Design-Time Management of Run-Time Data in Industrial Embedded Real-Time Systems Development*

Andreas Hjertström, Dag Nyström, Mikael Nolin and Rikard Land
Mälardalen Real-Time Research Centre, Västerås, Sweden
{andreas.hjertstrom, dag.nystrom, mikael.nolin, rikard.land}@mdh.se

Abstract

Efficient design-time management and documentation of run-time data elements are of paramount importance when developing and maintaining modern real-time systems. In this paper, we present the results of an industrial case-study in which we have studied the state of practice in data management and documentation. Representatives from five companies within various business segments have been interviewed and our results show that various aspects of current data management and documentation are problematic and not yet mature. Results show that companies today have a fairly good management of distributed signals, while internal ECU signals and states are, in many cases, not managed at all. This lack of internal data management results in costly development and maintenance and is often entirely dependent of the know-how of single individual experts. Furthermore, it has, in several cases, resulted in unused and excessive data in the systems due to the fact that whether or not a data is used is unknown.

1 Introduction

Most of today's embedded system developers are experiencing a vast increase of system complexity. The growing amount of data, the increasing number of electrical control units (ECUs) and inadequate documentation are in many cases becoming severe problems. The cost for development of electronics in for instance high-end vehicles, have increased to more than 23% of the total manufacturing cost. These high-end vehicle systems contain more than 70 ECUs and up to 2500 signals [1, 7].

A lot of research has been done in the area of run-time data management for real-time systems. This has lead to the development of both research-oriented data management solutions, such as [2, 14, 18], and commercial real-time data management tools, such as [4, 17, 22]. However, in most cases this research and these tools focus on run-time algorithms and concepts, but do not manage data documentation. In this case-study we investigate state of practice in design-time data management and documentation of run-time data in industrial real-time systems. An earlier case-study on data management in vehicle control-systems has indicated a lack of data management and documentation internally in the ECUs [19]. The case-study in this paper covers a broader scope of companies, and focuses on the development process and documentation of real-time data.

The study includes five companies, four vehicle companies active in different domains and one company producing electrical control systems. The study identifies ten problem areas in the development process and suggests remedies and directions for further studies. Furthermore, we show that the importance of adequate data management is growing along with the increasing complexity of real-time and embedded systems [9].

The main observation from our study is the rudimentary, or in some cases total lack of, data management and data documentation for internal ECU data. This should be compared to distributed network data that, due to adequate tool support, are fairly well managed and documented. We observed that this lack of management, in some cases leads to inadequate development routines when handling data.

Currently, companies developing safety-critical systems are becoming increasingly bound to new regulations, such as the IEC 61508 [11]. These regulations enforce stronger demands on development and documentation. As an example, for data management it is recommended, even on lower safety levels, not to have stale data or data continuously updated without being used. Companies lacking techniques for adequate data management and proper documentation will be faced with a difficult task to meet these demands.

The main contributions of this paper include:

- A case-study investigating state-of-practice in data management for real-time systems.
- Ten identified problem areas in current practice.
- Suggestions of remedies and future research directions.

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

The outline of the following parts of the paper is as follows. Section 2 describes our research method and the five participating companies. Section 3 reports the state-of-practice in data management and documentation of industrial embedded real-time systems. In section 4 we present four key observations and ten identified problem areas in current practice. In section 5 we propose six remedies and future research directions. In section 6 and 7 we conclude the paper and suggest future work based on the findings in this case-study.

2 Research Method

This qualitative case-study [13] has been conducted at five companies, mostly in various vehicular business segments, developing systems in various application areas within the embedded real-time domain. The main source of information has been interviews with open-ended questions [21] conducted at the companies. One person at each company with in depth knowledge of their system development, both on high and low level were interviewed. All interviews have, after promises of anonymity, been recorded to be able to have open discussions that could later be evaluated.

The work-flow of these interviews has been as follows; (i) the interviewee was contacted and asked to take part in this interview with a short explanation of the contents. (ii) A short summary explaining our area of interest was sent one week before the interview. (iii) The interview was executed, and set to last for approximately one hour. (iv) After each interview, the recording from the interview was analyzed and the answers written down as a summary question by question. (v) A document with all of the questions and their respective summaries where sent back to the interviewee for possible commenting and approval. In some cases, the document included additional requests for clarification of certain areas.

