Enhancing AUTOSAR Methodology to a COTS-based Development Process via Mapping to V-Model

Manish Kumar, Jonghun Yoo and Seongsoo Hong
School of Electrical Engineering and Computer Science
Seoul National University
Seoul, Korea
{manish, jhyoo, sshong}@redwood.snu.ac.kr

Abstract— **AUTOSAR, an open standard for automotive software, is currently being exploited by the automotive industry. Although the standard mainly focuses on software architecture, it also provides a development methodology. Unfortunately, the methodology in its current form is insufficient for industrial exploitation because it describes only an incomplete set of activities, work products and their dependencies. Specifically, (1) the activities to support COTS-based development are missing even though AUTOSAR encourages the use of COTS components; (2) it does not describe the roles and their responsibilities; and (3) it does not specify the mapping of activities onto a complete process model. In this paper, we propose a new software development process for AUTOSAR by extending the existing methodology. In doing so, we add activities to allow COTS component selection, evaluation and integration. Then, we define specific roles and assign responsibilities to those roles. Finally, we describe the overall timeline of various activities in detail by mapping the activities to the V-model. In order to present the process, we have used SPEM 2.0 notation, which is backward compatible with the AUTOSAR methodology and has improved expressiveness. We have composed the proposed process model using Eclipse Process Framework Composer which not only performs a sanity check of the model but also provides a way to publish it.**

Keywords-component, SPEM, COTS, AUTOSAR, Process Modeling

I. Introduction

AUTOSAR is an automotive software platform standard established by the European automobile manufactures and their first tier suppliers. The goal of AUTOSAR is to allow automotive software to be composed of independently developed components to improve reusability. Although the standard is currently considered as mature and stable, its adoption has not yet been widespread in industry. One of the main reasons is the lack of a development process suited for AUTOSAR. In fact, the AUTOSAR standard merely provides a partial list of dependencies among artifacts. Its specification does not define when artifacts are generated and who are responsible for what. To apply AUTOSAR in practice, developers need a more concrete development process. They need a wider set of activities which can help them in making better decisions. Furthermore, they need to know who is going to do what in an organization at what time during the product lifecycle.

Thus, we propose a complete software development process suited for AUTOSAR by extending the existing AUTOSAR methodology. First, we incorporate in the methodology those activities that enable COTS-based software development. Although AUTOSAR promotes the use of COTS components, the methodology does not provide the activities needed for COTS selection, evaluation and integration. Second, we define roles for activities and assign them responsibilities. They describe who in an organization will do what. Third, we define an overall timeline for various activities in the form of a process since the methodology does not prescribe a precise order of activities. Our proposed process is divided into two sub-processes. One takes care of system development and the other component development. We map the system development to the V-model [2] since it is widely used in the automotive industry.

To present our process, we model it using SPEM 2.0 notations. SPEM [4], Software Process Engineering Meta-model, is a UML profile, specified by the OMG, for modeling software development processes. AUTOSAR already uses a small subset of SPEM 1.1 notations to describe its methodology.

We have composed the proposed process model using Eclipse Process Framework Composer (EPFC) [3]. Using this tool, we could not only perform a sanity check of the proposed process model but also easily publish it. We could also clearly bring out collaboration between an original equipment manufacturer (OEM) and suppliers. This is not a trivial task since the AUTOSAR approach introduces many challenges such as circular dependencies among an OEM and suppliers and multiple layers of abstraction. Furthermore, we could easily customize the process according to our specific needs.

The rest of the paper is organized as follows. In Section 2, we discuss the shortcomings of the AUTOSAR methodology. In Section 3, we provide our AUTOSAR-specific software development process and explain its steps in detail. Section 4 gives our experience in composing the process with EPFC. Finally, Section 5 serves as our conclusion.

II. PROBLEM DESCRIPTION

The AUTOSAR methodology provides some guidance to work with AUTOSAR but it does not provide a complete process description. It describes only an incomplete set of activities, work products and their dependencies. Specifically, the methodology has the following three shortcomings.

