INCENSE: Information-Centric Development of Component-Based Embedded Real-Time Systems

Andreas Hjertström
Mälardalen Real-Time Research Centre, Västerås, Sweden
andreas.hjertstrom@mdh.se

*This work is supported by the Swedish Foundation for Strategic Research within the PROGRESS Centre for Predictable Embedded Software Systems.
## Contents

1. Research Introduction .................................................. 3

2. Problem Background and Description ................................. 4

3. Research Method ......................................................... 5
   3.1. Hypothesis .................................................................. 6
   3.2. Expected Outcome .................................................... 6

4. Research area and techniques overview .............................. 6
   4.1. Database Management Systems .................................... 6
   4.2. Real-Time Database Management Systems ..................... 8
   4.3. Component-Based Software Engineering ........................ 8
   4.4. PECOS .................................................................... 8
   4.5. Koala ...................................................................... 10
   4.6. SaveCCT Real-Time Component Technology ................ 10
   4.7. Rubus ..................................................................... 11
   4.8. COMET Real-Time Database ...................................... 11
   4.9. Polyhedra .................................................................. 12

5. Related Research and Research Groups .............................. 12

6. Key conferences .................................................................. 13

7. Project Planning ............................................................... 14
   7.1. Activities ................................................................... 14
   7.2. Planned papers .......................................................... 14
   7.3. Time plan and milestones ............................................ 15

8. Conclusions ...................................................................... 16
Abstract

This paper summarizes the basics concerning my research topic, methodology and a brief description of some related work within the areas of Component Based Software Engineering (CBSE) and Real-Time Database Management Systems (RT-DBMS). RTDBMS and CBSE are highly interesting upcoming techniques which both have great benefits. What we will do in our research is to integrate RTDBMS as a part of the CBSE framework. Included in this paper is also some key conferences and interesting references are also presented in this paper. I present a plan about my future goals, courses and upcoming papers.

1. Research Introduction

In my research I’m going to look into the problem of how to reduce the ever increasing complexity in today’s embedded real-time systems. The research is aimed at embedded real-time systems in general but specifically to vehicle systems. The project I am involved in is called Incense (Information-Centric Development of Component-Based Embedded Real-Time Systems). Incense aims at joining two strong upcoming techniques real-time database management systems (RTDBMS) and component based software engineering (CBSE). To integrate an RTDBMS as a part of the component-framework (run-time support is however not trivial since there are several differences and conflicting aspects that differ these two.

It would be desirable to achieve a component based system where data is reliably managed and structured to enable flexibility, a system where soft and hard real-time tasks can execute and keep isolation properties, a system that can handle critical transactions and at the same time enables openness, a system where new functionality can be added or removed without side effects to the system and all communication is done through component interfaces.

A first part has been to solve the problem of successfully integrate RTDBMS as a part of CBSE framework without jeopardizing important properties such as, isolation and reusability. To be able to solve this we have introduced database proxies as a connecting between a component and the database schema. This makes the component unaware of the how the database are constructed. As a consequence of this components do not loose the possibility of reuse. All database interactions, both soft and hard real-time transactions, are performed with this technique.

For components which needs to have deterministic and efficient access to individual data elements, a hard proxy is used. These hard real-time components use 2V-DBP protocol which provides a constant response time. A soft component often requires more complex data-structures and uses a soft proxy. As well for hard proxys, soft proxys are connected to a components out or in ports. This approach enables a more open system where standardized query languages can do run-time database queries.
Another part of my work will be to develop an information-centric design method for component-based real-time systems (design support), consisting of high level tools and design paradigms to manage and organize data in a logical view rather than a physical. During design, developers should have full control of each data item involved, who are the producers/consumers, timing requirements etc. The overall aim in this part of our work is to create an information-centric design paradigm for real-time systems, where data management is treated as its own design entity.

The two main technologies used in this project is COMET RTDBMS [12] and SaveCCT component technology[1].