The interview questions were divided into five parts, with some general questions in the beginning, more detailed in the middle, and open discussions towards the end.

The interviews consisted of the following five parts:

Part one of the interview was a series of personal questions to get background information such as the interviewees position at the company, years employed and area of expertise. This was made to ensure that the interviewee had the desired background and knowledge.

Part two was a series of short yes/no questions to get some basic understanding about the business domain, product characteristics, and how they manage the system and information today.

Part three was the main part which included more exhaustive questions about how data is managed and documented during the development. This section also included questions regarding how and if documentation is continuously updated when changes or corrections occur after release or during maintenance.

Part four covered the development process and the organization.

Part five consisted of a more open part with a chance for the interviewee to speak more freely about his/her own experiences and observed problems within the area.

2.1 Case-Study Validity

All of the studied companies are among the world-leaders within their respective domains which indicate a representative selection. Based on this and the fact that the findings are so conclusive among the companies, we believe that the study provides a representative overview of the common practice and can therefore be considered important. However, the purpose of this study is not to claim, based on this population, that the results are statistically confident or valid for all companies in these business segments.

2.2 Description of Companies

The studied companies have requested to be anonymous and are therefore described in this paper as COMP1-COMP5.

COMP1 is a producer of heavy vehicular systems. They have a production volume in the range of 50.000-70.000 units per year. Their system are resource-constrained, distributed and with both critical and non-critical parts. In their development they mainly use software components that are developed in-house. The information is distributed between ECUs via two redundant CAN [8] networks. The system is built on a software platform that is continually evolving.

COMP2 produces heavy vehicular systems in the range of 60.000-80.000 units per year and they base their systems on a software platform. Distribution of critical data is performed on three CAN networks with different criticality levels where the communication on the most critical bus is cyclic whereas the other two are event triggered.

COMP3 is another vehicular company with annual volumes in the range of 450.000-550.000 units. Their system can be considered highly safety-critical and resource-constrained. Furthermore, they use several different types of networks to distribute data. Most of the hardware and software are developed by subcontractors. Data is distributed using network protocols, such as CAN, LIN [15] and MOST [16].

COMP4 is a manufacturer of public transportation systems producing about 1000 units per year. Network communication is made on fieldbuses. They are now shifting to Ethernet communication with their own protocol layers in their latest platform. There are both periodic data and event triggered data on the bus. They have a small amount of software redundancy but are moving towards hardware redundancy. Almost all development is made in-house. Their systems are based on software platforms that are continuously refined during their 30 year product lifetime. Old products are during their lifetime updated with new software platforms.

COMP5 develops around 10.000 units of large stationary logic control systems that are less resource-constrained than the other systems in the study. Their systems are based on regular software development and where parts of the system are developed separately as components. Their systems are continuously changing and functionality is added throughout the life-time of the system. Network communication is quite limited and based on Ethernet [6]. They have developed their own standard for development based upon the waterfall model [20]. The ECUs in the system contain both critical and non-critical functionality. The system is built using a centralized configuration database where involved nodes collect information such as system parameters and store them locally before usage.

Company	Vehicular	Product Variance	Sales Volume	Hard real-time	Soft real-time	Resource-constrained	Component-based	Distributed	Platform-oriented
COMP1	Y	1	2	Y	Y	3	Y	Y	Y
COMP2	Y	5	3	Y	Y	4	N	Y	Y
COMP3	Y	4	5	Y	Y	5	Y	Y	Y
COMP4	Y	2	1	Y	Y	2	N	Y	Y
COMP5	N	2	1	Y	Y	1	Y	Y	N

Figure 1. Company description.
Range: Low=1 and High=5. Yes=Y and No=N

All five companies selected for this study have been active within research and development of distributed embedded real-time systems for many years. This also applies to the interviewees which all could be considered highly competent and have at least five years of company experience. The companies develop products that mainly incorporate both hard and soft real-time properties. The investigation concerns how their systems are developed and maintained throughout their life-cycle.