- The methodology fails to address important functional parts of the process. Specifically, it ignores activities and work products required for COTS selection, evaluation and integration. These activities and work products are important because AUTOSAR encourages COTS-based development. We believe that they should be incorporated in the process to ensure that decisions for acquiring COTS software are made at right time by right people [5]. In this paper, we assume that both AUTOSAR Software (ASW) components and Basic Software (BSW) modules are COTS software.
- The methodology fails to show the organizational perspective of the process since roles and responsibilities are not defined. It is not clear where and by whom in an organization, activities are performed.
- Furthermore, the methodology fails to show the behavioral perspective of the process since the overall timeline is missing. It is not clear when activities are performed (e.g., sequencing), as well as how they are performed through feedback loops, iterations, complex decision-making conditions, entry and exit criteria, and so forth [6].

While we understand that these shortcomings are intentional since the standard is not supposed to publish a suitable process, a lot of developers are disappointed by them. In fact, they expect to get more concrete process recommendation from the standard [1]. We attempt to propose a comprehensive process model that overcomes these shortcomings.

III. AUTOSAR DEVELOPMENT PROCESS MODEL

In this section, we describe the steps we have taken to fulfill the shortcomings of the AUTOSAR methodology in detail. We also discuss the rationale and the outcome of these steps.

A. *Adding Activities and Work Products for COTS-based Development*

Fig. 1 shows the conceptual model of our proposed COTS-based development. We assume that OEM is primarily

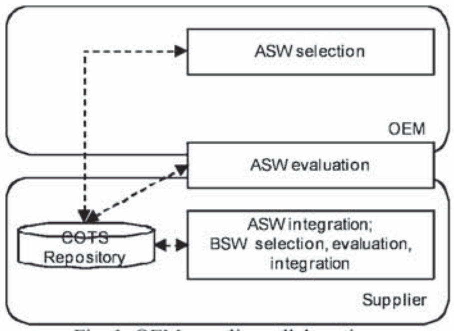

Fig. 1. OEM-supplier collaboration.

responsible for ASW component selection; that OEM and suppliers together evaluate the ASW components; and that suppliers are solely responsible for their integration. For BSW modules, suppliers are entirely responsible for selection, evaluation and integration.

We define activities, work products and their dependencies to enable COTS-based development. The process starts with the *Initial System Requirements* and finally delivers the *Final Acquisition List*, which contains the confirmed list of COTS components for building a system. Please note that these activities are performed for both ASW components and BSW modules at various stages of system development. The detailed description of each activity is given as follows.

1) Analyze requirements: This iterative activity checks if system requirements can be fulfilled by the available COTS components by surveying the COTS component market. The output of this activity is a set of refined system requirements which are mapped to COTS availability.

2) Extract suitable COTS components: This activity mainly involves extracting specific requirements from the refined system requirements and mapping them to specific COTS components which are available in the market.

3) Structural verification: This activity primarily involves verifying interfaces provided or required by the components. The output of this activity is a refined set of components which are structurally suitable for the system.

4) Behavioral verification: This activity is performed to check the behavioral aspects of the selected components. Even if they fit in the system structurally, they may not behave as per system requirements. This activity further refines the selected list of the components. The final list is possible candidates for integration.

5) Evaluate implementation: This activity confirms the COTS components for integration by verifying their implementation aspects. It results in a pre-acquisition list of the COTS components.

6) Verify modification: After the evaluation of their implementation, the components are integrated with the rest of the system. The integration generally involves some customization or optimization, which may affect the behavior of the components. This activity verifies the components again to ensure that they behave the same way as they behaved when tested in isolation. The output of this activity is the final acquisition list of COTS components.

B. *Defining Roles and Assigning Responsibilities*

Table 1 shows the roles we define for both the system and component development. These roles are very common in any software organization but their descriptions which determine their skills, competencies and responsibilities are AUTOSAR specific. For example, the role "architect" in the OEM row describes an actor who is responsible for designing software composition and hardware topology. Thus, the actor must have enough domain knowledge of automotive software systems and various hardware resources available in the market. Similarly, an "architect" in the Supplier/Vendor row

