Global Journal of Researches in Engineering: J General Engineering
Volume 18 Issue 4 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

Function Allocation and Bandwidth Reservation for Mixed-critical Adaptive Software Systems

Mahmoud Hussein
Menoufia University

Abstract- The new Auto SAR adaptive platform makes mixedcritical automotive systems able to adapt themselves in response to hardware and software failures at runtime. However, mapping functions of these automotive systems and reserving bandwidth for them are still major challenges. In this paper, we propose a model-based approach for mapping functions of an automotive system to its hardware nodes and reserving their bandwidth. To do so, an architecture description language for automotive systems (i.e. EAST-ADL) is used to design an embedded system, and to specify its timing requirements. The design model is then used for identifying functions allocation and their bandwidth in different system configurations.

Keywords: mixed-criticality; design space exploration, adaptive system; schedulability analysis; model-based.

GJRE-J Classification: FOR Code: 090299

Strictly as per the compliance and regulations of:

Function Allocation and Bandwidth Reservation for Mixed-critical Adaptive Software Systems

Mahmoud Hussein

Abstract- The new Auto SAR adaptive platform makes mixed-critical automotive systems able to adapt themselves in response to hardware and software failures at runtime. However, mapping functions of these automotive systems and reserving bandwidth for them are still major challenges. In this paper, we propose a model-based approach for mapping functions of an automotive system to its hardware nodes and reserving their bandwidth. To do so, an architecture description language for automotive systems (i.e. EAST-ADL) is used to design an embedded system, and to specify its timing requirements. The design model is then used for identifying functions allocation and their bandwidth in different system configurations. To schedule the critical functions of the system, the Earliest Deadline First (EDF) is used, while the Constant Bandwidth Server (CBS) is used for scheduling the non-critical functions. The quality of service for the non-critical functions is determined by their reserved bandwidth. In addition, a Tabu search-based approach is used for mapping the system functions to hardware nodes. Furthermore, there is a temporal isolation between the critical and non-critical functions. Thus, overruns of the non-critical functions do not affect the timing guarantees of the critical functions, and the quality of service for the non-critical functions is maximized.

Keywords: mixed-criticality; design space exploration, adaptive system; schedulability analysis; model-based.

I. Introduction

With the advances in micro-electronics, embedded system engineers are now able to integrate more system functions on a powerful System-on-Chips (see Figure 1)[1]. The automotive industry also benefits from these advances, where the engineers become able to integrate advanced vehicle functions on high performance electronic control units (ECUs).These functions can be classified as critical and non-critical functions. Thus, mixed-criticality concept has been introduced, where vehicle functions have different criticality levels as shown in Figure 1[2]. In addition, the new Auto SAR adaptive platform makes automotive systems able to adapt themselves at runtime to cope with hardware and software failures [3]. A major challenge is how to map functions of an adaptive systems to its hardware nodes, and to reserve bandwidth for these functions [4].

Author: Faculty of Computers and Information, Menoufia University, Egypt. e-mail: mahmoud.hussein@ci.menofia.edu.eg

In recent years, a number of approaches has been proposed for mapping a system's functions to its processing units, and for reserving their bandwidth to ensure that they are going to meet their deadlines at runtime (e.g. [5] [6] [7] [8] [9]). These approaches are aiming at functions mapping and bandwidth reservation for systems that do not have runtime adaptability. However, the new vehicle systems need to adapt themselves in response to hardware and software failures while they are in operation [10] [11]. To cope with system failures, a number of system configurations need to be specified as reactions to these failures. In addition, functions mapping and their bandwidth reservation in each system configuration need to be defined. Consequently, adopting the existing approaches for specifying a system's different configurations, and defining its functions mapping and their bandwidth reservation is difficult and error-prone task, where these approaches have not been proposed for adaptive systems.

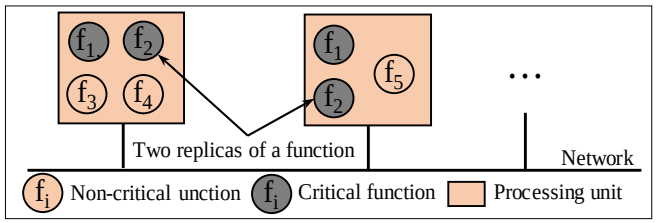

Fig.1: Multiprocessor architecture for a system with mixed critical-functions

In this paper, we propose a model-based approach to ease the specification of the different configurations of an adaptive system, and to identify functions mapping and their bandwidth reservation in each configuration. First, we use EAST-ADL (an architecture description language for automotive embedded systems [12]) for creating the system design model. This model captures the system's functionality, timing requirements, and adaptive behavior. To model the adaptive behavior, we adopted the state machine approach [13]. In this machine, the states are corresponding to the system configurations, and the transitions represent adaptations between these configurations. To capture the timing requirements, we have adopted TIMMO (TIMing MOdel) approach [14].

Second, the system design model is used for identifying the functions allocation and their bandwidth in the different system configurations automatically. To

schedule critical functions, Earliest Deadline First (EDF) [15]is used, while Constant Bandwidth Server (CBS) [16]is used to schedule non-critical functions. Quality of service for the non-critical functions is determined by their reserved bandwidth. In addition, a Tabu search-based strategy is used to map functions to the hardware nodes[17]. Furthermore, there is a temporal isolation between critical and non-critical functions. Thus, overruns of the non-critical functions do not affect timing guarantees for the critical ones, and the quality of service for the non-critical functions is maximized. To show the applicability of our approach, a case study using it is performed in the context of Safe Adapt project.

The remainder of the paper is organized as follows. A short description of related work is given in Section II. Our approach for designing an embedded system, and for functions mapping and their bandwidth reservation is described in Section III. In Section IV, we present our approach implementation. Finally, we conclude the paper in Section V.

