2010 2nd International Conference on Industrial and Information Systems

Modeling Requirements of Automotive Software with an Extended EAST-ADL2 Architecture Description Language

Xiaojian Liu, Xuqin Yan, Chengyong Mao, Xiaobo Che and Zhixue Wang
Shandong Key Lab of Automotive Techniques,
Institute of Automation, Shandong Academy of Science
Jinan, China
liuxjian@keylab.net

***Abstract*—Modeling software requirements is one of the grand challenges of ECU (Electronic Control Unit) development for vehicular applications. One of the main problems in this domain is: How to organize the large amount of complicated software requirements with a concise and manageable means. EAST-ADL2, as an architecture description language dedicated to automotive electronic systems, is well-suitable for describing the system structures, however it has no facilities for the system behavioral and communication aspects. In this paper, we extend the EAST-ADL2 language with timed automata and signal matrix to form a multiview requirement modeling language. Furthermore, we use a theory of *4-variable requirement model* to link these three formalisms together to form a consistent and complete understanding of automotive software requirements. The proposed modeling framework takes the advantages that: (1) it allows us to separate the whole requirements into three complementary aspects: structure, timing behavior and communication signals, which combined together to form a complete and consistent software requirements; (2) it builds numerous formal relationships, which can be checked furthermore for requirement verifications.**

Keywords- Model-driven development; Requirement modeling; Automotive software; Architecture description language; Timed automata

I. INTRODUCTION

Requirements modeling is one of the grand challenges of the automotive software development[1]. One of the main problems is: How to describe the requirements in terms of concise and manageable formal models; and How to integrate the models to form a consistent and complete understanding about the software to be modeled. As we are all acknowledged, requirements modeling and analysis are the most important and difficult activities in the software development, however, the requirements modeling for the automotive software are particularly difficult due to the reasons: (1) the close interactions with its environment; (2) the distribution over networks; and (3) the complicated timing behavior and the other non-functional properties.

To tackle these problems, we must find a suitable modeling notation which is expected to reflect all the above system characteristics. However, it is hard to find a single modeling notation which could cover all the above software aspects. Considering an example of the BCM (Body Control Module) control unit in a vehicle. To model its software requirements, we may choose the finite state machine (FSM) formalism to model its reactive behavior, but FSM is neither suitable for capturing the structural aspect, i.e., input/output ports, nor the communication aspects, such as signals exchanged between the BCM and the other ECUs. You, may certainly put all the structural and communication information on the FSM model, but this practice will definitely make the model's appearance cumbersome, destroying the model's conciseness and manageability.

To model the automotive software requirements in a concise and manageable way, we adopt the following strategy: first, separate the requirements into various aspects, and then, choose suitable modeling notations to model these different aspects, and finally, we need unify these aspects to formal a complete and consistent understanding of the software requirements.

We separate the requirements of automotive software into three aspects: *structural*, *behavioral* and *communication*, and choose *EAST-ADL2*[2], an architecture description language dedicated to automotive electronic systems, *timed automata*[3] and *signal matrix* to describe them respectively. In order to integrate these different aspects, we map the aspectual models to the variables and relations in the *4-variable requirement model*, a well-acknowledged relational model of requirements, to form a complete and consistent understanding of the requirements.

The rest of the paper is organized as follows: Section II introduces the modeling notations for different software aspects; Section III briefly introduces 4-variable requirement model, and presents how to unify aspectual models into an integral model; Section IV compare our work with a number of related works, and finally Section V concludes this paper.

II. MODELING NOTATIONS

To capture a comprehensive understanding of automotive software requirements, we need three kinds of information from requirement documents: (1) *Structural information*, which specifies the interface information, i.e., input and output ports of a functionality, and the decomposition of a functionality; (2) *Behavioral information*, which captures how a functionality react to the stimulus coming from the environment. Since the majority of automotive systems are hard or soft real-time systems, we need a formalism which has the ability to model timing behavior; (3) *Communication information* specifies the properties of communication media. In automotive systems, communication medias may be network (CAN bus, for example), serial transmission lines or some wireless channels (for example, bluetooth, WiFi etc.). However, at the requirement level, no matter what media is used, we always pay more attentions to the signals, such as signal name, signal priority (if the signal is

978-1-4244-8217-7/10/$26.00 ©2010 IEEE

transmitted by CAN bus), signal period, signal data type, and signal values etc..

