Communications-Oriented Development of Component-Based Vehicular Distributed Embedded Systems – a Ph.D. Research Proposal

Saad Mubeen
Mälardalen University and Arcticus Systems AB, Sweden
aad.mubeen@mdh.se

ABSTRACT
The model- and component-based development approach has emerged as an attractive option for the development of vehicular distributed embedded control systems. Within this context, we target the issues related to modeling of legacy communication; extraction of timing models; support for End-to-end Response-time and Delay Analysis (ERDA); and inter-operability of functional and execution models. We propose a novel approach for modeling legacy network communication to support the state-of-the-practice development of these systems. Further, we present a method to extract timing models to support the ERDA of these systems. As part of this analysis, we extend the existing response-time analysis for Controller Area Network (CAN) to support mixed transmission patterns which are implemented by some of its higher-level protocols that are used in the industry. We also take a step towards broadening the scope and usability of our techniques by developing methods to support inter-operability of research-oriented functional models and the component and execution models that implement our techniques. In order to show the applicability of our techniques, we provide a proof of concept by extending the Rubus Component Model which is used for the development of software for vehicular embedded systems by several international companies. We also implement the ERDA along with the extended analysis for CAN in the existing industrial tool suite the Rubus-ICE. Moreover, we conduct case studies to validate our methods and techniques.

Categories and Subject Descriptors
D.2.8 [Software Engineering]: Metrics—Design tools and techniques

Keywords
Distributed embedded systems, component model, real-time systems, component-based development.

1. INTRODUCTION
An embedded system is a microprocessor-based system that is designed to perform a dedicated functionality by means of hardware and software [8]. These systems are found in almost all electronic items around us. Often, an embedded system needs to interact with its environment in a timely manner, i.e., it is a real-time system. For such a system, the desired and correct output is one which is logically correct as well as delivered within a specified time (e.g., a deadline). In a distributed real-time embedded system, the functionality is distributed over many nodes (processors) that are connected to one or more networks. The software in these systems (embedded software) has drastically increased in size and complexity in the past few years. In the vehicular domain, e.g., a modern premium car contains 1 GB of software, nearly 100 million lines of code, more than 2000 software functions that are distributed over 70 to 100 Electronic Control Units (ECUs) that may be connected by more than five different buses (or networks) [9].

In order to handle the complexity of embedded software, lower development cost, allow reusability, the research community proposed model- and component-based development of these systems by employing the principles of Model-Based software Engineering (MBE) and Component-Based Software Engineering (CBSE) [19, 10]. MBE provides the means to use models throughout the process of system development. It uses models to describe functions, structures and other design artifacts. Whereas, CBSE facilitates the development of large software systems by supporting reuse and integration of software components and their architectures.

The safety-critical nature of many of these systems requires evidence that the actions by them will be taken at a time that is appropriate to their environment. For this purpose, several a priori analysis techniques including holistic or end-to-end Worst Case Response Time (WCRT) analysis have been developed by the research community [42, 39]. It is a schedulability analysis technique which calculates upper bounds on the response times of event chains that are distributed over more than one node in the system. The end-to-end timing model of the system should be available to perform the analysis. Ideally, a component technology for the development of these systems should support automatic extraction of such timing model.

1.1 Problem Statement and Challenges
The model- and component-based development of software architecture for real-time embedded systems in modern vehicles has had a surge in the last few years. The majority of these existing approaches allow for structural and functional modeling. They do not support execution modeling. The structural modeling is concerned with the structure definition of requirements and high-level architectural objects. The functional modeling refers to the structured way of representing software functions for the system to be modeled. Whereas, the execution modeling is concerned with the modeling of run-time properties and/or requirements (e.g., end-to-end deadlines, jitter, etc.) of software functions. The modeling of the systems should extend down to the execu-
tion level to allow precise control of resource utilization and that timing requirements are not violated when the system is executed. However, providing such a support for the systems is very challenging because their functionality can be realized with more than one execution model, e.g., separate execution models for nodes and networks.