Figure 1 shows some of the main similarities and differences between the companies. As seen in the figure, four of the involved companies produce vehicular systems and one company, COMP5 develops stationary industrial systems. The column "Product Variance" indicates if a company has large variances between their products. For example at COMP2, less than two products delivered have the same configuration while almost all of COMP1 products are off-the-shelf. Annual sales volume has a range between 1000 delivered products to several hundreds of thousands. The products of all five companies have both soft and hard real-time properties. Furthermore, all of the companies develop resource-constrained systems but systems developed at COMP1-COMP3 are more resource-constrained than the others. The most resource-constrained product developer is in this case COMP3 with high volumes and limited amount of system resources. Finally, the column "Platform-oriented", indicates if the company develops a company-common software platform as a base used in several manufactured products but with different configurations.

3 Design-time Data Management

In this section we present some state-of-practice issues on how the interviewed companies perform their documentation and process at design-time followed by a number of use cases and scenarios.

3.1 State of Practice

In the following part we present how the individual companies perform their documentation and what kind tools and processes are used. The main focus is to provide a better understanding of how data is managed throughout development and maintenance. Since there is a lot of information about each company in this section, we have classified each of the companies with a few keywords for readability and overall understanding.

COMP1 uses Rubus Visual Studio [3], which is a development environment that is tightly integrated with the Rubus operating system. In this tool they have adequate documentation complying with the J1939 and J1587 standard for bus messages. Except from bus messages they only have sparse documentation on data types in the internals of the ECUs. For most of the documentation they are entirely dependent on the person responsible for a specific part of the system. According to the interviewee this has worked quite well previously when their old software platform was used and the projects where smaller. Now they are introducing a new, more advanced, platform and are experiencing a big increase in data flow and system complexity. Current practice, where a small group or a single person alone is responsible for this information, is not sufficient anymore.

Company classification: Dependent on individual developers.

COMP2 Internally within an ECU, documentation and mechanisms such as special control groups evaluating the work are not so extensive. It is more up to the developer to manage data. The documentation and high-level development of internal behavior is made in Enterprise Architect [23] and follows Rational Unified Process (RUP) [12] as their development process.

For network communication they recently moved from text-based specifications to a database built on Vectors CAN db-admin [24], with their own company specific communication layer. An integration group has control of the network documentation and is responsible for how the signals are used. This enables them to have control of the network and its contents. Also, once a month, a more detailed review that works as a filter for detecting errors is performed. The documentation regarding the network is continuously updated with new information but old data is never removed. A problem for them has been the growing amount of documentation with several hundred pages of text to describe small parts of the system.

They strictly follow a defined process for adding, removing or searching for data or data properties but have also worked out a "speedy" process if you need fast decisions. They also have a routine to once a year go through the system and check if all data on the bus is used and all code really executes.

Company classification: Network controlled by an integration group. Little control on internal ECU data management

COMP3 uses Rational Rose [10] both for documenting internal signals within the ECU and for external, public network signals. From Rational Rose, function, system and software descriptions are generated. All signals are then semi-automatically put in a signal database and also in spread-sheets. From the spread-sheets, a special appendix is generated with specifications on timing requirements, semantics signals etc. The appendix is open for viewing to all involved developers. They struggle with large amount of text, sometimes several thousand pages, needed for describing models etc. Most of the development, both hardware and software is made by subcontractors.

This company builds their systems on different software platforms. Each platform has a leader that has a lot to say about documentation. Except from deciding what should be added or removed in the system, they also look at the entire business case if a change is doable from a technical and economical point of view. If not, they have the power to abort the introduction of new functionality if deemed necessary.

The company's knowledge about signals is documented in a signal database on a global level but it is more up to the responsible person for each software component to have internal control of each ECU. This is, according to them, a known problem. For internal ECU changes, there is a standardized document revision on dedicated meetings. Nothing is released to a subcontractor until it is approved due to legal aspects.

They work according to a so called "superset" thinking in their software platform where they have excessive signals to support different versions of the system. A unique configuration file containing specific information for a specific system is then distributed to all nodes in the system to enable or disable desired functionality.

Company classification: Good global knowledge on signals. The platform leader controls functionality. Each software developer is responsible for how data is managed internally on the ECU.