TABLE 1. ROLES AND THEIR DESCRIPTIONS

	Role	Description
OEM	Stakeholder	The management or the end user of the system whose needs must be satisfied by the project
	Analyst	Gathers input from the stakeholders to understand the problem and translate them into technical requirements (system level)
	Architect	Responsible for designing the system, both software composition and hardware topology
	Manager	Responsible for project planning and coordination between stakeholders and suppliers
	Developer	Responsible for integration of ECUs in the system
	Tester	Responsible for black-box testing of the ECUs and the System
Supplier / Vendor	Stakeholder	OEMs for Tier 1 suppliers or Tier 1 suppliers for Tier 2 suppliers
	Analyst	Gathers input from the stakeholders to understand the problem and translate them into technical requirements (ECU or component level)
	Architect	Responsible for designing ECU configuration or component architecture
	Manager	Responsible for project planning and coordination between stakeholders and development team
	Developer	Responsible for integrating components in ECUs or development of new components
	Tester	Responsible for functional testing of new components or of components integrated in an ECU

describes an actor who is responsible for configuring an ECU or its component architecture. This actor is supposed to know the details of the ECU and its execution environment.

The specific job of a role is described by assigning it specific activities and work products. Due to space limitation, we cannot show the responsibility assignments of all the roles; we explain how this is done by giving two examples. The "architect" in an OEM organization must perform four activities: *Decide on all SWC in the System*, *Generate System Topology*, *Generate Top Level Composition* and *Perform SWC-ECU Mapping*. These activities result in five work products: *Collection of Available SWC Implementation*, *System Configuration Description*, *System mapping Constraints*, *System Topology* and *Top-Level Composition*. These activities and work products are in accordance with the role description as mentioned in the last step. Similarly, the "architect" in a supplier organization performs two activities: *Decide on all Atomic SWC Implementations* and *Generate ECU SW Composition*. They result in two specific work products: *All Atomic SWC Implementations on the ECU* and *ECU Software Composition*. These activities and work products again match the role description. Similar assignments are made for all the roles shown in Table 1. These assignments make it clear as who in the organization is doing what activities. This helps ensure that no activity or work product is left out and also to ensure that there is no conflict in responsibilities between two roles.

C. Mapping AUTOSAR Methodology to Phases of Development Process and Defining a Lifecycle

As shown in Fig. 2, we map the extended set of activities to various phases of a development process. The AUTOSAR methodology starts with system configuration, ignoring the activities which are required to generate the inputs of system configuration. We introduce requirement and architecture phases which contain activities to generate the inputs of system configuration. After that, we map the system configuration activities to system design phase. Then, ECU configuration takes place and we map these activities to the subsystem design phase. Once ECU configuration is over, the components are integrated into an ECU and the executable is generated. These activities are mapped to the integration phase. The AUTOSAR methodology does not describe testing related activities. We add the following phases to take care of this issue: unit testing phase which contains activities to test the functionality of ECUs, subsystem testing phase which contains activities to test the ECUs as a black box before integrating them into the system, system testing phase which contains activities to test the system as a whole and finally, vehicle testing phase which contains activities for the user testing of the system.

For component implementation, we introduce the requirement phase which contains activities for the software requirement analysis. Then, the API generation activities of the AUTOSAR methodology are mapped to the structural design phase and activities required to implement the component are mapped to the implementation phase. To test the newly developed components, we add testing phase which contains testing related activities. Finally, the resource requirements of the component are measured in resource measurement phase.

Once the mapping is over, we define an end-to-end lifecycle for the system development and for the software component development. We use the Delivery Process of SPEM 2.0 to present the lifecycle. A Delivery Process is a Process that covers a whole development lifecycle from beginning to end [4]. Delivery Process defines what gets produced, how those items are produced, and the required staffing in the form of integrated Work, Work Product, and Team Breakdown Structures. We use it to describe only the sequencing aspects of the Process ignoring the business aspects such as the scale of the engagement and staffing information.

We define two Delivery Processes: one for the system development and the other for the component development. We map the system development phases to the V-model because of its popularity in the automotive industries whereas we map the component development to the waterfall model for the sake of simplicity. We observe that although component