These three kinds of information reside explicitly or implicitly in the requirement documents, and should be extracted through the careful reading and understanding of the requirement documents.

To facilitate the understanding of our idea, in the following, we will use the BCM (Body Control Module) as the running example. The BCM is an ECU which controls the vehicle body components, including doors, tailgate, hood, windows and lights etc. Note that, all the controls performed by the BCM are discrete, and no complicated vehicle continuous dynamics are involved in. Therefore, we need not introduce hybrid models in the requirements modeling.

EAST-ADL2

We choose EAST-ADL2 as the modeling notation for the structural aspect. EAST-ADL2 is an architecture description language specific to automotive systems. It describes a system from different abstraction levels: vehicle level, analysis level, design level and implementation level. Here we only use a subset of the language, i.e., the language for the function modeling, to model the structure of a functionality. The following Figure 1 illustrates the UML metamodel (i.e., the modeling concepts and their relations) for the function modeling.

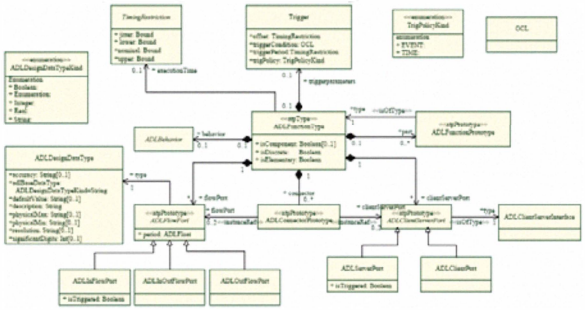

Figure 1 Metamodel of EAST-ADL2 for the function modeling

In EAST-ADL2, a functionality is modeled as an element of **ADLFunctionType**, which consists a set of ports (the elements of **ADLFlowPort** or **ADLClientServerPort**), a set of function prototypes (the elements of **ADLFunctionPrototype**). Any two function prototypes, or a function type and one of its function prototypes, can be connected with a group of connectors (the element of **ADLConnectorPrototype**) whose two ends are of ports. Each function prototype must associate an element of **ADLFunctionType** as its type. Figure 2 gives the structural model of BCM.

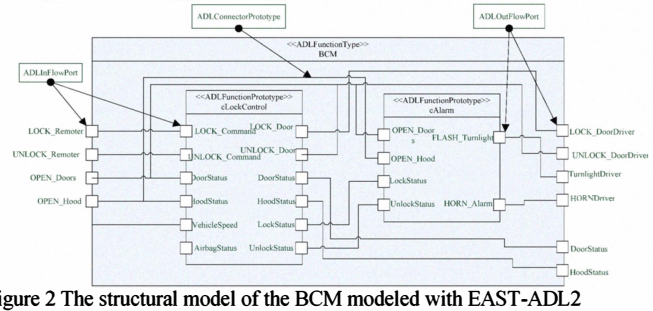

Figure 2 The structural model of the BCM modeled with EAST-ADL2

In Figure 2, the top level is BCM's functionality, which is modeled as an element of **ADLFunctionType**. **cLockControl** and **cAlarm** are of function prototypes of the BCM. Connectors of the BCM are classified as two categories: *delegate* connectors and *assembly* connectors, the former connect the ports between a function type and its prototype, while the latter connect the ports between function prototypes. Every port has a period attribute, and is associated with a data type, which restricts what data is allowed to be exchanged over the port.

Note that the structure mode is hierarchical. For example, when we want to observe the internal structure of the BCM, we view it as a white-box, and its structure can be illustrated as Figure 2. In this context, each function prototype of the BCM, say **cLockControl**, is viewed as a black-box, in other words, its internal structure is unobservable to observers. When we want to know the structure of **cLockControl**, we should return to the structure model of its associated function type. Obviously, following this way, we can form a structure hierarchy.

Timed Automata

The majority of automotive control systems are real-time systems which, for certain inputs, has to compute the corresponding outputs within given time bounds. Timed automata is an extension of Buchi automata with real-valued clocks. Currently, there are rich tool supports for timed automata, such as UPPAAL[4], TIMES[5], which allow us to verify the requirements automatically. Therefore we choose UPPAAL version of timed automata as the formalism to model the behavioral aspect of the requirements.