COMP4 uses spread-sheets for both signals and the fieldbus. During development all staff in a project can search and update these spread-sheets until they do a freeze. A first freeze is done before the actual implementation but is changed if faults are discovered. After a freeze, only a special reference group can perform changes in the freezed version. It is a living process until the product is type approved at the customer. All developers can read and reserve signals during development. A company defined process is used to decide when freezes are supposed to be done.

Company classification: Reference group controlled. Uses freezed version and spread-sheets for signals.

COMP5 uses Serena Dimensions [22], an application life-cycle tool where documentation is done together with the code. They also use high-level drawing tools for component development with a specified system interface and c-code generation. Both code and documentation is versioned in Serena Dimensions. The main idea with their system is that data values can be changed in their central database even when the system is up and running. When a change is made in the configuration database and committed, all involved nodes are notified that there are new data in the database. ECUs that use this data, collect a local copy from the database for internal use. Which kind of data a person is able to change in the central database depends on which authorization level the user is assigned. The majority of the data communication is done internally on the ECU and not on the network.

Company classification: Central configuration database. Access rights controlled.

3.2 Use Cases and Scenarios

This section illustrates some of the important use cases that occur during development and maintenance. What are the main differences in how the involved companies handle adding, removing, and searching for data in their system?

Adding data to the system This is done differently in all companies. In COMP1, the responsible technician verifies the system architecture, then decides which node to use and how the data should be transported. This is then discussed with the developer in an effort to find flaws in the solution. After that there are no special routines for how this is done. It is up to the developer. This same action is handled completely differently in COMP2 where a developer has to write a function specification which is approved by the configuration manager. In the change process, applicable on modeling and signaling COMP3 uses a rudimentary web interface to ask for a change. A team then examines the change and physically synchronizes it to see if the change is technically justified. If it is a major change to the system, the business case is also evaluated. In COMP4, adding data is managed within the project but all signals should be added and approved before implementation. If a change is requested after the documentation is freezed, a reference group has to verify and approve the change. COMP5 uses a similar process. If the new data is approved by an authorized person it can be added and used.

Removal of data Even if companies have some routine for adding data to the system, routines on how to remove data is usually non-existent. This raises the question if it could be the case that there are signals in the system that are produced but not consumed.

As in the previous section, in COMP1 it is up to the system responsible. Rubus has no support for checking if a produced signal is used or not. In COMP2, COMP3, and COMP5 they do not remove anything at all. COMP2 does a consistency check against a spread-sheet once a year to see if all code in the system actually runs. If a signal on the bus is not to be used anymore, its CAN ID is defined as occupied and is never used again. This is made in order to minimize future mistakes. COMP3 has no technique to automatically do a mapping and see if data is not used and can be removed without affecting the system. It is considered too time consuming to do this mapping. Instead they keep the old data and calculate with a 15% overhead in the system. In an effort to minimize the need for removal of data, COMP4 does a consistency check in the beginning of each project and only include required signals. They also try to remove unnecessary signals during system updates but normally there are buffers for extra signals in a project. This is because they want to avoid changes in the system that can possibly have unknown consequences. If something is removed in COMP5, it is verified in system tests. However they do not really remove the data, instead they hide it so that it cannot be used in the future.

What seems to be unanimous for all of these companies is that removal of signals is problematic. Since there is no good support for this in the tools or routines, it is again up to the developer in COMP1 to take such a decision. In the other companies they either try to eliminate signals when starting a new project, use overhead in the system or do a consistency check and hide unused signals.

Searching and usage of data How can a developer or system architect know if a data is already produced or not? COMP1 has a developer responsible for this knowledge and if a signal is needed by another developer, he/she has to ask that person. They have no documentation regarding the contents of the nodes. The network however is better documented. In the other companies it is possible to search for signals in a spread-sheet, signal database or some type of development tool with more or less detailed information. COMP2 is very strict on signals on the bus and developers have to go through an integration group to require information, if a signal exists and can be used. They have less knowledge about the internals of an ECU, what exists and can be used. However a group of people review the system regularly to avoid errors. Both COMP3 and COMP4 uses a spread-sheet where all developers involved can search for a signal. In COMP3 you have to go through the platform group for usage approval. COMP4 does not have the same control mechanism for the usage of signals. If a signal is broadcasted on the bus it is open for usage, no additional decisions has to be made when using the signal. There is however only one that can write to any given signal. Except from COMP1, it is possible to search for a signal and use after approval by some kind of control group.

