ASLA: Adaptive System Level in AUTOSAR

Amel Belaggoun, Ansgar Radermacher, Valerie Issarny

ASLA: Adaptive System Level in AUTOSAR

Amel BELAGGOUN
CEA LIST-Paris
LISE Labs,Point courrier 174
Gif-sur-Yvette,91191 France
amel.belaggoun@cea.fr

Ansgar RADERMACHER
CEA LIST-Paris
LISE Labs,Point courrier 174
Gif-sur-Yvette,91191 France
ansgar.radermacher@cea.fr

Valerie ISSARNY
Inria Paris-Rocquencourt
78153 Le Chesnay,France
valerie.issarny@inria.fr

ABSTRACT

This paper presents an initial approach towards making AUTOSAR dynamic starting from the application layer down to the operating system level (task model and RTE)(i.e. extending AUTOSAR ECU Software architecture) and describes ASLA, which is the framework that provides tasks-level adaptation techniques in AUTOSAR.

1. INTRODUCTION AND MOTIVATION

Nowadays, the complexity of the next generation of automotive embedded systems such as Fully Electric Vehicles (FEVs) is increasing due to the growing number of functionality (more than 2.500 functions such as power trains, steering or braking systems, "X-by-wire" systems etc.). These systems are by nature, real-time. Moreover, most of them work under several resource constraints, due to cost, space and energy limitations. In addition, they are running in highly dynamic environments. Combining real-time features in tasks with dynamic behavior, together with cost and resource constraints may create new problems to be addressed in the design of such systems. Using the classical design approaches adopted in hard real-time systems, such as WCET analysis to guarantee timeliness, for example, is no longer acceptable in highly dynamic environments because it would waste resources and increase costs [5]. Instead of allocating resources for the worst case, there is a need for smarter techniques to sense the current state of environment and react accordingly, which means, to cope with dynamic environments, automotive systems need to be *adaptive*; that is, they must be capable of changing their structure and/or their behavior to better reflect their current situation or in order to keep the system requirements at a desired level; if this is not possible, degrade it in a controlled way. Besides dealing with problems generated by adaptation in terms of meeting real-time constraints despite the system evolution; system reliability remains an important factor to be investigated within embedded systems due to their interactions with the physical word. Although, there has been much research on building reliable distributed real-time systems [3][7], a trend towards more complex features in a system with cost and resource constraints poses a major challenges in developing such a system. In order to address the above challenges we need to study the feasibility of applying adaptation techniques in real-time embedded systems. Specifically, in automotive systems it would be very useful to support adaptation in order to increase the availability and reliability of software-based applications without additional hardware costs. Currently AUTOSAR[1]– The standard architecture for automotive systems– has no support for runtime adaptation. Making it adaptive requires specific support at different levels of the software architecture. The most important component affecting adaptivity is the Operating System (OS), but some flexibility can also be introduced in the runtime environment(RTE). Therefore, we propose a layer called ASLA, which is used for task mapping, bandwidth allocation and adaptation for mixed-criticality distributed systems. The novelty of our approach is the capability to support the adaptation of applications with soft and hard real-time requirements (mixed) while respecting timing and safety requirements in AUTOSAR.

Overcoming AUTOSAR's limitations with ASLA: ASLA provides the ability to dynamically adapt the system, such as adding a new application or moving an existing application to a different ECU. In AUTOSAR, the system configuration is by design static: the AUTOSAR RTE is configured at design-time for specific ECUs and partly generated based on the requirements of the Software Components(SWC). A reconfiguration of the system, such as adding an application or moving an application from one ECU to another, cannot be done dynamically at runtime. ASLA extends AUTOSAR in two ways. It changes the scheduling policy from a fixed-priority (assigned to tasks at design time) to a dynamic preemptive policy based on EDF and CBS schedulers to guarantee mixed critical requirements. It also contains RTE extension that supports the runtime application migration between ECUs in response to both anticipated changes caused by the environment, such as network connectivity, as well as unexpected failures in both software and hardware. The deployment and re-configuration of an application onto ASLA is based on the work of [15], which uses a Tabu Search based meta-heuristic to search design space exploration to give the best task allocations and bandwidth allocation. Unlike the existing approaches [10, 18], ASLA explicitly introduces the concept of runtime adaptation in mixed-criticality applications in automotive systems.