Let C be a set of clocks, $B(C)$ is the set of conjunctions over simple conditions of the form $x \sim c$ or $x-y \sim c$, where $x, y \in C$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, >, \geq\}$. A timed automata is a tuple (L, l_0, A, E, I), where L is a set of locations, $l_0 \in L$ is the initial location, A is a set of actions, co-actions and the internal actions, $E \subseteq L \times A \times B(C) \times 2^C \times L$ is a set of edges between locations with an action, a guard and a set of clocks to be reset, and $I: L \rightarrow B(C)$ assigns invariants to locations.

UPPAAL modeling language, one of the versions of timed automata, extends timed automata by allowing both clocks and bounded integer variables (or arrays of these types) to appear in the expressions. This extension enhances the expressive ability of timed automata, allowing us to model the complicated guard conditions, assignments and invariants.

The following Figure 3 shows the time automaton model, edited by UPPAAL tool, for the BCM door locking/unlocking functionality.

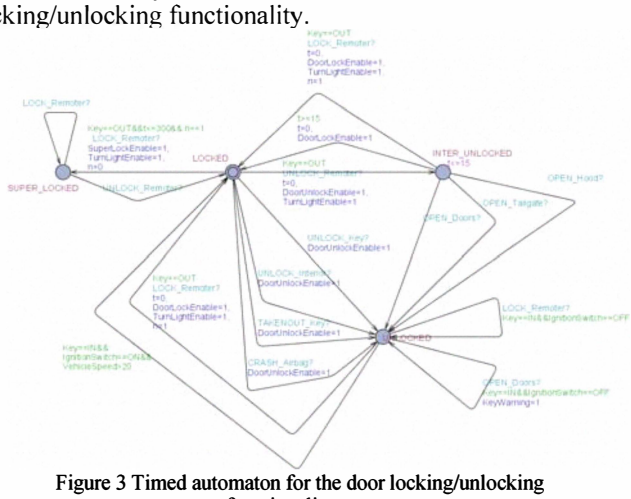

Figure 3 Timed automaton for the door locking/unlocking functionality

Table 1 Signal matrix of the BCM

name	*period*	*priority*	*size*	*unit*	*sender*	*receiver*	*resolution*	*offset*	...
LockControlOfDoor	100ms	-	2	bit	sensor	BCM	N/A	0	…
OpenControlOfDoor	100ms	-	2	bit	sensor	BCM	N/A	0	…
MasterVehicleSpeed	20ms	0x088	8	KPH	EMS	BCM	1KPH	0	…
ActualEngineTorque	10ms	0x082	12	NM	EMS	TCU	0.5NM	-100	…
DoorLock	150ms	-	1	bit	BCM	actuator	N/A	0	…

Signal Matrix

At the requirement level, the communication aspect of the automotive software is mainly about signals and their properties, not matter what communication media and what techniques are used. Signals are of the strings of bits, serving as the inputs or outputs of the software modules. Input signals may come from sensors, networks or some hardware/software drivers, while output signals will be sent to actuators, networks or some hardware/software drivers.

The properties of a signal include: *name*, *period*, *priority* (if the signal is transmitted through CAN bus), *size* (in number of bits), *unit*, *sender*, *receiver*, *resolution*, *offset*, *range of values*, *default value* and *value descriptions*. Here, we use signal matrix, a two-dimension table, to represent all the relevant signals and their properties.

Table 1 shows a part of the signal matrix in the BCM application. Some signals, such as MasterVehicleSpeed and ActualEngineTorque come from CAN bus, therefore they have the priorities, which used by CAN controllers to arbitrate message conflictions. The *resolution* and *offset* properties build a relationship between the real values and the encoded values. For example, if the value of signal ActualEngineTorque is 0x100, then its real value is

$$-100+256\times 0.5=28\text{NM}.$$

III. INTEGRATE ASPECTUAL MODELS

In this section, we will discuss how to combine these three different aspectual models (EAST-ADL2 model, timed automata and signal matrix) to an integral software requirement model.

4-variable requirement model

We apply the theory of 4-variable model [6] to explain how to integrate the different notations to form a sound requirement specification. An overview of the 4-variable model is shown in Figure 4.