II. Related Work

The work introduced in this paper is related to two research areas: designing adaptive systems, and functions mapping and their bandwidth reservation. In the following, we describe the related work from these two angles.

a) Designing Adaptive Systems

Rainbow framework provides mechanisms for monitoring the environment, performing the analysis of the environment to initiate the adaptation process, selecting the required adaptation strategy, and effecting the needed changes to a running system[18]. To capture the system reactions to environment changes, they used a language called *Stitch*. Sheng et al. have proposed a model-driven approach to ease the development of context-aware web services[19]. In their approach, they consider the system functionality as a single service, and the environment information is used by a set of rules to adapt the service output parameters in response to environment changes.

An approach was introduced by Zhang and Cheng to create formal models of a system behaviour[20]. In this approach, the system adaptive behaviour is separated from its non-adaptive behaviour. This separation makes the system models easier to specify and verify. They used Petri-nets to capture the system's adaptive behaviour, where they use context change as guidance for the transition between system states. The *SOCAM* (Service-Oriented Context-Aware Middleware) project introduces an architecture for building adaptive systems[21]. It uses a central server to gather context information from distributed context providers. This information is then processed, so that it can be used by the system functionality.

MUSIC project is a component-based framework that is used to optimize a system overall utility in response to context changes[22]. They have a quality of service (QoS) model that describes the system composition together with the relevant QoS dimensions, and how they are affected when the system is going to change from one configuration to another. The quality of service model is used for selecting a new configuration that has the best utility and is able to cope with the context changes. Heaven et al. have developed an approach to adapt a system in response to environment changes while preserving its high level goals[23]. They use Labelled Transition Systems (LTS) to capture the system states and the environment situations.

Andrade et al. have proposed an approach to cope with unanticipated changes of an adaptive system behaviour [24]. They separate the system adaptation from its functionality, and represent the adaptation logic as a set of condition-action rules. These rules are constructed as a component-based system that can be changed at runtime. Morin et al. proposed a technique to handle the exponential growth of the number of configurations that are derived from the system variability[25]. They combine model driven and aspect oriented approaches to cope with the complexity of adaptive software systems.

b) Function Mapping and Bandwidth Reservation

A technique that uses Constant Bandwidth Server (CBS) for integrating critical and non-critical functions on the same processor has been introduced by Abeni and Buttazzo [8]. This technique uses scheduling algorithms such as Earliest Deadline First (EDF) or Rate Monotonic (RM) to guarantee meeting the deadlines of critical functions. The non-critical functions are scheduled by a number of servers. A server parameters (i.e., its bandwidth and period) determine the probability of meeting the deadline of a non-critical function (i.e. its quality of service). The approach is also extended to adjust the server parameters by proportional integral derivative (PID) controllers at runtime. The idea behind this is to maximize the QoS of the non-critical functions [26]. Offline and online approaches to derive CBS parameters have been also proposed [27]. They are aiming at increasing the probability of meeting deadlines of the system non-critical functions.

A Tabu search-based algorithm has been introduced for performing functions mapping and their bandwidth reservation [28]. This algorithm considers mixed-critical real-time systems that should tolerate transient faults. It uses EDF to schedule critical functions and CBS to schedule non-critical ones. The faults are tolerated through defining check points with rollback recovery mechanism [29]. It also uses the probability density functions for the non-critical functions. Thus, decisions of the mappings and processor bandwidth allocations are improved.

In the context of systems that have mixed time triggered (TT) and event triggered (ET) functions, an approach has been introduced [30]. This approach schedules the TT functions by static-cyclic scheduling (SCS), while the ET functions are scheduled using fixed-priority scheduling (FPS). It can be also be extended to constrain the TT schedules by following a given partitioning. The problem of mapping and partitioning have been addressed [31]. However, the partitioning means deciding which functions are TT and which are ET.

Another approach to address the mapping of mixed critical real-time functions on distributed embedded architectureshas been introduced [32]. It assumes that the architecture provides spatial and temporal partitioning. Therefore, enough separation between functions are enforced. In temporal partitioning, each function executes in a separate partition. Each partition is also allocated a set of time slots on a processor (where the function is mapped). Time slots of all functions on a processor are also grouped within a major frame that is repeated periodically. The functions are scheduled using static-cyclic scheduling.

The approaches discussed above are focusing on functions mapping and bandwidth reservation for embedded systems that do not adapt at runtime. However, new vehicle systems need to adapt themselves in response to hardware and software failures with the support of the new Auto SAR adaptive platform [3][10] [11]. To cope with failures of such systems, a number of system configurations need to be specified as system reactions. In addition, functions mapping and their bandwidth reservation for each configuration need to be identified. Therefore, using the existing approaches for specifying the system's different configurations, and identifying the functions mapping and their bandwidth reservation is difficult and error-prone tasks.

III. The Proposed Approach

To ease the process of mapping functions of an embedded adaptive system to its hardware nodes, and for reserving their bandwidth, we have proposeda two-step process. In the first step, a model for the system is specified. This model includes the system's functionality, timing requirements, and adaptive behavior. In step two, based on the system model, mappings of the system functions in each system configuration and their bandwidth are identified by schedule ability analysis techniques. In the following, we describe the two steps in detail.

a) Modelling Adaptive Embedded System

To model a safe adaptive system, three aspects need to be captured: the system functionality, its timing requirements, and its adaptive behavior. The adaptive behavior specifies system reactions to anticipated changes such as hardware failures.

The System Functionality: The system functionality consists of functions that interact with each other to meet user requirements (see Figure 2). To model such functionality, an architecture description language for automotive domain (i.e. EAST-ADL [12]) is used. In EAST-ADL, the architecture is modeled at two levels of abstraction: an abstract functional model and a refined in form of a design model. Both levels are modelled as a composite structure that consists of components that interact with each other through functional ports. In our approach, we use cardinality of system components to specify the system variability. A component with cardinality {0 or 1} is optional while cardinality {2} means it has two instances and the system can switch between them at runtime. A cardinality {1} specifies that the component is mandatory and should always exist while the system is in operation (i.e. permanent).