Contribution. This paper presents the design of a real-time adaptive distributed architecture, ASLA, to provide task-level adaptation techniques in AUTOSAR.

The remainder of this paper is organized as follows. Section 2 summarizes the background, the scope and requirements needed to understand the proposed approach. Section 3 presents our approach and the architecture of ASLA. Section 4 presents the related work. We state our conclusion and future work in Section 5.

2. BACKGROUND AND DEFINITIONS

Before delving into the approach, it is important to clarify the scope and the requirements that need to be addressed

with our solution. We use Kiviat diagrams [16] to show visually the characteristics of our solution (Fig.1) and the requirements we consider for adaptive automotive systems (Fig.2). These diagrams provide developers of automotive software an easy way of viewing the characteristics of their applications. The dimensions represent axes of the Kiviat diagrams and characteristics of the dimensions represent the set of properties to be met by our solution (i.e. the red bullets).

2.1 The scope of our Research

The adaptation Model. The adaptation may typically be *synchronous* and/or *asynchronous* with respect to the execution of applications. In the *synchronous* case, the applications synchronize their execution, and new tasks (actions) are introduced only after all applications have finished performing the actions specified in the initial configuration. The schedulability analysis of the adaptation is thus not required, because there is no interference between tasks before and after the adaptation (i.e. in the new system configuration). On the other hand, in the case of *asynchronous* adaptation, all the applications start changing their configuration as soon as they receive an adaptation trigger without considering the behavior of the other applications. As a result, actions of the initial system configuration run concurrently with the new ones during the transition, which calls for schedulability analysis.

Promptness. Adaptation is well suited for systems that require reactive behavior, there is no need to wait until an idle period or slack before performing such a change as in Tindell's model [17] or at the end of cycles like in some approaches based on cyclic executive scheduling [14]. The adaptation may be classified in three different categories in the time domain: (i) *time-based adaptation:* These are adaptations where we know the arrival time of the adaptation request in advance. (ii) *event-driven adaptation:* These are adaptations triggered by events rather than time. We don't know exactly when they happen.(iii) *irregular event-driven adaptation*: These are adaptations when no prediction can be made about the arrival time of the adaptation request. An example of this type is a system fault. The system may change its configuration and migrate to a degraded one where not all the functionalities are provided. In our work, we consider all three types of adaptation.

Scheduling policy. The use of dynamic priorities scheme suits better with systems running in highly dynamic environments [14][9].

Scheduling algorithm. Combining two scheduling mechanisms CBS and EDF has proven its efficiency to solve the problem of temporal isolation due to the integration of soft and hard real-time applications on the same platform [9][15].

Architecture. Distributed architecture. In our work a real-time distributed system is defined to be a system with multiple autonomous processing units (ECUs) cooperating together to achieve a common goal. We use the term distributed architecture to refer to loosely coupled architectures where message passing is required (full connectivity is assumed between ECUs i.e. each of the ECUs is connected to each other).

Timing requirement criticality. In our solution we are dealing with mixed-criticality systems, to the best of our knowledge no one has applied runtime adaptation considering both soft and hard real-time constraints in AUTOSAR.

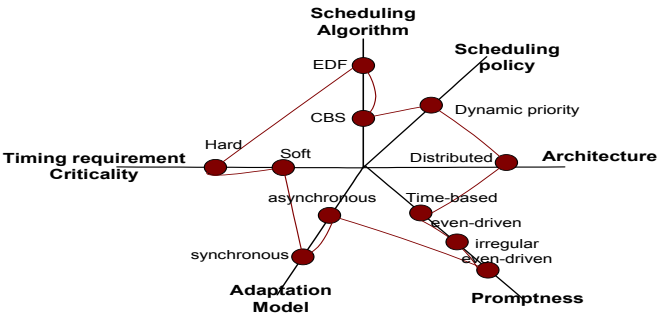

Figure 1: The scope of our Research.

