Database Proxy Tool Support in an AUTOSAR Development Environment

Andreas Hjertström, Dag Nyström and Mikael Sjödin
Mälardalen Real-Time Research Centre
Mälardalen University, Västerås, Sweden
{andreas.hjertstrom, dag.nystrom, mikael.sjodin}@mdh.se

***Abstract*—AUTOSAR has been introduced as a remedy for the increasing complexity and rising costs within automotive systems development. However, AUTOSAR does not provide sufficient support for the increased complexity with respect to data management. Database proxies have been presented as a promising solution to provide software component technologies with the capabilities of a state-of-the-art real-time database management system. In this paper, we show how an industrial AUTOSAR development environment can be extended to include support for real-time data management.**

***Keywords*-CBSE; RTDBMS; Real-Time; Embedded Systems; AUTOSAR;**

I. Introduction

This paper presents how real-time data management can be introduced into AUTOSAR development projects using a data management tool extension developed for the Arctic Core [1] AUTOSAR tool suite. The tool extension uses database proxies [2] that is a technique to provide run-time data management support to component-based real-time systems using a Real-Time Database Management System (RTDBMS).

Managing run-time data in complex embedded real-time systems is considered as one of the major challenges for the future [3], [4]. The automotive industry is continuously evolving and introducing new complex techniques, such as active safety and infotainment systems to improve the competitiveness of their products. This development introduces an increased dependency between vehicle-internal functions, as well as, increased demand for communication with external applications or infrastructure [5]. In turn, this increases the demand for open, secure, and flexible access to data. Solutions using a customizable database in an automotive context to manage the security aspects of interconnected systems have been proposed in [6].

It has been reported that the lack of techniques and tools for data management has led to use of ad hoc techniqes and redundant work in reinventing management of data for each Electrical Control Unit (ECU). Current solutions have reached their limits and are no longer adequate to deal with the future requirements on data, neither at design-time [7] nor at run-time [6], [8].

Database technologies are well established and have proven their usefulness to manage data in complex systems. In our previous work, database proxies have been presented as a promising solution to integrate an RTDBMS into a component-based setting [2]. In addition, the approach has been successfully implemented and evaluated on an AUTOSAR hardware node [9]. Database proxies allow components to communicate through their defined interfaces, remaining unaware of the database, while providing temporally predictable access to data maintained in a database. Database proxies use the state-of-the-art database pointer technique [10] which enable predictable hard real-time access to data along with a flexible SQL-based soft real-time interface. We have evaluated the efficiency of database proxies and shown that it is an affordable technique with very low overhead and large degree of flexibility [2]; thus we conclude that database proxies can be attractive technique when trying to improve data management in vehicular software.

In this paper, we present how database proxies can be integrated into the development of automotive systems using industrial tools. Our approach enables a clear separation of concerns between the system architect, component developer, and the DataBase Administrator (DBA). This separation of concerns allows each part to be managed and reconfigured independent from one another. Furthermore, a plug-in approach, developed for the Arctic Core tool suite and an integration of the Mimer SQL Real-Time [11] database into the basic software of AUTOSAR is presented.

In the remainder of this section, the main tools and technologies used are briefly presented.

A. AUTOSAR

Automotive Open System Architecture (AUTOSAR) [12] defines a standard component model and middleware platform for the automotive electronic architecture. AUTOSAR defines a set of layers to separate the underlying infrastructure from the interconnected software components. AUTOSAR employ the Component-Based Software Engineering (CBSE) approach, where software is encapsulated as components that communicate through well defined interfaces. The communication between system wide components

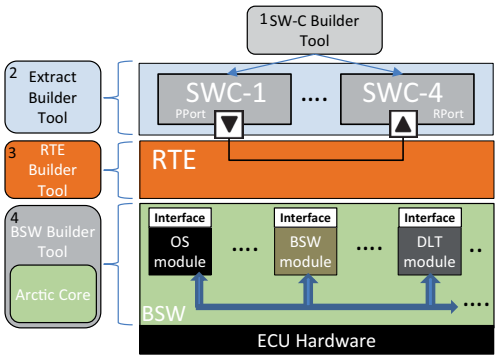

Figure 1. Arctic Core tool suite overview

is managed by a Virtual Function Bus (VFB), which acts as a virtual abstraction of the underlying hardware. The realization of the VFB to a concrete implementation of the final target system is the Run-Time Environment (RTE).