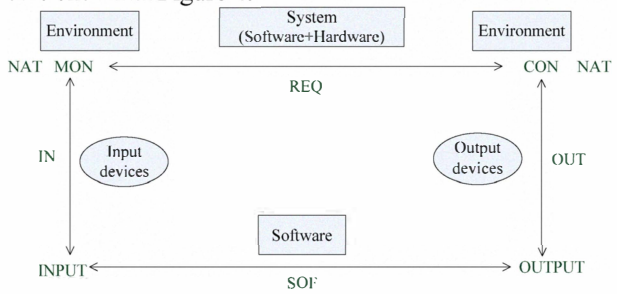

Figure 4 4-variable requirement model

According to this theory, a sound requirement specification for an embedded system should be specified in terms of four groups of variables and five kinds of relations. The variables in this model are functions of time: (1) Monitored variables **MON**, represents an environmental quantity that influences system (including both hardware and software) behavior; (2) Controlled variables **CON**, an environmental quantity that the system controls; (3) Input variables **INPUT**, the boundary of the software, represent the values that the devices write to the software; and (4) Output variables **OUTPUT**, the boundary of the software, represent the values that the devices read from the software.

The five mathematic relations are of: (1) **NAT** defines nature laws or physical constraints imposed on the variables of **MON** and **CON**. For example, a reasonable constraint on the values of engine torque is: -100NM ~1947NM; (2) **REQ**, a relation between **MON** and **CON**, which defines the responses of the controlled variables to the monitored variables; (3) **IN**, a relation between **MON** and **INPUT**, which is an abstraction of a sensor or some kind of input hardware/software device; (4) **OUT**, a relation between **CON** and **OUTPUT**, which is an abstraction of an actuator or some kind of hardware/software output device; and (5) **SOF** is a relation between **INPUT** and **OUTPUT** which is an abstraction of the software behavior.

Unify the aspectual models

In what follows, we argue that the three aspectual models actually specify the 4 groups of variables:

- The input and output ports in a structural model correspond to the monitored variables **MON** and the controlled variables **CON**;
- The signals in a signal matrix actually specify the variables in **INPUT** and **OUTPUT**.

Furthermore, the aspectual models also represent a number of relations:

- **NAT** constraints imposed on each monitored and controlled variable are represented by the data types, which associated with ports;
- **REQ** is actually represented by the timed automaton model.

Therefore, to unify the aspectual models to a sound requirement model, we need build the rest of the relations: **IN**, **OUT**, and **SOF** from the aspectual models. We use the models of the BCM to show how to build these relations.

Assume BCM's structural model, behavioral model and signal matrix are Figure 2, Figure 3 and Table 1 respectively. (Note that, due to space reason, Figure 2 and Table 1 only illustrate a part of ports and signals, i.e., some ports and signals in the following may not appear in these models.) We use the following two tables to demonstrate the relations **IN** and **OUT**.

From these two tables we can easily observe that: relation **IN** and **OUT** can be simple 1-1 mappings or can be more complicated data decomposition relation. For example, the variable DoorStatus is decomposed to two signals DoorAjarStatus and DoorLockStatus.

MON	**INPUT**	**IN**
LOCK_Remoter:Bit[1]	RemoterLockSwitch: Bit[1]	LOCK_Remoter= RemoterLockSwitch
UNLOCK_Remoter:Bit[1]	RemoterUnLockSwitch:Bit[1]	UNLOCK_Remoter= RemoterUnLockSwitch
VehicleSpeed: Real	MasterVehicleSpeed: Bit[8]	VehicleSpeed= $r\times$ MasterVehicleSpeed where r =1KPH is the resolution of the speed

CON	OUTPUT	OUT
LOCK_Door: Bit[1]	DoorLock: Bit[1]	LOCK_Door=DoorLock
DoorStatus: Bit[2]	DoorAjarStatus:Bit[1] DoorLockStatus: Bit[1]	If DoorAjarStatus=0 then DoorStatus=0 If DoorAjarStatus=1 and DoorLockStatus=0, then DoorStatus=1 If DoorAjarStatus=1 and DoorLockStatus=1, then DoorStatus=2

The relation **SOF** can be built from the timed automaton model (Figure 3), **IN** and **OUT**. Obviously, because the variables appearing in the guard expressions of the timed automaton belong to **MON**, and they can be substituted by the input signals via relation **IN**; Likewise, the variables appearing in the assignments can be substituted by the output signals via relation **OUT**. The synchronization actions in the timed automaton can be also represented by the input signals. For example, the action LOCK_Remoter? (See Figure 3) represents an event of "press LOCK button on a remoter", it can be represented by the input signal as:

@T(RemoterLockSwitch=1)

where the notation @T(*condition*) is borrowed from the work[7], denoting such an event that *condition* becomes **True**.