2.2 Definition of our Adaptation Requirements

(R1) Timeliness: the timeliness requirements of an adaptation characterize the time constraints under which the adaptation is executed. Hard real-time constraints require the execution of adaptation within a firm deadline. Adaptations executed under soft real-time constraints minimize the adaptation execution and blackout time (which is the time the application is unavailable due to state transfer and reconfiguration). Unbounded adaptations are executed without any time bound[6].

(R2) Consistency: preserving the system consistency and leaving the system under change in a correct state after adaptation are two major requirements that must be ensured when performing adaptation of the running system. Many adaptation approaches freeze the entities to be reconfigured into an adaptation safe state called quiescent state [8].

(R3) Flexibility: The dynamic behavior of real-time systems requires executing applications with certain flexibility requirements in which the temporal properties and the number of applications vary during runtime. That means, the tasks of flexible application provide implementations that can adapt their execution to the available processing resources. These tasks have variable period and/or may demand variable WCET (i.e. stochastic [9]). So executing flexible applications, prevents the use of an efficient static temporal partitioning of the processing time. A static temporal partitioning that lasts over the entire lifetime of a system would result in an oversized system. Hence, in order to efficiently use the processing time, the flexibility of applications and possible demand changes during runtime have to be considered during runtime analysis.

(R4) Adaptation trigger: adaptation can be triggered either *internally* due to the monitoring infrastructure or *externally*; requested from an outside entity, for example the user. Furthermore , adaptation triggers may arrive synchronously (i.e. at specific time) or asynchronously (i.e. with unknown arrival pattern).

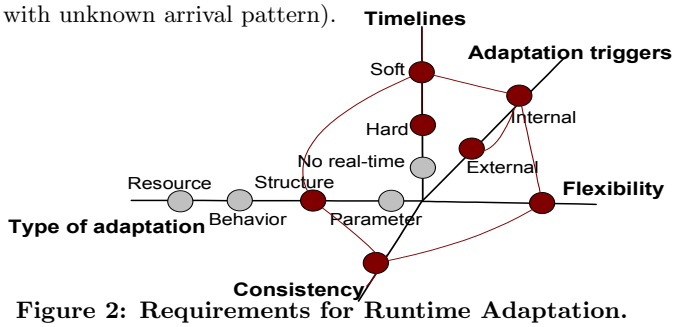

Figure 2: Requirements for Runtime Adaptation.

(R5) Type of adaptation: the type of adaptation defines what is being reconfigured. This type can be either, *Resource adaptation*(i.e.allocation of resources dynamically based on observed conditions), or *Software adaptation* which

includes three categories: (i) structural, (ii) behavioral and (iii) parameter. The first one (i) structural adaptation, changes the actual architectural parts of an application, e.g., by removing a SWC, introducing a new one or replacing /updating an existing SWC with another newer version. The second category (ii) behavioral adaptation allows changes of the behavior of the application and the last category (iii) parameter adaptation involves modifying variable values that determine program behavior.

3. TOWARDS ADAPTIVE AUTOSAR

We present an approach that provides an adaptive solution for AUTOSAR[1]. The approach is based on monitoring various applications distributed on different ECUs to detect the need for adaptation at the application and the system-level (i.e. the Basic SoftWare(BSW)) while maintaining the system consistency [8], which means the system under adaptation must be left in a correct state after adaptation.

