COMET: A Component-Based Real-Time Database for Automotive Systems *

Dag Nyström†, Aleksandra Tešanović*, Mikael Nolin†, Christer Norström†, and Jörgen Hansson*

†Mälardalen University
Mälardalen Real-Time Research Centre
Västerås, Sweden
{dag.nystrom, mikael.nolin,
christer.norstrom}@mdh.se

*Linköping University
Dept. of Computer Science
Linköping, Sweden
{alete.jorha}@ida.liu.se

Abstract

With the increase of complexity in automotive control systems, the amount of data that needs to be managed is also increasing. Using a real-time database management system (RTDBMS) as a tightly integrated part of the software development process can give significant benefits with respect to data management. However, the variability of data management requirements in different systems, and the heterogeneousness of the nodes within a system may require a distinct database configuration for each node. In this paper we introduce a software engineering approach for generating RTDBMS configurations suitable for resource-constrained automotive control systems, denoted the COMET development suit. Using software engineering tools to assist developers with design and analysis of the system under development, different database configurations can be generated from pre-fabricated components. Each generated COMET database contains only functionality required by the node it is executing on.

1. Introduction

In recent years, automotive control systems have evolved from simple single processor systems to complex distributed systems. At the same time, the amount of data that needs to be managed by these systems is increasing dramatically; data volume managed by automotive systems is predicted to increase 7-10% per year [5]. Current techniques for storing and manipulating data objects in automotive systems are ad hoc in the sense that they normally manipulate data objects as internal data structures. This lack of a structured approach to data management results in a costly development process with respect to design, implementation, and verification of the system [17]. It also makes the system difficult to maintain and develop while preserving consistency with the environment, e.g., maintaining temporal properties of data. As data complexity is growing the need for a uniform, efficient, and persistent way to store data is becoming increasingly important. Using a real-time database management system (RTDBMS) as a tightly integrated part of an automotive control system has the potential to solve many of the problems that application designers have to consider with respect to data management, e.g., locking of the data, persistency and deadlock situations. More importantly, incorporating an RTDBMS into an automotive control system can reduce development costs, result in higher quality of the design of the systems, and consequently yield higher reliability [9].

The variability of data management requirements in different automotive control systems requires distinct RTDBMS configurations specially suited for the particular system [27]. Since an automotive control system is heterogeneous, consisting of several nodes (called electronic control units, ECUs), see figure 1, the ability to configure the RTDBMS to suit the requirements of an individual node is crucial. For instance, an automotive system could consist of a small number of resource adequate ECUs responsible for the overall performance of the vehicle, e.g., 32bit CPUs with a few Mb of RAM, and a large number of ECUs responsible for controlling specific subsystems in the vehicle, which are significantly resource-constrained, e.g., an 8bit micro-controller and a few kb of RAM [17]. ECUs with greater amount of resources usually have real-time operating systems support, which is not affordable in small resource-constrained ECUs. Although different in their characteristics and available resources, all nodes in an automotive control system are exchanging, sharing and manipulating data, thereby requiring a uniform way of data

*This work is supported by SSF within the SAVE project, SAfety critical components for VEhicular systems.

management, e.g., via a RTDBMS.

The heterogeneous characteristics of nodes in an automotive control system result in a need to have distinct RTDBMS configurations suited for a particular node [17]. In safety-critical nodes, tasks are often non-preemptive and scheduled off-line, implying that a RTDBMS configuration for that node could be made small in size and provided functionality, since the majority of the RTDBMS's functionality, such as synchronization and concurrency-control, could be handled off-line. Less critical and larger nodes have preemptable tasks, requiring a RTDBMS configuration with run-time concurrency control mechanisms, and support for database queries formulated during run-time (ad-hock queries). A configurable RTDBMS supporting different types of nodes would, from the application's point of view, provide uniform access to the data regardless of the size and characteristics of an ECU.

Today, there exists a number of commercial databases suitable for embedded systems, e.g., Pervasive.SQL [19], Polyhedra [20], Berkeley DB [22], and TimesTen [32]. Although small in size and therefore suitable for resource-constrained automotive control systems, these databases do not incorporate real-time behavior. This in turn implies that their behavior cannot be analyzed, which makes them unsuitable for deployment in an automotive system. Research projects that are building real-time database platforms, such as DeeDS [2], RODAIN [12], STRIP [1], and BeeHIVE [24], mainly address real-time requirements, are monolithic, and targeted towards a larger-scale real-time application, which makes them unsuitable for use in embedded resource-constrained environments.