One way to deal with these challenges is to use a component technology that allows model- and component-based development of the systems with the support for modeling, analyzing, predicting and modifying the execution behavior. However, building such a component technology raises many challenges. One of the main reasons behind these challenges is that the development process for these systems in academia and industry may be very different from each other. In academia, the development process often starts with discussions about models and functions. The models are assumed to be platform independent. Further, it is assumed that the models and functions will be deployed on specific platforms at a later stage. However, this way of development is often not practiced in the industry, especially in the vehicular domain. The traditional process for the development of these systems in the industry starts with designing the bus (or network) communication. The infrastructure and platform (e.g., machine, types of ECUs and buses) for the system to be developed is already known. In the early stage of industrial development process, usually the focus is on finding the answers to the questions as follows: How many buses will be there in the system? Which nodes will be connected to which bus? How many messages will be there in the system? Which messages will be sent by each node? After finding answers to these questions, the focus is shifted towards the development of functions. Thus, communication-oriented development process is used for the systems and constitutes the state of the practice.

Within this context, we target those challenges that are concerned with the modeling of real-time network communication and support for end-to-end timing analysis. One such challenge is to support the modeling of legacy network communication and allow the use of legacy nodes in component-based distributed real-time embedded systems. In order to ensure that the system will behave in a timely manner during its execution, we need to analyze tasks, messages and event chains in distributed transactions and predict the end-to-end delays. The component technology for the industrial development of the systems should support state-of-the-art real-time analysis such as end-to-end response-time and delay analysis. The supported timing analysis should be able to incorporate the analysis of common message transmission patterns that are implemented by the real-time network protocols used in the industry. In order to perform the analysis, the end-to-end timing model of the system should be available. The extraction of the end-to-end timing model from the system is another challenge, which we target.

While the introduction of models into the development of the systems increases efficiency in some parts of the engineering process, the models are also cause of novel concerns. In particular, mismatch between structural and semantic assumptions in modeling languages used in different parts of the design process of the systems cause large problems when design artifacts are transformed between modeling languages. In the industry, productivity is hampered by incompatible tools and file-formats, in conjunction with the need for non-trivial, manual and tedious translations between different model-formats. Moreover, these translations are done in ad hoc fashion making the result of the translation unpredictable and potentially with altered semantics. There is a strong need to investigate how to effectively and efficiently work with existing modeling languages for real-time embedded systems. A solution must entail possibilities to make tools inter-operable to allow automated (and semi-automated) translations between modeling languages and tools with preserved semantics. Another challenge that we target in this thesis proposal is to bridge the semantic gap between functional models like EAST-ADL[3] and execution models like the Rubus Component Model (RCM) [17]. The aim is to build a seamless tool chain for communications-oriented modeling and development of software for vehicular embedded real-time systems.

The research problem addressed in this thesis proposal can be refined and formulated into three research challenges.

1. How to support the modeling of legacy network communication during the model- and component-based development of distributed embedded systems?
2. How to extract end-to-end timing models from component-based distributed embedded systems using the communications-oriented development process?
3. In order to build a seamless tool chain to support the communications-oriented development of embedded software in the segment of construction-equipment shown in Figure 1) to encapsulate and abstract the communication protocols in the systems, we allow the use of legacy nodes and legacy protocols in a component- and model-based software engineering environment. With the addition of these components, RCM is able to not only model real-time network communication, but also support state-of-the-practice development of the systems. These extensions also enable adaptation of a node when communication rules change (e.g., due to re-deployment in a new system or due to upgrades in the communication system) without affecting its internal component design. The special-purpose components can be automatically generated from the information about legacy communication or from early design decisions about network communication. The proposed extensions may be applicable to several other component technologies for the development of the systems that use pipes-and-filter style for component interconnection such as ProCom [38].