3.1 The ASLA Architecture

Fig.3 provides an overview of ASLA's overall design. Every ECU that supports adaptation through ASLA layer consists of a real-time OS with EDF and CBS scheduling policies, an Adaptive SWC that is responsible for reconfiguring applications running on the system, RTE, and application layers. The real-time OS is responsible for HW abstraction, communication, scheduling and executing tasks in real-time. We assume that the underlying HW is a fail-silent system and the communication network is fault-tolerant. Our application layer consists of a set of SWCs (similar to AUTOSAR's) and the new Adaptive SWC which can be distributed over several ECUs. The RTE provides a communication abstraction to SWCs. Unlike AUTOSAR, our RTE extension contains functions to support adaptation. These functions are managed by the Adaptive SWC (more precisely by the Reconfiguration Manager (RM)) which also communicates with the others Adaptive SWCs running on the different ECUs to make one of the adaptation actions such as: adding, deleting or updating application. When a new application is being added, the mapping between the application's SWCs and the ECUs is given to the RM, then each RM analyzes the mapping and renews the RTE's functions.

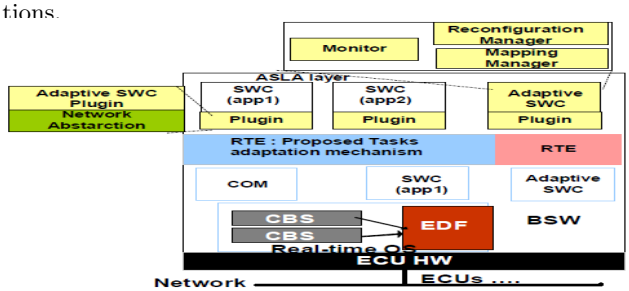

Figure 3: The ASLA architecture.

The ASLA layer is composed of an Adaptive SWC (one on each ECU) and plugin offering a task execution container. This plugin enables any task launched on ASLA layer to be periodically executed. The adaptive component has a coordination-based architecture. One Adaptive SWC acts as a coordinator of the other Adaptive SWCs which are responsible for handling tasks on each ECU and monitoring a health vector. The latter contains all Non-Functional Requirements(NFRs) needed for the adaptation such as the ECU's processor utilization, resources, QoS, HW NFRs..etc. All operational ECUs compute their resources and processor utilization in form of a health vector at a fixed time period and share their health vector with each other. This provides each ECU a consistent view of the available resources and utilization on the other nodes. Since our Adaptive SWC has a coordination based-architecture, we define a management protocol between the different Adaptive SWCs inspired by [11]. However, we differ from them in the sense that our protocol is much more simpler and it is used for managing the process of an adaptation in distributed real-time systems. In our protocol, all the Adaptive SWCs including the coordinator broadcast messages to each other. The coordinator can detect the failure of the others Adaptive SWCs by a lack of heartbeat messages. The major components of ASLA are described below.

A. The Adaptive SWC. As illustrated in Fig 3, an Adaptive SWC is composed of a monitor,a Mapping Manager (MM) and a Reconfiguration Manager(RM). The monitor is responsible for monitoring events that trigger the adaptation. The MM offers a dynamic deployment of tasks on the ECUs and the RM can automatically reconfigure tasks inside/or between the different ECUs :

- **The Monitor.** The monitor periodically sends messages to other ECUs in the system via the network[1]. The monitor allows ASLA to agree on the availability of each ECU. Any adaptation trigger received by the application during its execution may invoke the monitor which sends a message to the RM in order to adapt the application. The loss of a message for two consecutive cycles means that the ECU is no longer alive and the adaptation needs to be triggered to accommodate the desired changes.
- **The Mapping Manager.** The MM offers an automatic deployment of tasks on ECUs. We use TSMBA (Tabu search Mapping and Bandwidth Allocation)[15] as a base line algorithm for our work to do the allocation which provides a comprehensive solution that allocates mixed critical application to a distributed heterogeneous architecture and reserves processor bandwidth for guaranteeing timing requirements. We propose the extension O-TSMBA (Operational chains-TSMBA) a variant of TSMBA that supports task dependencies (i.e. pipeline tasks model). The MM takes as input the application description (an initial system configuration file) and changes the current mapping when it's necessary to do so. Changes of the allocation can occur due to the adaptation or in case of one or several ECUs failures.
- **The Reconfiguration Manager.** The RM is a sporadic task that gets triggered upon the reception of an adaptation trigger (requests for adding new tasks, requests for migrating failed tasks/and or failed ECUs, replacement of tasks with an improved version and removing tasks).