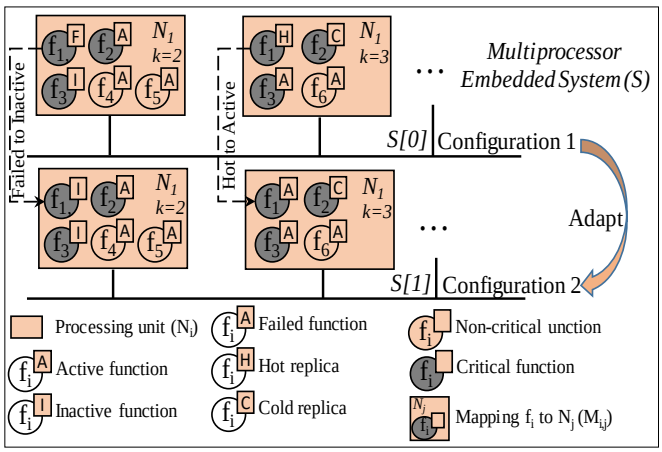

Fig. 2: A representation of an adaptive embedded system

A component may have two instances to increase the system availability and to ensure that the functionality of this component is always provided [33]. These two instances need to be allocated to different processing units, so that a failure does not lead to a total failure of the functionality (e.g. function f_1 in Figure 2). To specify such allocation, the system hardware need to be modelled and the functional allocation can be then defined. Similar to the system functions, the hardware model is specified using EAST-ADL modelling language. It is modelled a composite structure that contains the hardware elements such as processing units, sensors, actuators, etc.

The System Timing Requirements: To model the system's timing requirements, we have adopted the technique proposed in TIMMO-2-USE [14] (TIMing MOdel - TOols, algorithms, languages, methodology, and USE cases) project. To specify a timing requirement, events that are associated with a software function or

one of its ports are defined. These events are then used for defining timing constraints such as execution time constraint, periodic constraint, reaction constraint, etc. Using TIMMO, the execution time of a function is specified by the Execution Time Constraint concept where the start and the stop events of the function are defined with lower and upper bound for the delay between them (see Figure 3). Similarly, to specify a periodic constraint, an event that is associated with a function is defined. Then, the Periodic Constraint concept is used, where a periodic constraint is specified that references this event. This constraint specifies the inter-arrival time and the period of a function (e.g. 60 milliseconds).

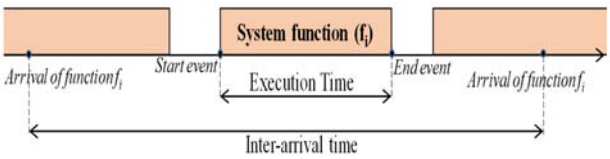

Fig. 3: Representation of timing constraintsof a function f_i

The System Adaptive Behavior: To adapt the embedded system in response to context changes, we introduce a system management component [34]. This component switches from a system configuration to another in response to an adaptation trigger (e.g. a failure of the function f_1 as shown in Figure 2). Therefore, we need to model adaptation triggers and different runtime configurations (states) of the system. Both represent a runtime system state. To model a system state, we adopted the concept of UML instance specifications where a system design instance (i.e. the system's functionality and hardware platform) is created and configured to specify a system runtime state or an adaptation trigger [35].

In order to model an adaptive behavior of a system, we adopted the state machine approach [13]. This technique makes adaptation policies easy to understand and it is useful for validation and verification purposes. In this machine, states are corresponding to the system configurations, while transitions represent the adaptations between these configurations. Each transition is guarded and triggered by an adaptation event. For example, in response to a function failure during a specific state, the system moves from its state or to a configuration that recovers from this failure as shown in Figure 2.

b) Task Mapping and Bandwidth Reservation

To identify functions mapping and bandwidth reservation of an embedded system, the design model is used as a base for doing that (i.e. Step 2). In the following, we first describe the formulation of functions mapping and bandwidth reservation problem. Then, we describe an approach to solve this problem.

The problem can be formulated as follows. Given a mixed critical system (S), and a distributed architecture (N) with a maximum number of transient faults (k) (an example system is shown in Figure 2). We are interested in determining a solution (L) consisting of a mapping M (f_i) ∈ N for each function f_i∈ S, and a set B containing the bandwidth $B_i$for each function f_i. Thus, the deadlines for critical functions are satisfied even in the case of transient faults. In addition, the probability for meeting the deadlines of non-critical functions is maximized.

To solve the above problem, we have used a Tabu search-based strategy. Tabu search is an optimization metaheuristic [17]. It explores iteratively solutions in the neighbourhood of current solution to select a solution that minimizes the cost function. The cost function we use (described later) captures schedulability of critical functions and quality of service (QoS) of non-critical functions. The minimization of the cost function aims to improve schedulability of the critical functions and to maximize the QoS for the non-critical functions.

Our Tabu search-based strategy is described in Table 1 (an extension for the algorithm described in [28]). The input is an adaptive system model in a form of configurations as described above (S), the hardware nodes (N), and maximum number of iteration for our strategy (MaxI). The output of the algorithm is a number of solutions (L) for the system configurations. The solutions consist of functions mapping M, and the set of bandwidth B for all critical and non-critical functions in each system configuration.

Table 1: Tabu search-based strategy for task mapping and bandwidth reservation

Input: S: Adaptive system, N: Distributed architecture, MaxI: Maximum number of iterations.