2.2 Contribution 2: Extraction of End-to-end Timing Models

This contribution addresses second research challenge. In order to perform the End-to-end Response-time and Delay Analysis (ERDA), the end-to-end timing models should be extracted from the systems. The extraction of these models can be challenging because the design and analysis models are usually built using different meta-models. We present a method in [31] to extract the end-to-end timing models from the component-based systems. This method is built upon the modeling approach that we discussed in the first contribution. We discuss and solve the issues concerning the model extraction such as extraction of unambiguous timing

1.2 Proposal Outline

The rest of the thesis proposal is organized as follows. Section 2 discusses the research contributions. Section 3 discusses the related work. Section 4 presents the research plan and current work. Section 5 summarizes the proposal.

2. TECHNICAL CONTRIBUTIONS

The contributions in this thesis proposal are organized in five parts.

2.1 Contribution 1: Modeling of Legacy Network Communication

This contribution addresses first research challenge. We introduce a new approach in [31, 33] for modeling real-time network and legacy communication in the component-based distributed embedded systems. In order to show usability of our modeling approach, we implement it by extending the existing industrial component model, i.e., RCM. By introducing special-purpose components (OSWC, ISWC and NS shown in Figure 1) to encapsulate and abstract the communication protocols in the systems, we allow the use of legacy nodes and legacy protocols in a component- and model-based software engineering environment. With the addition of these components, RCM is able to not only model real-time network communication, but also support state-of-the-practice development of the systems. These extensions also enable adaptation of a node when communication rules change (e.g., due to re-deployment in a new system or due to upgrades in the communication system) without affecting its internal component design. The special-purpose components can be automatically generated from the information about legacy communication or from early design decisions about network communication. The proposed extensions may be applicable to several other component technologies for the development of the systems that use pipes-and-filter style for component interconnection such as ProCom [38].
and linking information from all nodes and networks in the system and linking of event chains in distributed transactions. For example, the linking information is captured in the NS object shown by the curved arrows in Figure 1. The model extraction method and the solutions of encountered problems may be applied to several component technologies that use a pipe-and-filter style for component interconnectivity. The end-to-end timing model that we considered is also general as it incorporates the analysis of several real-time network protocols used in the vehicular domain. To show the applicability of our approach, we demonstrate the implementation of the model extraction method in the analysis framework of the existing industrial tool the Rubus-ICE.

In [32], we further extend the analysis for mixed messages in [26, 25] that considers hardware and software limitations in CAN controllers such as abortable and non-abortable transmission buffers. In [32], we further extend the analysis for mixed messages in the CAN network where some nodes use priority queues while others use FIFO queues. Moreover, in [27, 28], we extend the analysis for mixed messages that are scheduled with offsets and have arbitrary jitter and deadlines. All these extended analyses are applicable to any higher-level protocol for CAN that uses periodic, sporadic and mixed transmission of messages. Figure 2 graphically depicts the relationship between the existing and extended analysis for CAN in this contribution.

2.4 Contribution 4: Proof-of-Concept Implementation

In this contribution [30], we validate our solutions to the first two research challenges. While validating our solutions, we found out that the process of implementing and integrating state-of-the-art timing analysis with existing industrial tool suite offers many challenges. The Implemener has to not only code and implement the analysis in the tool suite, but also deal with several other issues. We discuss the implementation of the ERDA as two individual plug-ins for Rubus-ICE. As part of the ERDA, we implement the existing as well as the extended analysis for CAN. The implemented analysis is general as it supports the integration of WCRT analysis of various networks without a need for changing the end-to-end algorithm. We also implement the extended analysis in a freely-available tool namely MPS-CAN³ analyzer [29]. We describe and solve the problems encountered and highlight the experiences gained during the process of implementation, integration and evaluation of the analysis plug-ins for Rubus-ICE. We believe that most of the experiences gained and solutions to the issues encountered in this work may be applicable when other complex real-time analysis techniques are integrated in any industrial tool suite that supports a plug-in framework (for the integration of new tools) and component-based development of the systems. Finally, we provide a proof of concept for all modeling approaches and extended analyses discussed in the first three contributions by modeling an automotive application using extended RCM and analyzing it with the implemented analysis in Rubus-ICE.