2. Problem Background and Description

In vehicular systems there are a number of computer nodes, called Electronic Control Units (ECUs), often developed by different hardware vendors, controlling engine, brakes, gearbox etc. The cost for development of electronics in high-end vehicles have increased to more than 23% of total manufacturing cost [5]. Including subsystems an automotive system can include over 70 ECUs communicating on different networks through, in today’s cars 2500 signals [10]. This leads to an increasing amount of data that needs to be managed. A lot of the tasks involved in these are critical hard real-time transactions, often operating at high frequencies and updated periodically. Furthermore, there is an increase of tasks running non-critical, soft transactions in the system at lower frequency. These transactions, often read transactions, use the same data as hard critical tasks for logging or to present statistical information about the current state of the vehicle to the user.

To handle the increasing complexity in these systems, new approaches and design paradigms to reduce complexity are needed, since current techniques (internal data structures) are becoming increasingly insufficient. Two upcoming approaches to reduce complexity are Component-Based Software Engineering (CBSE) and DataBase Management Systems (DBMS). Real-Time Database (RTDB) [15] and RTDBMS (Real-Time Database Management System) are upcoming technologies both within research society and in industry [9] to help developers solve information management problems regarding synchronization, deadlock and persistency. Main focus in this area has mainly been towards concurrency-control, temporal consistency, overload management and scheduling.

Vehicular industry has already identified the problem of accelerating number of ECUs and large development costs. They have together tried to attack this problem by creating a standard called, Automotive Open System Architecture (AUTOSAR) [7].

Focus within CBSE is to create software components to be reusable entities mounted together as building blocks with a possibility to maintain and improve systems by replacing individual components[3]. Even though RTDBMS and CBSE share the same goal, their means of achieving it is unfortunately conflicting. One
key philosophy for most component models is that all communication with the surrounding environment should be performed through the component’s interface, i.e., eliminate all component side-effects. By introducing a real-time database management system (RTDBMS) and giving components direct access via shared data in the database introduces such side-effects. Furthermore, constructing components that query a certain database engine, with a certain database structure (schema) severely reduce the component reuse.

3. Research Method

A first step has been read into the two main technologies within my research area and get an understanding of the problem at hand. I knew some basics concerning components and real-time databases but needed more in depth knowledge. This first part of the research to be able to integrate a RTDBMS into a component framework using proxies is based on earlier ideas.

A next step was to summarize ideas of how to use a database proxy to make components database unaware and to combine CBSE and RTDBMS with run-time support, into a WiP paper which has been accepted to RTAS conference. To continue this work a further step will be to continue development of these ideas of how to integrate a real-time database into a component-based framework and still keep the important aspects such as reusability. These ideas will also result in an implementation.

To get more understanding of the process concerning how data is treated within industry, I will investigate and gather information and requirements regarding the current usage of data and signals in embedded real-time systems. 3-5 different companies in different industrial areas will be included.

A selection of suggested questions included in this case-study on data as an entity are:

(i) How is data handled today? (ii) Are all signals in the system specified? (iii) Is it possible to search among system signals? (iv) How is the connection towards documentation? (v) Who has "ownership" of a data? How approves usage etc. (vi) How are critical and non-critical data treated? (vii) Are all facts about a data documented? (viii) When during the process is the decision made regarding interface? (ix) Order of adding a new component to the system? (x) A removed/added data, producer/consumer notified or documented? (xi) Time demands on data, are they documented? (xii) Who has overview and responsibility? (xiii) Connection to bus capacity, time-slot?

When this part is done I will be able to evaluate the information received from the case-study. This is not something that I will build my research on, but rather to an overview investigation to get a better understanding of the current situation.
We also have plans to extend the information-centric view with high level tools and design paradigms to manage and organize data in a logical view rather than a physical. During design, developers should have full control of each data item involved, who are the producers/consumers, timing requirements etc. The overall aim of this part is to create an information-centric design paradigm for real-time systems, where data management is treated as its own design entity.

Some of the work within these different areas of research and implementation will hopefully be performed in cooperation with some students thesis.

3.1. Hypothesis

To develop a framework suitable to meet demands of today and tomorrow where a RTDBMS supporting both hard and soft real-time transactions is predictably integrated into CBSE, a framework where data is lifted and introduced already at design level and treated as its own entity. To manage this our aim is to introduce an example where we use Mimer SQL RT edition integrated with Save CCT in one information-centric framework together with high level tools.