B. ArcCore - Development Environment

ArcCore AB [1] is a provider of the Arctic Core open-source AUTOSAR platform developed in Eclipse [13]. Arctic Studio is an Integrated Development Environment (IDE) for Arctic Core, which offers a number of professional graphical tools to facilitate development. Figure 1 shows the different tools provided in the Arctic Core tool suite. (1) Components and their interfaces are developed, configured, and generated using the SoftWare Component (SWC) builder tool. (2) The Extract Builder tool is used to add components to an ECU, connect ports and to validate the extract. (3) The Run-Time Environment Builder models the VFB and generates a run-time implementation of the component communication. (4) The configuration of the target platform e.g. operating system, communication, and memory etc., and generation of target c-code is done in the Basic Software Builder tool.

C. Mimer SQL Real-Time

Mimer SQL Real-Time (Mimer RT) [11] is a commercial RTDBMS that allows applications with both hard real-time and non real-time requirements to safely share data without risking real-time predictability. Hard real-time applications use the RTAPI interface to access data using database pointers [10] while non real-time applications use standard SQL interfaces. Mimer RT combines the standard client/server architecture for SQL queries to enable tools and/or applications to access to data both locally and remotely, with an embedded library architecture for real-time access.

D. Database Proxies

Database proxies [2] translates data between component ports and an RTDBMS that resides in the component framework (i.e., the AUTOSAR BSW) and vice versa. This allows full decoupling of the RTDBMS from the component, i.e., the component and the RTDBMS are unaware of the existence of the other. Database proxies remove the need for database calls within the component, thus preserving component encapsulation and enable component reuse. The schema of the database is modeled and optimized separately and is thereby independent of the component implementation.

The internal mapping of data to the components is performed either through *hard database proxies* that uses time-predictable database pointers for components with hard real-time requirements, or *soft database proxies* that utilizes flexible SQL for components with soft real-time requirements. A pre-compiled database statement enables a developer to compile and bind a certain database query to a statement. This has a decoupling effect since the internal database schema is hidden from the component.

Database proxies are shown to be an efficient and predictable technique that offers a range of valuable features to real-time embedded systems development, maintenance and evolution at a minimal cost with respect to resource consumption [2].

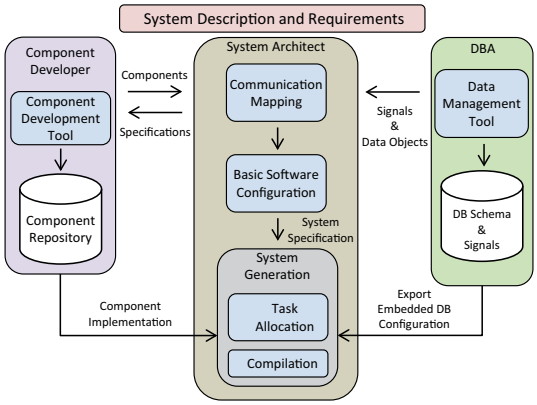

Figure 2. System development roles

II. System Development Roles

To support efficient use of database proxies in a development setting we divide system development into the following three roles, see figure 2:

- **The DataBase Administrator (DBA)**, which manages the database design, data modeling & optimization, managing access rights, and generation of the database configuration. Database objects are mapped to signals, which are used by the system architect. With this approach, a complete separation of concerns is achieved, since the database model is separated from the run-time signal. This allows for a reconfiguration of the database without affecting the system configuration. At compile-time, an embedded custom-made database is synthesized into the final system.
- **The component developer**, which provides the system architect with a set of components according to

specifications provided by the system architect. The component developer takes no consideration of database connections and calls, since database proxies provide a full separation of concern between these entities. The component implementation details are provided to the system generation and compilation.

- **The system architect**, which has the overall architectural system view. The system architect assembles the system from components available in the repository or provides the component developer with specifications. Components ports are mapped to signals provided by the DBA. BSW modules such as operating system, memory, and network communication are configured and components are mapped to tasks. Finally the system is compiled and ready to deploy.

This approach enables the three roles to be fully decoupled from each other. Thus, the tasks of each role can be performed independently.