2.5 Contribution 5: Inter-operating Functional and Execution Models

In this contribution, our aim is to take a step towards broadening the scope and usability of our techniques by developing a method to extract interoperable end-to-end timing models at various abstraction levels. At higher levels, the method extracts end-to-end timing information from models of the systems that are developed with EAST-ADL language using TIMMO methodology [6] and are annotated with timing information using Timing Augmented Description Language (TADL2) [5]. At the lower level, the method exploits RCM and Rubus-ICE to extract the timing information that cannot be clearly specified at higher levels such as trigger paths in distributed chains. We also provide a proof of concept for the method by conducting a case study.

³https://github.com/saadmubeen/MPS-CAN.

---

Figure 1: Example of a two-node distributed real-time application modeled with the new approach

Figure 2: Graphical representation of the Response Time Analysis (RTA) and its extensions as part of the contribution
2.6 Discussion
We selected Rubus (RCM and Rubus-ICE) to provide the proof-of-concept implementation for our new modeling techniques and extended analysis for several reasons such as its existing support for structural, functional and execution modeling; capability for developing predictable and analyzable control functions with a support for modeling real-time properties and requirements; separation of interconnections between the functions in terms of data flow and control flow; and automatic generation of run-time framework.

With the proposed extensions, RCM along with Rubus-ICE can be considered a suitable choice for the industrial development of the systems for many reasons. For example, it complements the structural and functional modeling with the execution modeling; supports communications-oriented development process; enables the modeling of legacy communication and legacy systems; models and specifies the timing related information easily; has a small run-time footprint; and supports communications-oriented development and analysis tools.

2.7 Impact of the Contributions
The new approaches for modeling legacy network communication and extraction of end-to-end timing models may be suitable for other component models that use a pipe-and-filter style for component interconnection. The extended analysis supports common message transmission patterns that are software protocols used in the industry today. Further, the analysis engine supports integration of the analysis of various real-time networks without a need for changing the holistic algorithm. Most of the encountered issues, proposed solutions and gained experiences in this work may provide guidance for the implementation of other complex real-time analysis in any industrial tool suite that supports a plug-in framework and component-based development of the systems. The new release of RCM and Rubus-ICE (version 4.0), which is already in the industrial use, incorporates the contributions and results presented in this thesis proposal. Moreover, MPS-CAN analyzer includes the extended analysis for CAN.

3. RELATED WORKS
First, we discuss the works related to the model and component-based development approaches that are suitable for resource-constrained embedded systems and are targeted towards the vehicular domain. In the second part, we discuss the works related to the Response Time Analysis (RTA) for CAN network and ERDA.

3.1 AUTOSAR
AUTOSAR [1] is an industrial initiative to provide standardized software architecture for the development of automotive embedded software. It defines the application software in terms of Software Components (SWCs). The earlier versions of AUTOSAR did not focus on specifying and handling timing information such as real-time requirements and properties. Whereas, these requirements were the key focus throughout the development of RCM. Software development in AUTOSAR is described at a higher abstraction level compared to that of RCM. An SWC in RCM more resembles to a runnable entity (scheduleable element) compared to AUTOSAR SWC. Unlike AUTOSAR, RCM separates control flow from the data flow among SWCs within a node. At modeling level, RCM distinguishes between intra- and inter-node communication which is unlike AUTOSAR modeling. AUTOSAR hides modeling of the execution environment. On the other hand, RCM explicitly allows it at an abstraction level that is close to the functional specification while abstracting the implementation details. Despite these differences, there are some similarities between AUTOSAR and RCM, e.g., the sender receiver communication in AUTOSAR is similar to the pipe-and-filter communication among SWCs in RCM. In conclusion, AUTOSAR is more focussed on the functional and structural abstractions, hiding the implementation details about execution environment and communication. Whereas, RCM is all about modeling, analysis and synthesis of the execution environment of software functions.