Strategy:

1: c = 0;
2: ***foreach*** configuration C: S[c] ***do***
3: ***if*** (c > 0) L_0 = GenerateInitialSolution (L[c-1]);
4: ***else*** L_i= GenerateInitialSolution(S[c], N);
5: ***end if***
6: $L_{current}$ = L_{best} = L_0;
7: $Cost_{best}$= CostFunction(L_0);
8: TabuList = ***new*** List;
9: ***While*** iter < MaxI or $Cost_{best}$== 0 ***do***
10: NL = GenerateNeighborhood(L_{best});
11: $L_{current}$= SelectSolution(NL);
12: ***if*** Cost Function($L_{current}$) <$Cost_{best}$ ***then***
13: $Cost_{best}$= CostFunction($L_{current}$);
14: L_{best}= $L_{current}$;
15: ***end if***
16: Add ($L_{current}$, TabuList);
17: ***end while***
18: L[c] =L_{best};
19: c = c+1;
20: ***end for***

Output: L: Solutions for the system configurations

Our strategy starts by generating an initial solution L_0(see Table 1: Line 3-4). In this solution, the functions are allocated randomly to the hardware nodes, and critical tasks are assigned a bandwidth equals to their worst case execution time (WCET) while the non-critical functions assigned a bandwidth equals to their average execution time (AET). The initial solution can be schedulable or not. A schedulable solution is the one that all critical functions meet their deadlines. The initial solution is used as a starting point in finding a better solution for a system configuration (state). However, in the case of it is not the first state, an initial solution is generated that takes into account the previous solution as shown in Table 1: Line 4.

The strategy then starts to search the neighbourhood of the current solution for finding a better solution. New solutions are generated either by changing functions mappings or through increasing/reducing the bandwidth of the non-critical functions. The neighbourhood can be very large. Thus, we only consider a limited number of solution in each iteration. The generated solutions are evaluated by computing their costs. The one with the lowest cost is then selected as current solution (Lines 9-16 in Table 1). This process is repeated until maximum number of iterations is reached or the cost becomes zero. After finding a solution for a system configuration, the strategy is repeated for finding solutions for other system states which is the output of our strategy.

One feature of Tabu-search based techniques is the storage of solutions history that are visited (called *Tabu List* in our strategy). The idea behind this list is to avoid revisiting already explored solutions. The solutions history is initialized in Line 8 of our strategy, and updated with the currently visited solution at Line 16.

To compute the cost of a solution, schedulability analysis for critical and non-critical functions needs to be calculated. In the following, we describe these calculations in detail.

Schedulability Analysis for Critical Functions: To analyse schedulability of the critical functions and content bandwidth servers for the non-critical functions (describe below), we use a utilization-based test. The utilization of a hardware node N_j is computed by Equation 1 (below)[36]. In this equation, first, $C`_i$ is worst case execution time (WCET) of a critical function f_i allocated to the hardware node N_j. To compute the WCET while considering failures possibility, we use a number of checkpoints (n_i) together with checkpoints overhead (o_i), error detection overhead (e_i), and the function's execution time as shown in Figure 4-A: $C`_i = C_i + (n_i - 1)(o_i + e_i) + e_i$ [28].

$$\sum_{\substack{\forall f_i : R(f_i)=Critical \\ \wedge M_j(f_i)=N_j}} \frac{C_i^{'}}{T_i} + \sum_{\substack{\forall f_i : R(t_i)=Non-critical \\ \wedge M_j(f_i)=N_j j}} \frac{B_i}{T_i} + U_R^{N_j} < 1 \qquad (1)$$

Second, T_i is the deadline for a critical or a non-critical function f_i. Third, B_i is the bandwidth that is allocated to a non-critical function. Forth, when a fault occurs during the execution of a function f_i, this function has to be restored from a previously saved checkpoint. Also, an execution segment of length (C_i/n_i) needs to be executed. Therefore, the utilization needed for recovering from a fault is $((C_i/n_i) + e_i + m_i)/T_i$, where m_i is the time required to recover from the error. For a processing unit N_j that can have up to k failures during the system execution, its utilization can be computed following Equation 2[28]. In this equation, the utilization is determined for recovery critical functions. In addition, the worst-case is the occurrence of the k faults, which is corresponding to largest recovery utilization.

$$U_R^{N_j} = \max_{\substack{f_i : R(f_i)=Critical \\ \wedge F(f_i)\neq\phi}} k \times \frac{(C_i / n_i) + e_i + m_i}{T_i} \qquad (2)$$

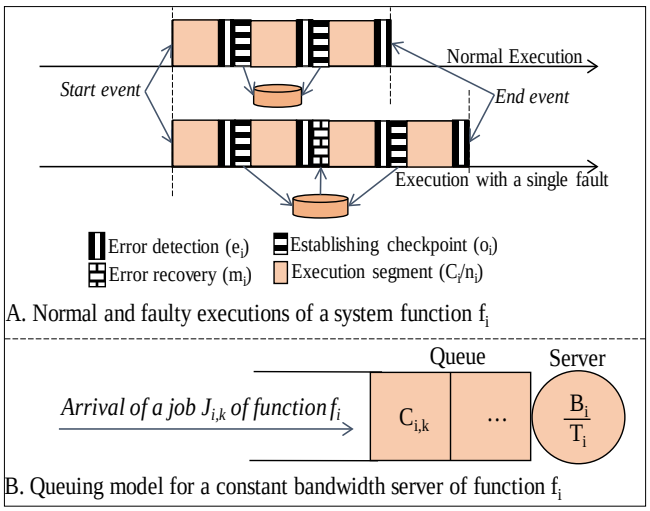

Fig. 4: Execution of a function f_i and a queuing model for its constant bandwidth server

Schedulability Analysis for Non-critical Functions: The schedulability analysis of non-critical functions is probabilities of meeting their deadlines (i.e. their quality of service (QoS)). These probabilities depend on the allocated bandwidths B for them. Because of the temporal isolation property of Constant Bandwidth Server, each non-critical function can be analysed individually. Also, the computation of the system functions' QoS is expensive, and then they are computed and stored to be later used by our strategy.