3.2. Expected Outcome

1. A successful integration of a RTDBMS into a CBSE framework where already at design-time introduce data as its own entity.

2. To organize data, develop high level tools and design paradigms.

4. Research area and techniques overview

4.1. Database Management Systems

A Database Management Systems (DMS) is most often used to organize and structure large amounts of data. Typical applications has so far been large enterprise systems such as libraries, commercial web-sites and banking. Examples of enterprise mainstream DBMS are Oracle, Microsoft Access and MySQL. Increasing amount of data and growing data complexity has made DMS useful also in smaller embedded systems.

The main purpose of DBMS is to provide a number of software programs to organize data. The three most used methods for organization is to organize data relational, hierarchical and network. An example of a relational organization is Structured Query Language (SQL). SQL is one of the most common language for uniform query and updating data. SQL enables high level tools to require desired information from large amount of data.

DMS have several important parts which includes a query language, optimized data structures and mechanisms for various transactions. Figure 1 shows an high level picture of a DBMS system. As an example, an application requests a data
Figure 1. DBMS overview

with a database query from the DBMS which processes the query, retrieve the data from database and returns to the application. An important part is the transaction mechanism properties. To ensure a correct behavior of the database it should apply to the ACID properties.

ACID properties ensure:

- **Atomicity**, a transaction runs to completion or not at all.
- **Consistency**, according to the integrity constraints of the database. The database have to be in a legal state before a transaction begins and when it ends.
- **Isolation**, a database transaction should isolated from all other operations. No other transaction can see data thats completely processed by another transaction.
- **Durability**, when a data has been committed it is permanent.

Embedded systems is a quite new area for DBMS. These systems have different requirements on the DBMS compared to large enterprise systems. For embedded systems, cpu usage, footprint, availability and specialized operating systems suitable for embedded systems[13].

- **CPU usage**,

Since embedded systems are resource constraint systems the amount of cpu usage is a very important. The same argument goes for the amount of memory needed by the DBMS, called footprint, is also important. Most embedded systems run without any system administrator to deal with possible problems which calls for low failure rate and high availability.
4.2. Real-Time Database Management Systems

"A real-time database (RTDB) is a data store whose operations execute with predictable response, and with application-acceptable levels of logical and temporal consistency of data, in addition to timely execution of transactions with the ACID properties", C. D. Locke Chief Scientist, TimeSys Co.

Real-Time Database Management Systems (RTDBMS), has some additional requirements to meet. A RTDBMS is similar to a DBMS but executes transactions according to deadlines. Because a RTDBMS has to ensure transactions to meet their deadlines it has to be predictable and a worst-case scenario has to be known. This can be done with off-line scheduling or using predictable algorithms. Central for RTDBMS is that transactions has to be committed before a specified deadline but temporal validity is also a important. A sensor value can for instance be valid for a certain number of seconds and there after useless. There can be two types of real-time transactions, soft and hard. A hard transaction has to meet its deadline whereas a soft transaction can miss its deadline without catastrophic consequences.

Some of the benefits of a RTDBMS are:

- Consistency and concurrency control in applications
- A more efficient way of handling large quantities of data
- Improved timeliness and predictability
- Overload and recovery management.

4.3. Component-Based Software Engineering

In Component-Based Software Engineering, the aim is to have a high level of abstraction and divide systems into building blocks or components, with well defined communication interfaces. One of the major benefits of this technique is to make components reusable entities that can be put together as building blocks. In this way a lot of development effort and cost can be saved.

4.4. PECOS

PECOS[11] (pervasive component systems), is a European project component technology developed by industry in cooperation with researchers, targeting the automation industry constructing embedded systems such as "field devices". A central part of PECOS component technology is their component model aimed at embedded devices. For PECOS as in most cases of component based engineering the main focus is reuse, shorter time-to-market and reduced complexity. Interaction between components is only done through data ports. Because of the design choice
to make PECOS component communication only where ports are connected, its is
quite easy to analyze timing aspects with a technique called rate monotonic [16].
It also enables schedule verification using constant logic programming approach.
The number of concurrent tasks needed to implement the system is also down to a
minimal.

A PECOS component can be either active, passive or event driven[6].

- Active components control its own thread and models ongoing activities with
  no short cycle-time for completion. A composition of components is always
  modeled as active and schedule them to meet the included components timing
  constraints.