Summarily, the relation **SOF(INPUT, OUTPUT)** can be derived by the transformation of **REQ(MON,CON)** through the relations **IN(MON, INPUT)** and **OUT(CON, OUTPUT)**.

Therefore, based on the theory of 4-variable requirement model, we can establish a corresponding relationship from the aspectual models to the variables and relations of the 4-variable model. Thus, the aspectual models are integrated together to provide a complete and consistent understanding of the whole system.

IV. RELATED WORKS

We mainly compare our work with the following two works: SaveCCM model[8] and EAST-ADL2.

SaveCCM is a component-based design methodology specific to automotive software development. Like our work, it also separates a software into structural and behavioral aspects. It employs SaveCCM (SaveComp Component Model) graphic language to describe the software structure, and also use timed automata to describe the timing behavior. However, our work takes the following two advantages over SaveCCM. First, we take communication information into account. Network technologies are widely used in modern automotive systems, a functionality usually crosses over several different ECUs, communication signals take essential roles in the collaboration between functionalities. We use signal matrix to describe the communication aspect, and build a relation between the signals and the ports, this will facilitate the further software design activities; Secondly, our work mainly concerns about requirement modeling of automotive software, we present a consistent and sound semantic base, i.e., 4-variable model, for the integration of these three aspectual models, this will allow us to verify the requirements furthermore.

As mentioned previously, EAST-ADL2 language is only suitable for describing the structural models without language elements for behavioral modeling. Our work presents a possible extension to the EAST-ADL2 with timed automata formalism, and builds a close relationship between the structural and behavioral models. Combining these two formalisms will definitely improve the ability of EAST-ADL2, and allow us to model, simulate and verify the software behavior in the early phase of system development.

V. CONCLUSION

In this paper, we introduce an extension of EAST-ADL2 with timed automata and communication signal matrix, to form a complete and consistent requirement modeling framework for automotive systems. EAST-ADL2 is a widely used architecture description language in automotive domain, timed automata is a rich tool-supported formalism for timing behavior, and signal matrix presents the communication information. Integration of these three models allows us to model software requirements in a concise and manageable means. Furthermore, we unify these three models to the 4-variable model, this actually explain the aspectual models as variables and relations, which can be checked furthermore for requirement verifications.

REFERENCES

[1] M. Broy, I. Kruger, A. Pretschner and C. Salzmann. Engineering Automotive Software. Proceedings of THE IEEE. 95(2): 356-373, Febrary 2007.

[2] EAST-ADL2: http://www.atesst.org/

[3] R. Alur and D. L. Dill. A theory of timed automata. Theor. Comput. Sci., 126(2):183-235, 1994.

[4] Gerd Behrmann, Johan Bengtsson, Alexandre David, Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal implementation secrets. In Proc. of 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems, 2002.

[5] Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. Times – a tool for modelling and implementation of embedded systems. In TACAS 2002, volume 2280 of Lecture Notes in Computer Science, pages 460–464. Springer–Verlag, April 2002.

[6] Parnas, D. L. and Madey, J. 1995. Functional documentation for computer systems. Sci.Comput. Program. 25, 1 (Oct.), 41–61.

[7] Heitmeyer, C. L., Bull, A., Gasarch, C., and Labaw, B. SCR*: A toolset for specifying and analyzing requirements. In Proceedings of the 10th Annual Conference on *Computer Assurance* (*COMPASS '95*) (Gaithersburg, Md., June). IEEE, New York, 109–122.

[8] Hansson, H., Akerholm, M., Crnkovic, I., Torngren, M.: SaveCCM - A component model for safety-critical real-time systems. In: 30th EUROMICRO Conference 2004, 31 August -3 September 2004, Rennes, France, IEEE Computer Society (2004) 627–635