For a non-critical function f_i, the QoS (f_i) is defined as the probability of meeting its deadline d_i which is P $\{ft_{i,k} \leq rt_{i,k} + d_i\}$. The $ft_{i,k}$ and $rt_{i,k}$ are the finishing and the arrival time of the k^{th} job of the function. To calculate this probability, a CBS that serves the function f_i is modelled as a queuing system [37]. The function jobs J_i are seen as tokens that need to be served by the server having the capacity $B_i$as shown in

Figure 4-B. A Markov matrix is then built and its steady state probability is computed to determine the probability of meeting the deadline of f_i when its allocated bandwidth is B_i (for more detail about these calculations, see [26] and [38]).

In our approach, we consider values of a bandwidth in the interval [*AET, WCET*]. In the case of $Bi \geq WCET$, the deadline will be met in 100% of the cases, while if $Bi < AET$, the probability of meeting the deadline is very small.

Cost Function: In our strategy, the solutions are evaluated based on a cost function that needs to be minimized. The cost function for a solution L is computed using Equation 3. The weight ($w_{penalty}$) is corresponding to a very large penalty added when a critical function is not schedulable (i.e. utilization of a hardware node is more than 1). If critical tasks are schedulable, the first part of the equation is 0. The second part of the cost function is corresponding to maximizing the QoS of the non-critical functions. For each function, a weight (w_i) is assigned to differentiate these functions in terms of their importance.

$$\sum_{\forall N_i \in N} \max(0, U_{N_i} - 1) \times w_{penalty} + \sum_{\substack{\forall f_i : R(f_i) = \\ Non-critical}} (1 - QoS(f_i)) \times W_i \quad (3)$$

IV. Implementation

In this Section, we use the concepts previously explained in Section III to model an adaptive vehicle system, and to identify its functions mapping and their bandwidth reservation. We also describe the tool that supports our approach. This case study has been done in the context of the Safe Adapt project [39].

a) Modelling an Adaptive Vehicle System

As discussed previously to model an adaptive system, there is a need for specifying its functionality, timing requirements, and adaptive behavior. In the following, we discuss the design model of an adaptive vehicle system.

Modelling System Functionality: To design an adaptive vehicle system following our approach, we use the Papyrus UML modeler [40]. Part of the system's functional model is shown in Figure 5. The model consists of a set of functions that are linked with each other through functional ports. In Figure 5, the full adaptive cruise control has the cardinality {2} that means it has two instances. The two instances can replace each other at runtime in case of one's failure. The SomnoAlert is an optional function, i.e., it can exist or not while the system is in operation (the cardinality is {0 or 1}).

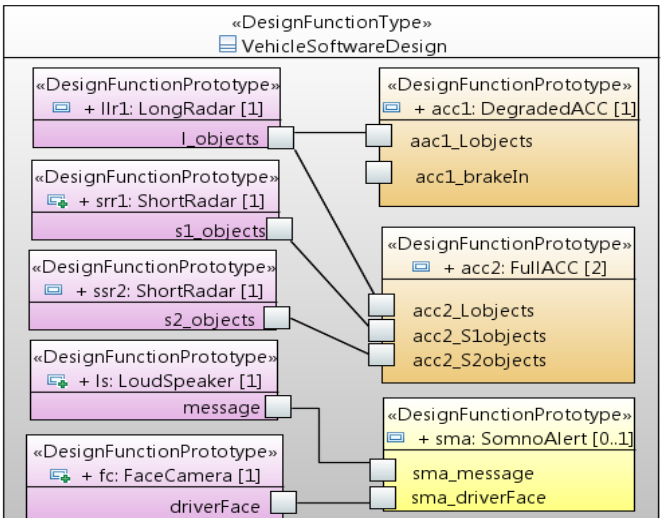

Fig. 5: Part of the design model for an embedded adaptive vehicle system

Similar to the functional model, a hardware model of the system is also designed as a composite structure that consists of two electronic control units (i.e. Delphi TMDP (Trusted Multi Domain Platform) and RACE [41]). In addition, the two units are connected with each other by a hardware connector. The number of failures is also specified for these control units (e.g. "*k = 2*").

Modelling the System Timing Requirements: To model the system timing requirements, we have used TIMMO modelling. For the adaptive vehicle system, a number of constraints have been defined. Some of them are presented in Figure 6. First, to specify a periodic constraint, an event associated with the full cruise control (FullACC) function is defined (i.e. FACCEvent). Then, a periodic constraint is specified that reference this event (see Figure 6). The FullACC function has a minimum inter-arrival time and period of 300 milliseconds. Second, execution time (i.e. 50 milliseconds) of the steering controller function is defined based on the event (SCEvent) as shown in Figure 6.

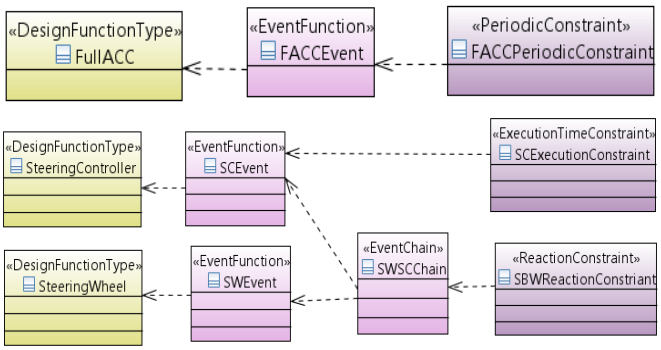

Fig. 6: Example of timing requirements

Using our tool, the different information needed to execute our strategy for functions mapping and bandwidth reservation can be easily defined. For

example, number of checkpoints, checkpoints overhead, and time of error detection and recovery can be specified as attributes of a UML class that represents a system function.