- A Passive component however has no thread of its own to control and are
  scheduled by the closest active parent. Opposite to an active component,
  passive components are used to encapsulate behavior and has short cycle time
  for completion.

- Event components are event triggered, mostly used to model hardware that
  periodically generates events.

A typical PECOS Component development process is done in several different
steps if there are no existing components in the component repository.

- **Requirements, Election and Analysis**, functional and non functional com-
  ponent requirements, timing constraints, memory consumption is collected
  as well as input and output flows.

- **Interface design**, is formally defined with the COCO language, an archi-
  tectural description language, where both information such as data ports and
  properties such as WCET and execution time can be found. When scheduled
  the component reads form its in-ports and writes to its out-ports. Also the
  kind of component has to be specified from the three choices active, passive
  and event driven.

- **Component Implementation**, Is currently supporting C++ and Java. Only
  one language within a PECOS component is supported. A tool generates the
  class code where component functions are to be implemented. The method
  interfaces are different depending on the type of component that are used
  (active, passive or event).

  Composed components are validated in three steps, structurally checked for ad-
  herence, timing aspects such as deadline (if WCET and maximum blocking tim are
  known) and if the chosen schedule is valid or if one should be generated[6].
4.5. Koala

Koala[17], a component technology for resource constraint consumer electronics originally designed by Philips for TV products. Because of the increasing complexity, demands on short Time-To-Market together with a more and more competitive market Philips wanted to extend and reuse existing products. Although the software components often are quite small there are many developer man months invested. In this way they save time, development cost and can use the same software components in several of their products. To help developers to manage and structure configuration koala consists of an explicit architecture where it is possible with the help of an architectural description language (ADL), is a valuable tool for developers to visualize the system, see Figure 2. The component model are made recursive so that any component assembly can be viewed as one component with a set of interfaces. They also use static bindings when possible to minimize overhead. Koala was inspired of COM but had their own requirements.

There is a strict separation between component and configuration development. No assumptions about component configuration are made by the component builders and configuration designers are not allowed to change a components configuration. Component communicates trough globally unique environment interfaces using a small set of semantically related functions, even for memory management. This gives system architects a good view of system resource usage. Just like in COM convention, a Koala interface can not be changed once defined [18].

Koala also uses an interfaceless module that glues component interfaces together. A module implements all functions of components who has its tip connected to it.

4.6. SaveCCT Real-Time Component Technology

The SaveComp Component Technology (SaveCCT) [1] is described by categorize manual design, automated activities, and execution. The entry point for a devel-
oper is the Integrated Development Environment (IDE), a tool supporting graphical composition of components, where the application is created. Developers can utilize a number of available analysis tools with automated connectivity to the design tool. SaveCCT is based on a textual XML syntax which allows components and applications to be specified. Automated synthesis activities generate code used to glue components together and allocate them to tasks. Resource usage and timing are resolved statically during the synthesis instead of using costly run-time algorithms. SaveCCT is, as Mimer RT, intended for applications with both hard and soft real-time requirements.

![Figure 3. Save graphical application design](image)

In SaveCCT applications are built by connecting components input and output ports using well defined interfaces, see Figure 3. Components are then executed using trigger based strict "read-execute-write" semantics. A component is always inactive until triggered. Once triggered it starts to execute by reading data on input ports to perform its computations. Data is then written to its output ports and outgoing triggering ports are activated.

4.7. Rubus

Rubus Component Model [2], developed by Articus systems and researchers, is used today by Volvo Construction Equipment. Rubus is used for resource constrained systems and supports both soft and hard real-time embedded systems which are distinguished by two separate kernels. Hard real-time applications are time-triggered and soft real-time applications are event-triggered. Development environment is tightly integrated with Rubus operating system which makes Rubus Component Model less portable.