B. ASLA Plugins. All applications will run on the top of ASLA plugins. ASLA plugins support the mechanisms for task reconfiguration and bandwidth allocation (i.e.TSeRBA algorithm "Tabu Search Reconfiguration and Bandwidth Allocation") and also enables tasks to have guaranteed and protected access to required processing resources during reconfiguration in a timely manner.

[1]The Network is beyond the scope of this paper. We assume a synchronous communication network

3.2 ASLA Development Process

Our process for developing automotive software in compliance with AUTOSAR standard is shown in Fig.4. It starts with an application description, in terms of a SWC architecture, dependencies between SWCs, real-time constraints, HW resource requirements and other information needed in the vehicle. In our approach we are interested in mixed critical applications with soft and hard real-time requirements. We consider that each SWC contains one runnable and is represented by one AUTOSAR task. We construct an operational chain OP which is composed of periodically executing runnables generating data and events regularly that flow through multiple runnables (i.e. they are connected by data flow or/and control flow). OP correspond to the AUTOSAR execution model (see Fig4-❶). We distinguish chains with soft and hard timing requirements (OP_{Soft} and OP_{Hard}). We assume that the initial mapping of runnables to AUTOSAR tasks is given at design time similarly to AUTOSAR and all runnables are executed periodically within the context of an AUTOSAR Task (see Fig4-❷). At this step we will have an initial solution which is not necessary schedulable and it is given as input to the task mapping algorithm O-TSMBA which we designed for the operational chain model. The output of O-TSBMA algorithm will be used as input for TSeRBA (i.e. a configuration schedulable and tagged optimized (see Fig4-❸). TSeRBA algorithm is used for task mapping, bandwidth allocation and reconfiguration for mixed-criticality distributed systems. The runtime support to AUTOSAR will be realized by TSeRBA as it allows the dynamic allocation of SWCs with both hard and soft real-time constraints as well as supports the insertion, the deletion and the migration of SWCs at runtime (see Fig4-❹).

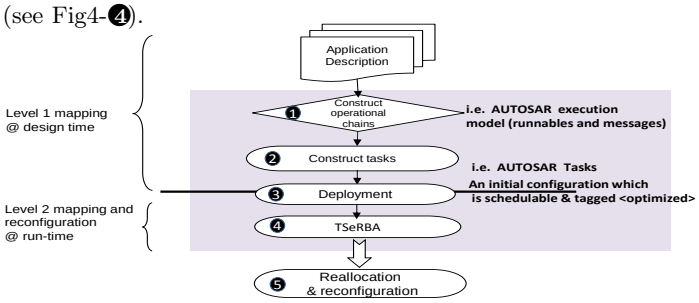

Figure 4: ASLA Development Process

4. RELATED-WORK

Runtime adaptation in real-time systems have been extensively studied in[4, 18, 7]. Unlike existing works, ASLA provides a framework to support runtime adaptation with taking into account schedulability analysis and task allocation for mixed-criticality applications in AUTOSAR. Except SAFER[7] which can be used as a complementary solution for our proposal for the dependability point of view. Another well researched topic in real-time systems–mode change concept, which has a close relationship with the adaptation. In order to guarantee timing requirements in the presence of changes in the system several approaches have been proposed and focused on mode change protocols (see the survey of [14] for more details),[12], which can be potentially used as an extension to ASLA. The authors of [13] proposed SIRAP, a protocol for synchronization in a hierarchical real-time scheduling framework. Their approach is relevant to our case because we are using a server-based technology to schedule AUTOSAR SWCs. However, we have a different focus; we are tackling the challenge of making runtime adaptation in AUTOSAR.