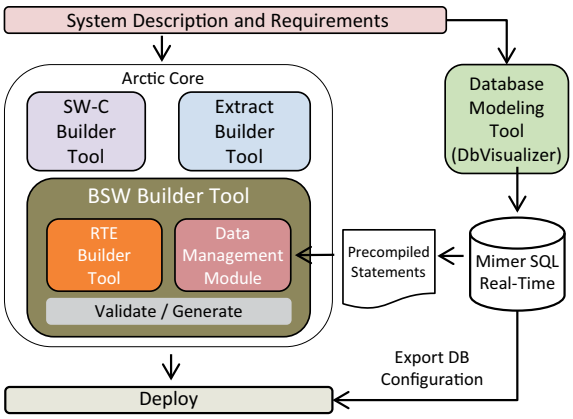

Figure 3. Data management module integration overview

III. Data Management Tool Suite Extension

This section describes the proposed database proxy integration into the Arctic Core tool suite. The tool suite is extended to support the three development roles introduced in section II. Figure 3, illustrate the tool suite extended with database management support. In short, a database modeling tool and a data management module plug-in has been introduced in the BSW builder. Since database proxies decouple the RTDBMS from the components, the SWC-builder is kept as is. Finally, the tool suite is extended with a synthesis tool that incorporates the database and database proxies into the run-time system.

A. The Database Modeling Tool

During the development, the DBA uses a standard database modeling tool (DbVisualizer [14]) to model, optimize and implement the database schema. DbVisualizer was selected because of its tight integration with the Mimer SQL Real-Time database and its support for precompiled statements. When the database is modeled, the DBA maps database values to signals using precompiled statements that are stored in the database. These signals are later used to map database proxies to objects in the database.

B. The Data Management Module

The Data Management Module, which is implemented as an Arctic Core Eclipse Plug-in, allows the system architect to create database proxies to connect component ports to signals in the database.

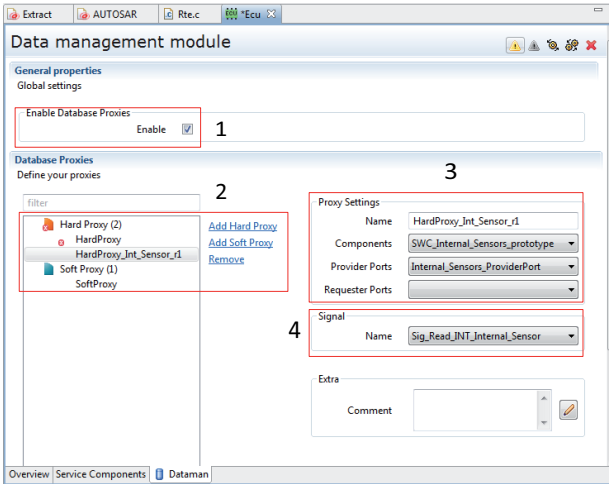

Figure 4. The Data management module

The Data management module includes four main configuration settings for each added database proxy, see figure 4:

1) A global setting to enable the usage of database proxies and to include the Mimer RT in the BSW.
2) A list of all hard and soft proxies in the system.
3) A configuration area where the database proxy is named and assigned to either a provider or requester port of a component.
4) A configuration area where the database proxy is associated with a specific signal in the database. The tool automatically retrieves the available signals from the database.

C. Synthesis Tool

When the runnable system is generated, a synthesis tool is called that automatically performs the following:

- Extraction of the relevant information from the development database and transformations of this into an embedded database file that is optimized for execution on the target platform.
- Generation of the initialization code for database proxies. This includes the creation of the *proxies.h* and *proxies.c* files which contain the initialization and uninitialization

```
/*** Original Arctic Core code ***/
void Rte_ActuatorRunnable(){
 Rte_PRE_ActuatorRunnable();
 ActuatorRunnable();
}
void Rte_PRE_ActuatorRunnable(){
DisableAllInterrupts();
Rte_ReadBuffer_Rte_Buf_Actuator_RPort_1_data_1
(&Rte_Inst_Actuator.Act_RPort_1_data_1->value);
EnableAllInterrupts();
}
/*** Database proxy code ***/
void Rte_ActuatorRunnable(){
Rte_PRE_ActuatorRunnable_DBProxy();
 ActuatorRunnable();
}
void Rte_PRE_ActuatorRunnable_DBProxy(){
MimerRTGetInteger(DBP_Actuator, &Rte_Inst_
Actuator.Act_RPort_1_dataElem_1->value);
}
```

Figure 5. Example of original code and database proxy code

functions of the database proxies that the RTE invokes during system startup and shutdown [2].