In this paper we propose a software engineering approach for generating RTDBMS configurations suitable for resource-constrained automotive control systems. This approach is supported by the **COMET development suit**. The suit consists of a set of data management, analysis and configuration tools, as well as a library of pre-defined software artifacts providing specific RTDBMS functionality. The library of artifacts and the possible configurations of the RTDBMS are referred to as the COMET RTDBMS platform. With the COMET development suit we aim at providing software developers an automated way of tailoring and analyzing the data management for a particular automotive control system, or a node in the system. The COMET RTDBMS platform, a part of the COMET development suit, is developed using an approach to aspectual component-based software development (ACCORD) [30]. ACCORD enables us to utilize the benefits of component-based software development (CBSD) [25] by developing components that encapsulate specific real-time database functionalities. ACCORD also enables us to exploit the benefits of aspect-oriented software development (AOSD) [11] by providing a way of encapsulating, managing, and implementing cross-cutting concerns in a RTDBMS in a predictable manner; crosscutting concerns include concurrency control, logging, and recovery. In AOSD, a crosscutting concern is a functionality or non-functional feature that cannot cleanly be encapsulated in a procedure, function, object or a class [11].

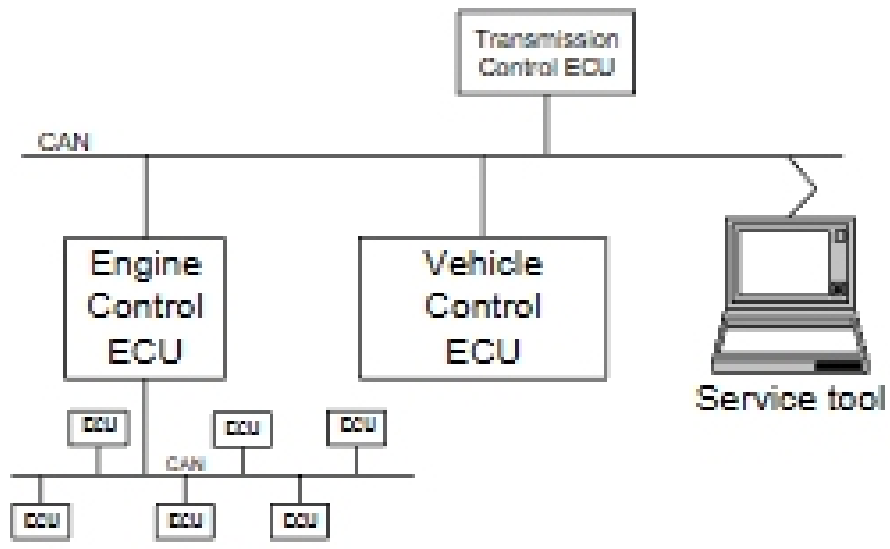

Figure 1. An heterogeneous automotive control system

The paper is organized as follows. In section 2 we present the COMET development suit. We present the key concepts used in the COMET development suit in section 3, including COMET aspects and components and possible COMET RTDBMS configurations. We conclude the paper and discuss our future work in section 4.

2. The COMET development suit

To successfully and efficiently generate systems from a library of pre-defined artifacts, the development process should be supported by appropriate tools. In this section we present our view of the overall development process to obtain system-specific RTDBMS configurations. Figure 2 shows the constituents of this process.

As shown in figure 2, the development of a RTDBMS configuration starts with specifying the requirements of an automotive control system, which are then used as input for making a model of the system. This model consists of the nodes, their interconnections and the individual run-time properties, e.g., the scheduling policy of the node, if the tasks are preemptive or not, available memory, and CPU resources. The goal of making the model of the underlying system is to derive required database functionality for each of the node in the system. Examples of functionality are support for ad-hoc queries (queries dynamically created during run-time), and to enable the data organization to be changed during run-time (i.e., provide a dynamic database schema). Next, a model of the database, i.e., the actual data, and any precompiled queries are derived with the help of