4.8. COMET Real-Time Database

COMET RTDBMS [12] is a Real-Time database management concept that is well suited for automotive real-time control-applications. COMET allows both soft and hard real-time database transactions to coexist in the same application without jeopardizing concurrency or consistency rules. The algorithm used to achieve this is called 2-Version DataBase Pointer concurrency control (2V-DBP) has two
different user interfaces which makes soft and hard transactions to coexist without compromising real-time properties of the hard transactions[14]. It eliminates abortions caused by hard transactions and avoid starvation of soft transactions from hard. In this way there are no long delays for hard transactions since hard transactions has direct assess to data. Similar to what Kuo et al.[8] proposed a two-version priority ceiling protocol, two versions of all data elements connected must exist in the database.

A queries from soft transaction is done with SQL and uses 2PL-HP protocol. In this way its possible to do soft ad hoc queries at any time. A completely different approach is used for hard transactions. A database pointer binds to a specific data. To ensure atomicity only one data access is allowed. This is done in hardware with interrupt disable. It has been proven to be consistent and data that has not yet been committed is visible to other transactions. 2V-DBP has also been shown to outperform 2PL-HP protocol.

4.9. Polyhedra

Polyhedra is a family of fault-tolerant databases developed by Enea [4]. It is a relational SQL (DRBMS) database management system based on a client-server architecture that has been successfully used world wide in embedded systems. IT includes fault tolerance, instant failover and fast reconnection to ensure availability. It is available for several platforms such as Windows CE, Embedded Linux and VxWorks. To increase performance Polyhedra also supports active queries to avoid the need for polling. To ensure data persistence, journal logging is used.

5. Related Research and Research Groups

There are several different research areas on real-time databases and there are extensive use within industry. An example of this is the stock-trading market. They have to handle massive amounts of data distributed all over the world in real-time. There are also a lot of research within the area of Component Based Software Engineering (CBSE), both within the research community and within several different industry areas. However research to combine these two technologies into one framework has so far been sparse.

As I stated above I could not find any specific group targeting our research area but are leading research groups, working within the areas of RTDBMS and CBSE.

**Real-Time and Embedded Systems Laboratory**, Department of Computer Science University of Virginia is an active group within real-time databases. Professor Sang H. Son is one of the leading researchers.

**The Distributed Real-Time Systems Research Group**, Skövde University, Sweden. Main area is to study real-time database systems with soft, firm and hard
deadlines, reactive mechanisms and event monitoring, as well as software timeliness testing based on techniques developed in this area.

http://www.his.se/iki/research/drts

Save, Mälardalen Real-Time Center, Västerås, Sweden, SAVE will develop a general framework for component-based development of safety-critical vehicular systems.

6. Key conferences


- **(RTECC) Real-Time and Embedded Computing Conferences**, are complimentary, technical and educationally-focused seminars, workshops and keynotes.

- **(RTCSA) IEEE International Conference on Embedded and Real-Time Computing Systems and Applications**, is aimed for researchers and developers from academia and industry for advancing the technology of embedded and real-time systems and ubiquitous computing applications.


- **(RTAS) IEEE Real-Time and Embedded Technology and Applications Symposium**, usually includes both workshops and tutorials.

- **Euromicro Conference on Real-Time Systems**, The conference has very strong roots within the Real-Time Laboratories and research groups in both Academia and Industry throughout Europe and across the world.

- **(ICSOFT) International Conference on Software and Data Technologies**, The conference tracks are "Software Engineering", "Information Systems and Data Management", "Programming Languages", "Distributed and Parallel Systems" and "Knowledge Engineering".
7. Project Planning

7.1. Activities

Attended Conferences


In this paper we address how important it is to consider the constant increase of complexity in embedded systems. Our proposition is to use two exiting techniques, CBSE and RTDBMS which both aim to reduce complexity both during development and during run-time. To integrate a RTDBMS into CBSE framework is not trivial. To give a component direct access to the database schema makes a component database aware which effects the possibility of component reuse.

In this paper we present a technique to use a database proxy as a connection between components and the database. This technique makes components database unaware and therefore reusable. To achieve this we have added some additional information to the SaveCCT framework. To get a predictable access to data and a possibility to use powerful on-the-fly queries to the database we use Mimer SQL Real-Time Edition, (based on the COMET project).

2007-05-26, ICSE, Workshop on Software Engineering for Automotive Systems. I will not present on this conference, but it is an important conference and an excellent opportunity to learn more about the vehicular domain.

7.2. Planned papers

We the planned case study on how data as an entity is treated in industry today has been conducted. The gathered information, result and conclusions will be presented in a paper.