- Substitution of the original ArcCore component communication code with the database proxy component communication code.

Figure 5, shows the differences between the original communication code sequence and the code sequence that uses database proxies. In the original code, *Rte_ActuatorRunnable* is called from within the task. Since the component has a requester port, the call to retrieve the data from memory is made before the component itself is called. The *Rte_PRE_ActuatorRunnable* function disables all interrupts, calls an additional function that reads the data from the buffer, and enables all interrupts. After this, the component is called.

When using database proxies, *Rte_ActuatorRunnable* calls the *Rte_PRE_ActuatorRunnable_DBProxy* function to read the value from the database using a database pointer. In difference to the original code, the interrupt disable is not needed within the database proxy, since Mimer RT manages this.

Since Mimer RT in addition to the RTAPI provides a standard SQL-based interface, data access is not limited to database proxies. Data can also be made available to external tools and 3rd party applications as well as to other internal BSW modules such as the Data Log and Trace (DLT) module that can utilize the data for diagnostics purposes.

IV. Conclusions and Future Work

This paper has presented an approach for introducing real-time data management in a commercial AUTOSAR development tool suite. This has been made possible with the usage of database proxies that is a technique to provide predictable run-time data management support for component-based real-time embedded systems. Our approach, together with a proposed tool support, enables a clear separation of concerns for the system architect, component developer and the database administrator.

As future work, we plan to complement the tool suite presented in this paper to support automated validation and an augmentation of our existing synthesis tool [2] to fully support the AUTOSAR component model.

References

[1] ArcCore, "Open Source AUTOSAR Solutions, Göteborg Sweden," http://www.arccore.com.

[2] A. Hjertström, D. Nyström, and M. Sjödin, "Data Management for Component-Based Embedded Real-Time Systems: the Database Proxy Approach," *Journal of Systems and Software*, vol. 85, pp. 821–834, April 2012.

[3] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software Engineering for Automotive Systems: A Roadmap," *Future of Software Engineering*, pp. 55–71, 2007.

[4] S. Schulze, M. Pukall, G. Saake, T. Hoppe, and J. Dittmann, "On the Need of Data Management in Automotive Systems," in *BTW*, ser. LNI, J. C. Freytag, T. Ruf, W. Lehner, and G. Vossen, Eds., vol. 144. GI, 2009, pp. 217–226.

[5] M. Z. Gereon Weiss and D. Eilers, *Towards Automotive Embedded Systems with Self-X Properties*, M. C. (Ed.), Ed. InTech, 2011.

[6] M. P. G. S. Thomas Thüm, Sandro Schulze and S. Günther, "Secure and Customizable Data Management for Automotive Systems: A Feasibility Study," *ISRN Software Engineering*, vol. 2012, 2012.

[7] A. Hjertström, D. Nyström, and M. Sjödin, "A Data-Entity Approach for Component-Based Real-Time Embedded Systems Development," in *14th IEEE International Conference on Emerging Technology and Factory Automation*, Sept 2009.

[8] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, "Engineering Automotive Software," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356 –373, feb. 2007.

[9] A. Hjertström, D. Nyström, and M. Sjödin, "Introducing Database-Centric Support in AUTOSAR," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES12)*, June 2012.

[10] D. Nyström, A. Tešanović, C. Norström, and J. Hansson, "Database Pointers: a Predictable Way of Manipulating Hot Data in Hard Real-Time Systems," in *Proceedings of the 9th International Conference on Real-Time and Embedded Computing Systems and Applications*, 2003, pp. 623–634.

[11] Mimer SQL Real-Time Edition, Mimer Information Technology, Uppsala, Sweden, http://www.mimer.se.

[12] AUTOSAR Open Systems Architecture, http://www.autosar.org.

[13] The Eclipse Foundation, Ottawa, USA, http://www.eclipse.org/.

[14] DbVisualizer, "DbVis Software AB, Stockholm, Sweden," http://www.dbvis.com/.