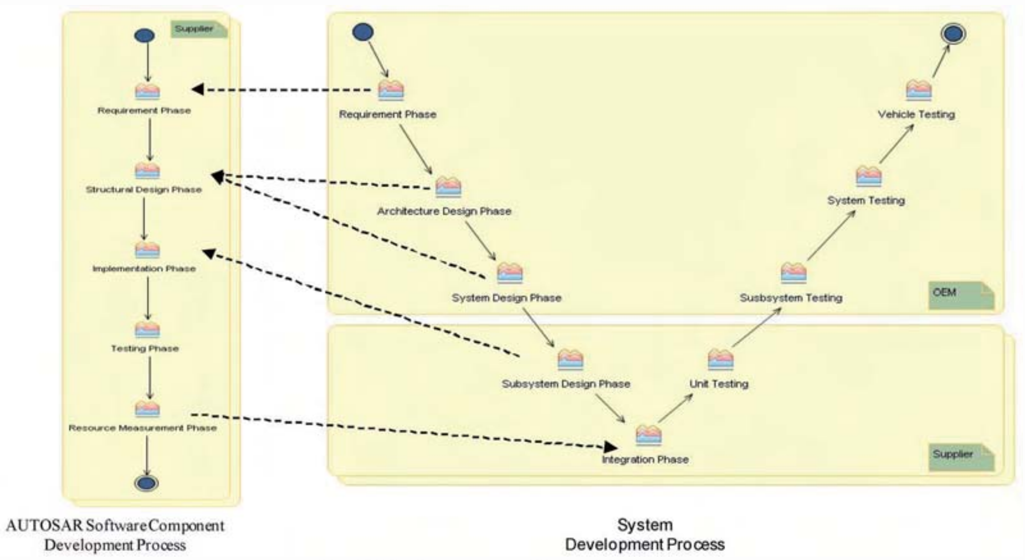

Fig. 2. Final process modeled in SPEM 2.0.

development is supposed to be independent of system development, there exist some dependencies between them as shown by dotted arrows in Fig. 2. For instance, requirements for a needed component are derived from the system development. The structural design of a component is done in close collaboration with architecture and system design phases. The implementation of a component happens by taking feedback from the subsystem design phase. Finally, the tested component is integrated during the integration phase of the system development.

IV. EXPERIENCE OF MANAGING THE PROCESS MODEL WITH ECLIPSE PROCESS FRAMEWORK COMPOSER

To manage the process effectively, we have composed it using the EPFC. It is a process management tool used for tailoring and deploying a development process using SPEM 2.0 notations [3]. We have created a *method content* library for the AUTOSAR process which contains the basic elements such as roles, tasks and work products and their relationships. Then, we created *Processes* in the form of Delivery Processes in the EPFC's process editor as the breakdown of nested activities using the basic elements of *method content*. Two Delivery Processes were created, one for system development and another for ASW component development. They were added to a configuration and then published.

In our published configuration, the left pane contains the basic view of the system development and component development process. Developers can navigate through these processes by expanding various development phases. They can also get detailed information about an activity by clicking the activity. The detailed view displayed on the right pane shows the roles that perform the activity, additional performers if any, the input and output work products of the activity and the process usage of the activity.

With the EPFC, we could perform a sanity check of our process model. By presenting the model in a structured view, we could show the possible collaboration that can exist between an OEM and suppliers in an efficient manner. Most importantly, by publishing the process, we could collaborate with other developers concerned with the process. They further were able to customize the process according to their understanding and provided us with their valuable feedback.

V. CONCLUSION

In this paper, we have proposed a development process for AUTOSAR by extending its existing methodology. First, we have introduced additional activities and work products to support COTS selection, evaluation and integration. Then, we have added the missing elements of a process by defining roles and assigning responsibilities to them. Finally, we have mapped these activities to traditional software development process models. We have used SPEM 2.0 to present the process and the EPFC to compose the process.

We believe that the AUTOSAR way of working can be greatly simplified when presented in a form of a process model. The model can provide guidance and can be even tailored according to specific needs of a project. Since the EPFC provides a structured view of the model, it can also serve as a tool for learning about AUTOSAR which by itself is a complex set of standards.

REFERENCES

[1] Automotive Embedded Systems Handbook, edited by Nicolas Navet and Francoise Simonot-Lion, 2008.

[2] The V-Model, http://www.v-modell.iabg.de.

[3] Eclipse Process Framework project, http://www.eclipse.org/epf.

[4] SPEM. SPEM Software Process Engineering Meta-Model home page, http://ww.omg.org/technology/documnets/formal/spem.htm.

[5] B. Henderson-Sellers, C. Gonzalez-Perez, M.K. Serour and D.G. Firesmith, "Method Engineering and COTS Evaluation," MPEC'05 at ICSE'05, May 21, 2005.

[6] Bill Curtis, Marc I. Kellner and Jim Over, "Process Modeling," Communications of the ACM, September 1992, Vol. 35, No. 9.