What we want to achieve with this paper is to get a basic idea of data is controlled throughout its life-cycle. Who is responsible for how data is used and by who. Are the producers and consumers documented and how are timing demands and criticality considered.

As a result of some additional research an extension of the WiP paper accepted to RTAS conference with some deeper information and dept on how we successfully are going to integrate RTDBMS into a CBSE framework. Hopefully we will have some implementation were we can show some of our ideas presented in this paper.
<table>
<thead>
<tr>
<th>Year</th>
<th>Courses</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>RTDBMS Skövde</td>
<td>5p</td>
</tr>
<tr>
<td></td>
<td>Progress reading course</td>
<td>5p</td>
</tr>
<tr>
<td></td>
<td>Doktorandskolan</td>
<td>3p</td>
</tr>
<tr>
<td></td>
<td>Summer school</td>
<td>1p</td>
</tr>
<tr>
<td>2007</td>
<td>Research planning</td>
<td>3p</td>
</tr>
<tr>
<td></td>
<td>Real-time fk</td>
<td>5p</td>
</tr>
<tr>
<td></td>
<td>Winter school</td>
<td>1p</td>
</tr>
<tr>
<td>2008</td>
<td>Pedagogic</td>
<td>5p</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td>Paper B Extended WiP</td>
</tr>
<tr>
<td>2010</td>
<td>Statistics</td>
<td>5p</td>
</tr>
<tr>
<td>2011</td>
<td></td>
<td>Paper D Design time support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PhD</td>
</tr>
</tbody>
</table>

Figure 4. Suggested PhD Time-plan

A further step will be to implement the low level research in at a higher level, earlier into the development face. A paper to present the research and extend the information-centric view with high level tools and design paradigms where data is organized and controlled together at design time. As an example, when a developer at design time connects two components and thereby enables data access between these two, he/she should be able to have full control of the data. Are there any other transactions using this data, what are the timing requirements on the data, who are producing the data, etc.

7.3. Time plan and milestones

I have tried to foresee what my future will bring. Figure 4 shows an overview of the time-plan and milestones towards my PhD. The table should be seen a dynamic plan that could change before I reach my PhD.

Attended courses so far:
- RTDBMS, Skövde
- Progress reading course, Västerås
- Doktorandskolan, Västerås
- Summer school, Nässlingen
- Winter school, Trento Italy
• Research planning, Västerås

8. Conclusions

I my opinion, this to reduce the growing complexity in development and main-
tenance of today’s real-time embedded systems area is going to be more and more
important. Hopefully some of the ideas that we will present can be used outside the
research world to help with this.

The fact that there are not so many people working within this area "yet" and that
this is a step towards something new is very exiting.

How exactly the methodology within my research will be performed, will proba-
bly change over time. Our first paper has been accepted to RTAS, Work-in-Progress
session. The conference will hopefully give me some new ideas regarding my re-
search.

The suggested time-plan is meant as a way to give me some kind of heading to-
wards my PhD. The content of the papers will probably change some over time and
also the planned courses.

References

[1] Mikael Åkerholm, Jan Carlson, Johan Fredriksson, Hans Hansson, John
Håkansson, Anders Möller, Paul Pettersson, and Massimo Tivoli. The Save
Approach to Component-Based Development of Vehicular Systems. Journal


[5] Leen Gabriel and Hefferman Donal. Expanding Automotive Electronic Sys-

[6] Thomas Genßler, Alexander Christoph, Michael Winter, Oscar Nier-
strasz, Stéphane Ducasse, Roel Wuyts, Gabriela Arévalo, Basti-
aan Schönhage, Peter Müller, and Chris Stich. Components for
embedded software: the pecos approach. In CASES ’02: Proceedings of the
2002 international conference on Compilers, architecture, and synthesis for

[7] Harald Heinecke, Klaus-Peter Schnelle, Helmut Fennel, Jürgen Bortolazzi,
Lennart Lundh, Jean Leflour, Jean-Luc Maté, Kenji Nishikawa, and Thomas
Scharnhorst. Automotive open system architecture - an industry-wide initia-
tive to manage the complexity of emerging automotive e/e-architectures.


