of *preliminary symptoms* gathered by the garage mechanic. These preliminary symptoms are fault codes, client symptoms and other preliminary garage mechanic observations.

Then, the fault isolation problem is defined as the determination of the additional information (obtained by tests) which allow the best discrimination among the diagnostic hypotheses generated with the preliminary symptoms. One test is defined by the variable which has to be observed,

[4]Most ECUs are equiped with an auto-diagnosis function which reliably detects which of the electric circuits connected to one ECU are failing. The failed electric circuits are associated with fault codes

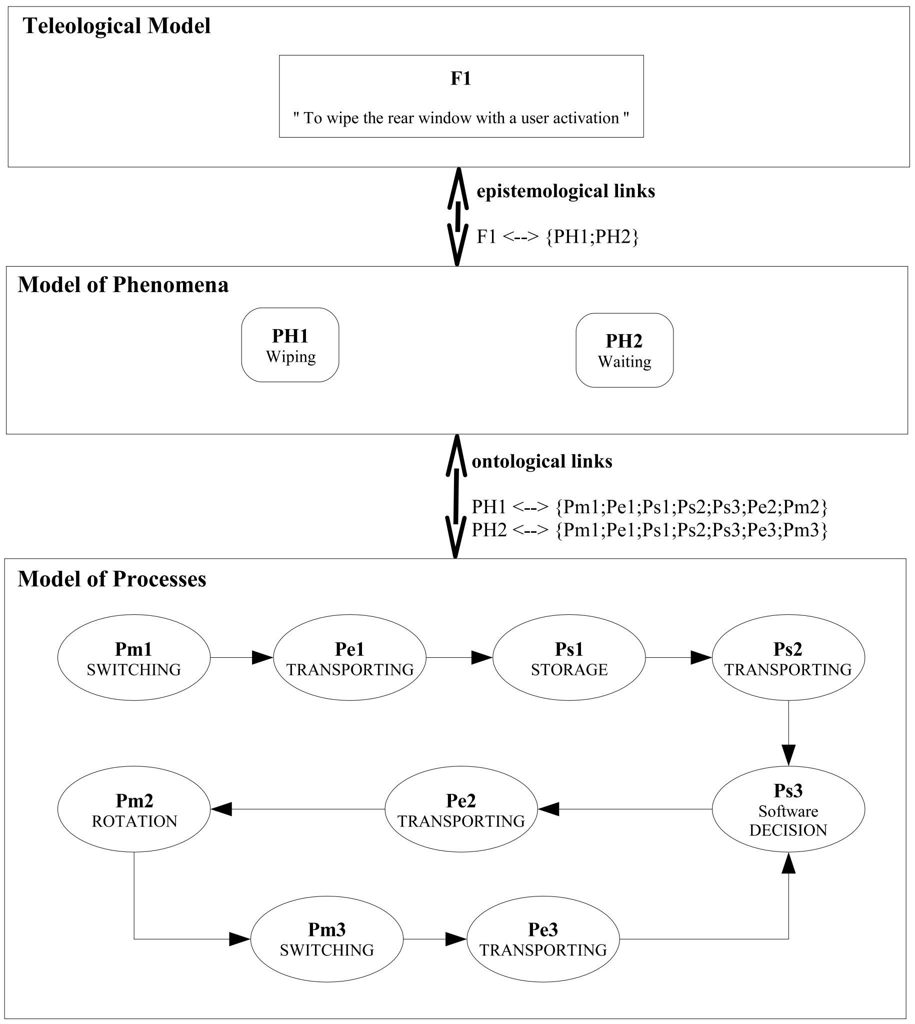

Figure 5: Teleological abstraction of the rear wiper system's behaviour.

the configuration in which the system must be to perform the test and the possible outcomes of the test (generating new symptoms). Some previous works have proposed solutions to diagnose electric circuits in the automotive domain [Faure, 2001][Olive, 2003][Price *et al.*, 1995][Sachenbacher and Struss, 2001].

4.2 Perspectives for the diagnosis task

There are few approaches in the litterature which use the four epsitemological types in a cooperative way for the diagnosis task. Chittaro [Chittaro *et al.*, 1993] has suggested one method for focusing the diagnostic activity. Interpretative knowledge indeed permits to achieve the diagnostic task in a hierarchical way. At the teleological level, client symptoms allow one to identify the functions which undergo failures. By exploiting the bridge between teleology and behaviour, only those parts of the structural and behavioural models which are responsible for the unachievement of the functions can be considered.

The different steps of the diagnosis process, illustrated in Fig. 6, are organised as follow:

- The *diagnosis model generation* consists in computing the hierarchical models from design data (electrical diagrams, State-Charts, functional descriptions).
- The *symptom translation* consists in propagating the symptom up and down the different levels of the hierarchy through the different links in order to glean as much information as possible delivered by the symptom description.
- The *Diagnostic hypothesis generation* consists in the isolation of faults in the system. If the symptoms which are already available are not sufficient to correctly identify a unique hypothesis, the reasoning algorithm needs more information.
- The *test selection* issue depends on the selected diagnosis method. For this task, our objective is to suggest at each step the best next test for which the associated symptoms result in the maximal information gain.