5. CONCLUSION AND FUTURE WORKS

Runtime adaptation in embedded real-time systems is a topic expected to grow in the coming years, gaining particular moment in the context of designing FEVs. Yet there is a lack of techniques and tools for performing such adaptation. We presented ASLA, a novel framework that supports task-level reconfiguration features in AUTOSAR. We have built an experimental platform using three ARM-based STM32F4Discovery boards, an Open-source AUTOSAR implementation, ERIKA-OS[2] served as the BSW. A CAN bus communication was established between the three ECUs and currently we are focusing on the implementation of the algorithms behind ASLA framework to demonstrate the theoretical ideas. As future work, ASLA will be validated by means of a real case study from the SafeAdapt project. We will measure the overhead of the adaptation mechanisms, in particular the impact on timeliness.

Acknowledgments: We would like to thank Robert Davis for the interesting discussions about this work.

6. REFERENCES

[1] Autosar,http://www.autosar.org/.
[2] Erika entreprise,http://erika.tuxfamily.org/drupal/.
[3] J. Balasubramanian, A. Gokhale, A. Dubey, F. Wolf, C. Lu, C. Gill, and D. Schmidt. Middleware for resource-aware deployment and configuration of fault-tolerant real-time systems. In *RTAS*, pages 69–78, Los Alamitos, CA, USA, 2010.
[4] B. e. a. Becker. Model-based extension of autosar for architectural online reconfiguration. MODELS'10, pages 83–97, Berlin, Heidelberg, 2010.
[5] G. Buttazzo and L. Santinelli. Adaptive mechanisms for component-based real-time systems". In *NASA/ESA Conference(ASH 2015), June 15-18, Montreal, QC, Canada.*, pages 1–8, 2015.
[6] S. Fritsch, A. Senart, and D. C. Schmidt. Time-bounded adaptation for automotive system software. ICSE '08, pages 571–580, New York, NY, USA, 2008.
[7] J. Kim, G. Bhatia, R. Rajkumar, and M. Jochim. SAFER: system-level architecture for failure evasion in real-time applications. In *RTSS '12, San Juan, PR, USA, December 4-7, 2012*, pages 227–236, 2012.
[8] J. Kramer and J. Magee. Self-managed systems: An architectural challenge. FOSE '07, pages 259–268, Washington, DC, USA, 2007.
[9] L.Abeni and G. Buttazzo. Stochastic analysis of a reservation based system. In *IPDPS'01, pp.946-952*, 2001.
[10] H. Martorell, J.-C. Fabre, M. Roy, and R. Valentin. Improving adaptiveness of autosar embedded applications. SAC '14, pages 384–390, New York, NY, USA, 2014.
[11] M. Mitzlaff and Kapitza. Enabling mode changes in a distributed automotive system. CARS '10, pages 75–78, New York, NY, USA, 2010.
[12] V. Nelis and B. e. a. Andersson. Global-edf scheduling of multimode real-time systems considering mode independent tasks. ECRTS'11, pages 205–214, Washington, DC, USA, 2011.
[13] T. Nolte, I. Shin, M. Behnam, and M. Sjödin. A synchronization protocol for temporal isolation of software components in vehicular systems. 5(4):375–387, November 2009.
[14] J. Real and A. Crespo. Mode change protocols for real-time systems: A survey and a new proposal. *Real-Time Syst.*, 26(2):161–197, Mar. 2004.
[15] P. K. Saraswat, P. Pop, and J. Madsen. Task mapping and bandwidth reservation for mixed hard/soft fault-tolerant embedded systems. RTAS '10, pages 89–98, 2010.
[16] K. W. K. T. software empiricist. *SIGMETRICS Perform. Eval. Rev.*, 2(2), 1973.
[17] K. Tindell, A. Burns, and A. Wellings. Mode changes in priority preemptively scheduled systems. In *RTSS'92, USA*, pages 100–109, 1992.
[18] C. Zeller, Marc; Prehofer. Timing constraints for runtime adaptation in real-time, networked embedded systems. SEAMS '12, pages 73–82, 2012.