Modelling the System Adaptive Behavior: To model the adaptive behavior of the system, both adaptation triggers and system configurations need to be specified. We model both of them using the instance specification concept. A set of UML in stance specifications that describe component instances along with values for their attributes. Such a set is called deployment plan, a term inspired from the CORBA component model[42]. In our approach, a deployment plan represents a system trigger or configuration (state).

An example of an adaptation trigger modeled as an instance specification is shown in Figure 7 (see the top part). For each function, a runtime state is defined {e.g. Active, Inactive, Hot, Cold, and Failed}. In this example: BBW0, SMA, ACC0, and AEB are in "Active" state, SBW1 is "Hot", BBW1 and ACC1 are in "Cold" state, and SBW0 is "Failed". A system state to recover from the failure of SBW0 is shown in Figure 7 (bottom part). The instance specification for each function is defined as *<Function Name, and State>*. Therefore, this configuration is defined as follows:

{<SBW0, Inactive>, <SBW1, Active>, <BBW0, Active>, <BBW1, Cold>, <SMA, Active>, <ACC0, Active>, <ACC1, Cold>, <ACC2, Hot>, <AEB, Active>}

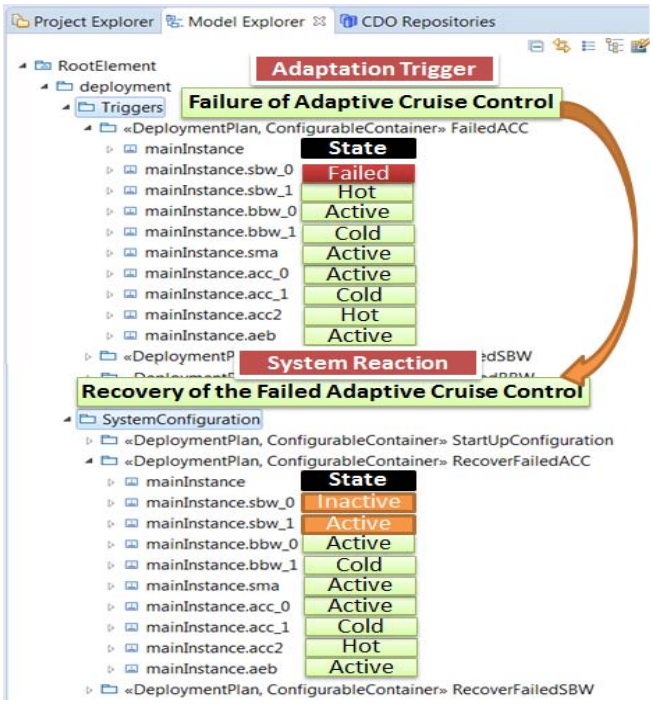

Fig. 7: Specifying an adaptation trigger and a system response

To model the switching between the system configurations in response to the adaptation triggers, a state machine is created as shown in Figure 8. For example, in response to a failure of the steer-by-wire (the adaptation trigger specified on top part of Figure 7), the system adapts from its initial configuration to another that recovers this failure (i.e. the system configuration shown in the bottom part of Figure 7). This system switching is specified using the fourth transition shown in Figure 7 (i.e. "Failure of SBW1"). In this transition, the state of the first instance of the SBW is changed from "Failed" to "Inactive", while the state of the second instance of the SBW is changed from "Hot" to "Active".

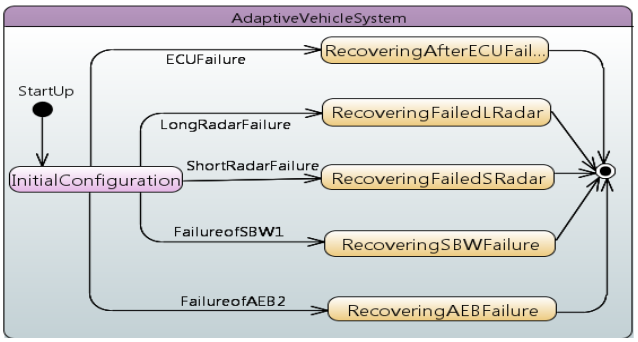

Fig. 8: Adaptive behavior for the vehicle system

b) Task Mapping and Bandwidth Reservation

Based on the timing information specified into the design model (see above), our tool (which implements the strategy described in previous section) finds the allocation and required bandwidth for each system function. An example screenshot of the tool is shown in Figure 9. It determines the allocation for each function. For example the degraded ACC is allocated on TMDP control unit. It also finds the required bandwidth for the degraded ACC which is "53" with QoS of 100% probability of meeting its deadline. The tool also shows how many iterations (e.g. 12) and how long (e.g. 64 millisecond) it takes to find the allocations and to determine the bandwidth reservation.

A main feature of our approach is the consideration of all configurations in identifying the function mappings. Thus, our approach improves existing techniques (e.g. Saraswat et al. [28]), where we find allocations that reduce (limit) the changes of function allocation from a configuration to another as shown in Figure 10. Therefore, at runtime, functions re-allocation is reduced or restricted (if possible), while achieving the best quality of service for the non-critical functions.