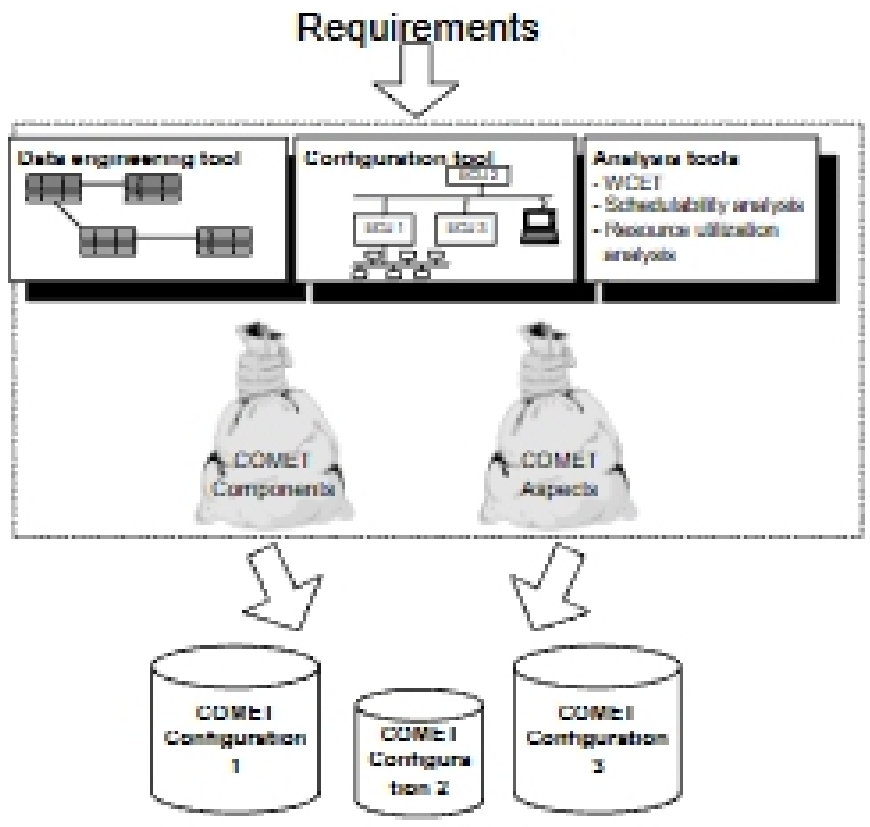

Figure 2. The COMET development suit

the data engineering tool. This step also involves specifying which parts of the database should be available on which node, and the temporal properties of the data, such as temporal consistency [21]. This information, i.e., desired database functionality for each node, data model, and database schema, is then used by the configuration tool to select a set of aspects and components from the library to form a database configuration suitable for each of the nodes (see figure 2). The overall decomposition of the database functionality into aspects and components, and the development of components and aspects, is done according to the ACCORD design principle (see section 3.1). The obtained COMET configurations can then be analyzed with respect to run-time properties, e.g., worst case execution time, memory requirements, and response time analysis, by the analysis tools. If the analysis indicates that the configuration is unfeasible, the configuration step and analysis step could be further iterated until an acceptable solution is found.

The resulting RTDBMSs are configured to contain no more than the needed functionality, thus reducing both computational costs and memory requirements.

3. The COMET key concepts

As mentioned, different nodes in the automotive control system may require distinct RTDBMS configurations. Component-based databases [4, 7, 8, 10, 13, 18, 22] using the component-based software development paradigm [25] can be partially or completely assembled from a pre-defined set of components with well-defined interfaces. Therefore, these are suited for tailoring a database system towards an application. However, there are aspects of database systems that are difficult to encapsulate into components with well-defined interfaces; typical examples include synchronization, transaction models, and database policies such as concurrency control [3]. These aspects are crosscutting concerns that permeate the whole system and affect multiple components. Hence, using traditional component-based approach is necessary but not sufficient to enable efficient development of configurable RTDBMSs. Therefore, in COMET we use an approach to aspectual component-based real-time system development (ACCORD) [30, 31] (discussed in section 3.1) that provides a notion of a reconfigurable component, and thereby enables both encapsulation of RTDBMS functionality into components and efficient handling and implementation of crosscutting concerns via aspects.

Using the ACCORD approach, different COMET components and aspects can be developed, and then used for assembling COMET configurations suitable for a specific automotive control system. Existing COMET components and aspects are discussed in section 3.2. We illustrate the COMET concepts introduced in this section with an example of the COMET configuration suitable for a particular node in the automotive control system in section 3.3.

3.1. Aspects and components in RTDBMSs

ACCORD utilizes notions from both component-based and aspect-oriented software development, integrating them into real-time system development. While CBSD traditionally use black box as an abstraction metaphor for the components, AOSD utilizes the white box metaphor to emphasize that all details of the implementation should be revealed. ACCORD supports the notion of a reconfigurable real-time component model (RTCOM) [26, 30, 31]. Components built using RTCOM are grey boxes as they are encapsulated in interfaces but changes to their behavior can be performed in a predictable way using aspects. Aspects are allowed to modify the code of the components in predefined, explicitly declared, reconfiguration points. In this section we briefly review RTCOM and its configurability via aspects, while detailed descriptions of ACCORD and RTCOM can be found in [26, 30, 31].

Aspects are programming-language level constructs encapsulating crosscutting concerns that invasively change the code of the component and correspond to the traditional aspects in existing aspect languages. The main constituents of aspects are: (i) components, written in a component language, e.g., C, C++, and Java; (ii) aspects, written in a corresponding aspect language, e.g., AspectC [6], AspectC++ [23], and AspectJ [33]; and (iii) an aspect weaver, which is a preprocessor that inserts code from the aspects into the