3.2 TIMMO, TADL, MARTE and EAST-ADL
TIMMO [6], a large EU research project, is an initiative to provide AUTOSAR with a timing model. Its outcome consists of a methodology; a language TADL [5] to express timing requirements and constraints; and several validators. TADL is inspired by MARTE [7] which is a UML profile for model-driven development of real-time and embedded systems. TIMMO methodology makes use of EAST-ADL for structural modeling and AUTOSAR for implementation. TIMMO-2-USE project [4], a follow-up on the TIMMO, includes a major redefinition of TADL and provides AUTOSAR extensions for timing model. These projects are initiatives to annotate AUTOSAR with a timing model. This will be hard to accomplish all the way since AUTOSAR aims at hiding implementation details of execution environment and communication. At the modeling level, there is no information in AUTOSAR to express low-level details such as linking information in distributed chains. These details are necessary to extract the timing model from the software architecture. In this initiative, there is no focus on extracting this information from the model or perform timing analysis or synthesize the run-time framework. In our view, timing model means extracting enough information to be able to perform certain type of timing analysis, e.g., ERDA.

3.3 ProCom
ProCom [38] is a two-layered component model for the development of distributed embedded systems. It is inspired by RCM and there are a number of similarities between the two, e.g., both have passive components; both differentiate between control flow and data flow; and both use pipe-and-filter style of communication among SWCs. However, ProCom does not differentiate between intra- and inter-node communication which is unlike RCM. It hides communication details, whereas RCM lifts them up to the modeling level. It will be very hard in ProCom to extract the timing model and perform the timing analysis at the level where it is done in RCM.

3.4 Middleware-based Approaches
There are middleware approaches, e.g., RT CORBA, minimum CORBA and CORBA lightweight services for the development of the systems [2]. The run-time framework of RT CORBA is heavyweight. On the other hand, RCM has a small run-time footprint. RCM does the timing analysis and actually synthesize the application as run-time and communication platform efficient as possible. Due to high resource requirements at run-time, RT CORBA is not efficiently usable in the type of applications that RCM focuses on. There are other middleware solutions such as iLand project [22] in which a middleware-based framework is introduced to support predictable and time-bounded reconfiguration at run-time for the service-oriented systems. Whereas, our approach is based on component-based development. In fact, a component-based modeling approach is introduced that enables dynamic replacement of components at run-time while preserving the temporal properties. Whereas, we do not consider dynamic reconfiguration and replacement of components in our work. Most of these techniques have been validated for soft real-time systems in the multimedia-applications domain. However, our approach mostly focuses on hard real-time systems in the vehicular domain.

3.5 RTA for CAN and ERDA
Tindell et al. [43] developed the schedulability analysis
of CAN which was later refuted, revisited and revised by Davis et al. [11]. This analysis assumes that all CAN device drivers implement priority queues [16]. We extended the analysis of CAN with FIFO and work-conserving queues while supporting arbitrary deadlines of messages. The analysis in [43, 11] assumes that the CAN controllers have very large transmit buffers. However, most CAN controllers have small number of transmit buffers [21, 12]. If all these buffers are occupied by lower priority messages, higher priority message released in the same controller may suffer from priority inversion [43, 20, 34, 23]. RTA in [43, 11] was extended in [20, 21] to support the analysis of CAN that contains abortable and non-abortable transmit buffers in the controllers respectively. All these analyses assume the messages are queued for transmission periodically or sporadically. Mubeen et al. [24] extended the existing RTA [43, 11] to support mixed messages in CAN where nodes implement priority queues. Mubeen et al. [32] further extended their analysis to support mixed messages in the network where some nodes implement priority queues while others implement FIFO queues. RTA for mixed messages in CAN [24] was extended to support the analysis of CAN that contain abortable and non-abortable transmit buffers in the controllers in [26] and [25] respectively.