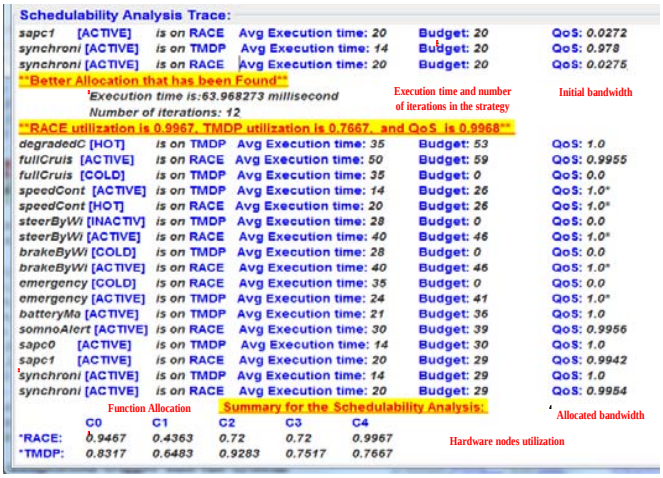

Fig. 9: Output of the strategy for bandwidth reservation

Using Saraswat et al. Approach

	C0	C1	C2	C3	C4
*RACE:	0.93	0.45	0.7133	0.7133	0.93
*TMDP:	0.835	0.5517	0.8367	0.665	0.835
*QoS :	1.0	1.0	1.0	1.0	1.0

**Applications Allocation in the Different Configurations:

	C0	C1	C2	C3	C4
degradedCruis	TMDP	RACE	TMDP	RACE	TMDP
fullCruiseCon	RACE	TMDP	RACE	TMDP	RACE
fullCruiseCon	TMDP	RACE	TMDP	RACE	TMDP
speedControl0	TMDP	RACE	RACE	TMDP	TMDP
speedControl1	RACE	TMDP	TMDP	RACE	RACE
steerByWire0	TMDP	TMDP	RACE	TMDP	RACE

Using our approach

	C0	C1	C2	C3	C4
*RACE:	0.95	0.4329	0.7267	0.7267	0.9967
*TMDP:	0.8283	0.645	0.925	0.7517	0.7633
*QoS :	1.0	1.0	1.0	1.0	1.0

**Applications Allocation in the Different Configurations:

	C0	C1	C2	C3	C4
degradedCruis	TMDP	TMDP	TMDP	TMDP	TMDP
fullCruiseCon	RACE	RACE	RACE	RACE	RACE
fullCruiseCon	TMDP	TMDP	TMDP	TMDP	TMDP
speedControl0	TMDP	TMDP	TMDP	TMDP	TMDP
speedControl1	RACE	RACE	RACE	RACE	RACE
steerByWire0	TMDP	TMDP	TMDP	TMDP	TMDP

Fig. 10: Output of our strategy for function allocation in each system configuration

V. Conclusion

The new AutoSAR adaptive platform makes mixed-critical automotive systems able to adapt themselves at runtime to cope with hardware/software failures. However, mapping functions of these automotive systems and reserving bandwidth for them are still major challenges. In this paper, we proposed a model-based approach for specifying different configurations of an embedded adaptive system, and for defining functions mapping and bandwidth reservation in each system configuration. First, we have used EAST-ADL to create the system design model. This model captures the embedded system functionality, timing requirements, and its adaptive behavior. Second, the system's design model is used as a base for finding functions allocation and for reserving their bandwidth in the different system states (configurations). To schedule the critical functions, the Earliest Deadline First (EDF) is used, while the Constant Bandwidth Server (CBS) is used for scheduling the non-critical functions. Finally, to show our approach applicability, in the context of the Safe Adapt project, a case study has been conducted.

As future work, we plan to extend our approach to enable code generation from the system design model, and automatic deployment of the generated system functions to its hardware nodes. Further evaluations will also be carried out to assess the approach robustness by applying it to a number of case studies.

References

1. U. Abelein, H.Lochner, D. Hahn, and S. Straube, "Complexity, quality and robustness - the challenges of tomorrow's automotive electronics," in Design, Automation, Test in Europe (DATE), Dresden, Germany, 2012, pp. 870-871.
2. Izosimov, P. Pop, P. Eles, and Z. Peng, "Scheduling of fault-tolerant embedded systems with soft and hard timing constraints," in Proceedings of Design, Automation & Test in Europe DATE '08, 2008, pp. 915–920.
3. S. Furst., "AUTOSAR the Next Generation – The Adaptive Platform," in CARS@EDCC 2015, Paris, France, 8th September 2015.
4. A. Burns and R. Davis, "Mixed criticality systems-a revie," Department of Computer Science, University of York, Tech. Rep, pp. 1–69, Januray 2017. [Online]. http://www-users.cs.york.ac.uk/burns/review.pdf
5. P. Sahu, S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design," ournal of Systems Architecture, vol. 59, no. 1, pp. 60-76, January 2013.
6. S. Manolache, P. Eles, and Z. Peng, "Task mapping and priority assignment for soft real-time applications under deadline miss ratio constraints," ACM Transactions on Embed. Comput. Syst, vol. 7, no. 2, pp. 1-35, 2008.
7. N. Zamora, X. Hu, and R. Marculescu, "System-level performance/power analysis for platform-based design of multimedia applications," ACM Transactions on Design Automation of Electronic Systems, vol. 12, no. 1, 2007.
8. L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in roceedings of 19th IEEE Real-Time Systems Symposium, 1998, pp. 4-13.
9. S. Chakraborty and L. Thiele, "A new task model for streaming applications and its schedulability analysis," in Proceedings of Design, Automation and Test in Europe DATE'05, 20005, pp. 486–491.
10. P. Schleiss, M. Zeller, G. Weiss, and D. Eilers, "Safe Adapt: Safe Adaptive Software for Fully Electric Vehicles," in 3rd Conference on Future Automotive Technology (CoFAT), 2014.