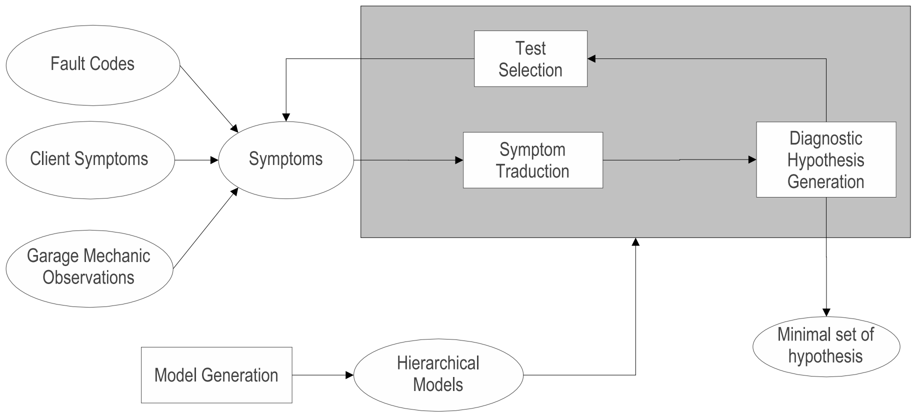

Figure 6: The synopsis of the diagnosis strategy

5 Conclusion

The work presented in this paper has pointed out the potential benefits of using a multimodel cooperation for troubleshooting embedded systems.

High level symptoms, like client symptoms, are directly linked to functions described at the teleological level. Thus, the multimodel hierarchy maps these symptoms to the behaviour of each individual components.

The multimodelling framework applied to the test sequential problem needs further investigation. The links between each level of the functional hierarchy have to be clearly defined. They should allow to propagate the observations made at a given level up or down the other levels, increasing observability and observability.

References

[Autio and Reiter, 1998] K. Autio and R. Reiter. Structural Abstraction in Model-Based diagnosis. In *13th European Conference on Artificial Intelligence ECAI-98*, pages 269–273, Brighton (UK), 1998.

[Bell and Snooke, 2004] J. Bell and N. Snooke. Describing System Functions that Depend on Intermittent and Sequential Behavior. In *18th International Workshop on Qualitative Reasoning QR'04*, Evanston, (USA), 2004.

[Bell *et al.*, 2005] J. Bell, N. Snooke, and C. Price. Functional Decomposition for Interpretation of Model-Based Simulation. In *19th International Workshop on Qualitative Reasoning QR'05*, Austria, 2005.

[Chandrasekaran, 1994] B. Chandrasekaran. Functional Representation and Causal Processes. *Advances in Computers*, 38:73–143, 1994.

[Chittaro and Ranon, 2004] L. Chittaro and R. Ranon. Hierarchical Model-Based Diagnosis Based on Structural Abstraction. *Advanced Engineering Informatics*, 155(1-2):147–182, 2004.

[Chittaro *et al.*, 1993] L. Chittaro, G. Guida, C. Tasso, and E. Toppano. Functional and Teleological knowledge in the multimodeling approach for reasoning about physical systems: A case study in diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1718–1751, 1993.

[Faure, 2001] P. P. Faure. *An Interval Model-Based Approach for Optimal Diagnosis Tree Generation : Application to the Automotive Domain*. PhD thesis, LAAS-CNRS, 2001.

[Iwasaki and Simon, 1994] Y. Iwasaki and H. Simon. Causality and Model Abstraction. *Artificial Intelligence*, 67(1):143–194, 1994.

[Kitamura *et al.*, 2002] Y. Kitamura, T. Sano, K. Namba, and R. Mizoguchi. A Functional Concept Ontology and Its Application to Automatic Identification of Functional Structures. *Advanced Engineering Informatics*, 16(2):145–163, 2002.

[Lind, 1982] M. Lind. Multilevel Flow Modelling of Process Plant for Diagnosis and Control. In *Proc. International Meeting on Thermal Reactor Safety*, pages 1653–1666, Chicago (USA), 1982.

[Modarres and Chehon, 1999] M. Modarres and S.W. Chehon. Function-Centered Modelling of Engineering Systems Using the Goal Tree-Success Tree Technique and Functional Primitives. *Reliability Engineering and Systems Safety*, 64:181–200, 1999.

[Mozetic, 1991] I. Mozetic. Hierarchical Model-Based Diagnosis. *Int. J. of Man-Machie Studies*, 35(3):329–362, 1991.

[Olive, 2003] X. Olive. *Approche Intégrée Base de Modèles pour le Diagnostic Hors-Ligne et la Conception: Application au Domaine de l'Automobile*. PhD thesis, LAAS-CNRS, 2003.

[Paynter, 1961] H.M. Paynter. In MIT Press, editor, *Analysis and Design of Engineering Systems*. Cambridge, 1961.

[Price and Snooke, 1998] C. Price and N. Snooke. Hierarchical Functional Reasoning. *Knowledge Based Systems*, 11:301–309, 1998.

[Price *et al.*, 1995] C. Price, D. R. Pugh, M. S. Wilson, and N. Snooke. The flame system: Automating electrical failure mode effects analysis (fmea). In *Annual Reliability and Maintainability Symposium*, Washington D.C., 1995.

[Sachenbacher and Struss, 2001] M. Sachenbacher and P. Struss. Aqua: A framework for automaed qualitative abstraction. In *15th International Workshop on Qualitative Reasoning, QR-01*, San Antonio, Texas (USA), 2001.

[Thétiot *et al.*, 1998] R. Thétiot, F. Zouaoui, M. Dumas, P Dague, and T. Renaud. Automatic construction of processes from bond graph representation. In *Proc. of the International Workshop on Qualitative Reasoning QR'98*, pages 131–136, Cape Code, (USA), 1998.

[Travé-Massuyès and Pons, 1997] L. Travé-Massuyès and R. Pons. Causal Ordering for Multiple Mode Systems. In *Proc. of the 11th International Workshop on Qualitative Resoning QR'97*, pages 203–214, Cortona (Italia), 1997.