But, none of the analysis discussed above supports messages that are scheduled with offsets. The WCRTs of lower priority messages in CAN can be reduced if the messages are scheduled with offsets [41, 16]. A method for the extraction of offsets to improve the overall bandwidth utilization is proposed in [16]. The WCRT analysis with offsets has been developed by several researchers [44, 14, 41, 18, 45]. None of the above analyses with offsets supports mixed messages. Offset-based analysis [44] was extended in [27, 28] to support WCRT calculations for mixed messages in CAN.

Stappert et al. [40] formally described end-to-end timing constrains for automotive multi-rate systems. A framework for the calculation of end-to-end delays for the systems is presented in [15]. A scalable technique, based on model checking, for the computation of end-to-end latencies is described in [35]. In [30], we implemented the end-to-end delay analysis [15] as a plug-in for the Rubus-ICE tool suite. We implemented the holistic analysis of [42] in [30] because it provides the flexibility to use several network-communication protocols used in the automotive domain.

4. RESEARCH PLAN AND PROGRESS

First four contributions presented in this thesis proposal have been completed. The research work concerning the fifth contribution is ongoing. Since different modeling languages support different types of expressions, it is often impossible to define a one-to-one mapping between different constructs in different languages, as discussed in Section 2.5. However, if designers choose to use a certain subset of the full expressiveness of a language, or choose to use a certain style of expression in the language, it can be possible to define unambiguous mappings from these subsets or styles to other languages. Such subsets or styles can be expressed by a set of patterns. Our goal in the fifth contribution is to identify such patterns to allow expression of common solutions in a transformable style. Given above identified patterns that can be translated with preserved semantics, we are investigating which of these can be automatically translated and derive the corresponding translations. We are also working on identifying when a construct cannot be translated with preserved semantics. This functionality is useful both to find errors in the model and to prevent erroneous translations with (potentially subtle) modifications to model semantics. We plan to extract a large enough set of patterns from existing solutions at the partner companies and existing literature. Development of automated translations and also automatic detection of design-patterns that do not allow unambiguous transformation will be the second step. Based on these automated translations, it will be possible to assemble a seamless tool chain for the development of software for vehicular distributed embedded systems. A relationship among the research challenges, contributions and publications included in this thesis proposal is depicted in Figure 3.

5. CONCLUSION

In this thesis proposal, we proposed a novel approach for modeling legacy network communication in component-based vehicular distributed embedded systems. By introducing special-purpose components to encapsulate and abstract the communication protocols, we allow the use of legacy nodes and legacy protocols in a component- and model-based software engineering environment. We also presented a method to extract end-to-end timing to facilitate the End-to-end Response-time and Delay Analysis (ERDA) of these systems. Further, we identified that the existing analysis for CAN does not support common message transmission patterns which are implemented by some higher-level protocols based on CAN that are used in the vehicular industry. We extended the existing analysis to facilitate the worst-case response-time calculations for these transmission patterns. The extended analysis is generally applicable to any higher-level protocol for CAN that uses periodic, sporadic and mixed transmission of messages. In order to show the applicability of our modeling techniques, we provided a proof of concept by extending the Rubus Component Model which is used for the development of embedded software by several international companies. We also implemented the ERDA along with the extended analysis for CAN in the existing industrial tool suite the Rubus-ICE. Moreover, we conducted automotive-application case studies to validate our methods and techniques. We also took a step towards broadening the scope and usability of our techniques by developing methods to support inter-operability of research-oriented functional models and the component and execution models that implement our techniques. The goal is to build a seamless tool chain for communications-oriented development of embedded software in the segment of construction-equipment vehicles. We plan to validate our models inter-operability approach by conducting an industrial case study.

Acknowledgement

This work is supported by the Swedish Research Council (VR) and Swedish Knowledge Foundation (KKS) within the projects SynthSoft and FEMMVA. We thank our industrial partners Arcticus Systems AB and Volvo CE, Sweden.

6. REFERENCES