11. M. Hussein, R. Nouacer, A. Radermacher, "A Model-driven Approach for Validating Safe Adaptive Behaviors," in 19th Euromicro Conference on Digital Systems Design (DSD 2016), Limassol, Cyprus, August 31 – September 2, 2016.
12. D.J. Chen, S. Gerard, H. Lonn, M.O. Reiser, D. Servat, C.J. Sjostedt, R.T. Kolagari, M. Torngren, and M. Weber P. Cuenot, "Managing Complexity of Automotive Electronics Using the EAST-ADL," in 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS '07), Washington, DC, USA, 2007, pp. 353-358.
13. B. Morin, O. Barais, J.M. Jezequel, F. Fleurey, and A. Solberg, "Models@ Run. Time to Support Dynamic Adaptation," Computer, vol. 42, pp. 44-51, 2009.
14. O. Scheickl, M. Rudorfer, "Automotive Real-Time Development Using Timing augmented AUTOSAR Specification, BMW Car IT," in 4th European Congress Embedded Real-Time Software (ERTS 2008), Toulouse, France, January 29 - February 1, 2008.
15. M. Andrews, "Probabilistic end-to-end delay bounds for earliest deadline first scheduling," Proceedings IEEE INFOCOM 2000. Conference on Computer Communications," in Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel Aviv, 2000, pp. 603-612.
16. G. Lipari and S. Baruah, "2001. A Hierarchical Extension to the Constant Bandwidth Server Framework," in Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01), 2001.
17. F. Glover and M. Laguna, Tabu Search. Norwell, MA, USA: Kluwer Academic Publishers, 1997.
18. D. Garlan, S. Cheng, A. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," Computer, vol. 37, no. 10, October 2004.
19. Q. Z. Sheng, J. Yu, A. Segev, K. Liao, "Techniques on developing context-aware web services," Int. Journal of Web Information Systems, vol. 6, no. 3, pp. 185-202, 2010.
20. J. Zhang and B. H. C. Cheng, "Model-based development of dynamically adaptive software," in the 28th international conference on Software engineering, Shanghai, China, 2006.
21. T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," J. Netw. Comput. Appl., vol. 28, pp. 1-18, 2005.
22. R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S. Hallsteinsen, J. Lorenzo, A. Mamelli, and U. Scholz, "MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments," Software Engineering for Self-Adaptive Systems, vol. LNCS 5525, pp. 164-182, 2009.
23. W. Heaven, D. Sykes, J. Magee, and J. Kramer, "A Case Study in Goal-Driven Architectural Adaptation," Software Engineering for Self-Adaptive Systems, vol. LNCS 5525, pp. 109-127, 2009.
24. S. S. Andrade and R. J. de Araujo Macedo, "A non-intrusive component-based approach for deploying unanticipated self-management behaviour," in Software Engineering for Adaptive and Self-Managing Systems (SEAMS '09), 2009.
25. B. Morin, O. Barais, G. Nain, and J. Jezequel, "Taming Dynamically Adaptive Systems using models and aspects," in the 31st International Conference on Software Engineering, 2009.
26. L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in Proceedings of 23rd IEEE Real-Time Systems Symposium, 2002, pp. 71-80.
27. A. Oliveira, E. Camponogara, and G. Lima, "Dynamic reconfiguration in reservation-based scheduling: An optimization approach," in Proceedings of 15th IEEE Real-Time and Embedded Technology and Applications Symposium, 2009, pp. 173–182.
28. P. Kumar Saraswat, P. Pop, and J. Madsen, "Task Mapping and Bandwidth Reservation for Mixed Hard/Soft Fault-Tolerant Embedded Systems," in 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '10), 2010.
29. P. Pop, V. Izosimov, P. Eles, and Z. Peng, "Design optimization of time and cost-constrained fault-tolerant embedded systems with check pointing and replication," IEEE Transactions on VLSI Systems, vol. 17, pp. 389-402, 2009.
30. T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing analysis of the Flex Ray communication protocol," Real-Time Systems, vol. 39, no. 1, pp. 205-235, 2008.
31. P. Pop, P. Eles, Z. Peng, and T. Pop, "Analysis and optimization of distributed real-time embedded systems," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 11, no. 3, pp. 593-625, July 2006.
32. D. Tămaş-Selicean and P. Pop, "Design Optimization of Mixed-Criticality Real-Time Embedded Systems," ACM Trans. Embed. Comput. Syst., vol. 14, no. 3, pp. 1-29, April 2015.
33. R. France, and B. Rumpe, "Model-driven Development of Complex Software: A Research Roadmap," in Future of Software Engineering (FOSE 2007), Minneapolis, MN, USA, 2007, pp. 37-54.
34. M. Salehie, and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," ACM Trans. Auton. Adapt. Syst, vol. 4, no. 2, pp. 1-42, 2009.

35. M. Fowler, "UML Distilled: A Brief Guide to the Standard Object Modeling Language (3 ed.).,", Boston, MA, USA, 2003.
36. H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications: Kluwer Academic Publishers, 1997.
37. L. Abeni and G. Buttazzo, "Stochastic analysis of a reservation based system," in Proceedings of 15th International Parallel and Distributed Processing Symposium, 2001, pp. 946–952.
38. L. Abeni, G. Lipari, and J. Lelli, "Constant bandwidth server revisited," ACM SIGBED Review - Special Issue on the 4th Embedded Operating Systems Workshop, vol. 11, no. 4, pp. 19-24, January 2015.
39. SafeAdapt: Safe Adaptive Software for Fully Electric Vehicles. [Online]. http://www. safeadapt.eu/
40. S. Gérard, C. Dumoulin, P. Tessier, and B. Selic, "Papyrus: A UML2 tool for domain-specific language modeling," in International Dagstuhl conference on Model-based engineering of embedded real-time systems (MBEERTS'07), Berlin, Heidelberg, 2007, pp. 361-368.
41. RACE: Robust and Reliable System Architecture for Future eCars. [Online]. http://www.projekt-race.de/
42. N. Wang, D. Schmidt, and C. O'Ryan, "Overview of the CORBA component model," in Component-based software engineering. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001, pp. 557-571.