4 Observations and Problems Areas

In this section we have, based on the above use cases and scenarios, formulated four key observations and ten problem areas.

4.1 Key Observations

O1. *Impact of product variability on documentation.* All of the involved companies in this study have different approaches and a variation of techniques for preserving knowledge about their systems. These companies also produce products that vary more or less. It seems that there is a relationship between the quality of the documentation and the product variability. Figure 1 showed how variances differ between different companies. COMP1 manufactures off-the-shelf products. COMP2 and COMP3 both have large product variances to support usage in different environment settings or to suit various vehicular equipment alternatives. COMP4 and COMP5 have small variances. In COMP4 the variances mostly concern HMI settings.

With this information in mind, we can clearly see that this is reflected in their system documentation. COMP1 that produces off-the-shelf products has the least amount of documentation on their system. COMP2 and COMP3 has large variances and both have a more rigorous documentation process. One of the reasons for this could be that large product variances in COMP2-COMP3 are one of the reasons that have forced them to have a more developed preservation of system knowledge.

O2. *Inclusion of Excessive Signals.* All of the involved companies have excessive signals in the system as well as functionality that is turned on an off. COMP2 always has excessive signals included in the system to support several vehicle variations. Each system is then configured to suit the individual vehicle configuration. An example of this is to have signals that support both automatic and manual gearboxes. In this way they turn on and off required functionality to suit their needs. The reasons for having excessive signals in their systems vary. In most cases excessive signals are included, either to support product variations or because there is a desire to keep them in the system since a change can have unknown effects to the system.

One reason for having excessive signals and functions in the system is to minimize modifications to the system. If proper tools and documentation techniques were available, it would be possible to build the system more optimized, without unused signals and functionality to save system resources and reduce cost.

O3. *Prioritization of selected parts of the system.* As a result of ineffective and inadequate tools for documentation, parts of systems are prioritized. Although COMP3 uses several different techniques to manage and document their system, it is a known problem that they prioritize more critical parts of the system as engine control, compared to the more soft infotainment functionality which is lagging behind.

O4. *Awareness that common practice is not enough.*

To get a flavor of how companies and interviewees consider their documentation and development process they where asked to rank themselves and how they compare to their competitors at the end of each interview.

When ranking themselves on a scale from one to ten where one is the lowest, a majority of the companies ranked themselves below average. One company ranked itself high with the motivation that as long as they don't have to extend their system with new signals and interfaces, current practice is sufficient. This indicates that it is hard to expand, change or add new functionality to their system, which could be a direct result of poor system documentation. The fact that most companies rank themselves below average regarding their documentation and development process indicates that there is much to be done within this area.

When they ranked themselves compared to their competitors, the ranking follows the same pattern with a below average score. This is interesting since these companies use a variation of documentation, from person dependent to extensive signal databases and processes to handle signals, although mostly for distributed signals.

In order to successfully manage these advanced systems, new techniques for how to handle data has to be introduced. As stated earlier one single person having extensive knowledge about the internals of an ECU is not ideal and could be considered as a possible single point of failure.

The overall statement here is that this is how documentation is believed to be handled within their application area. A question that arises here is why companies that produce highly safety-critical applications in their own opinion have below average control of their system, documentation and process.

4.2 Identified Problem Areas

There are several important aspects to consider regarding how these companies treat and documents data internally on ECUs or on the communication network. We have from the above use cases and scenarios identified ten problems, divided into three areas:

Documentation volume and structure

P1. *Growing information volume.* A major problem that was repeatedly raised during the interviews was the growing volume of information [9]. As an example, model descriptions are today made in different tools and sometimes in plain text. This is a major problem since there sometimes can be several thousand pages of text. In most cases everything is backward compatible and nothing is ever removed. This continuously adds to the complexity of the documentation and the amount of text. It is not efficient to supply a system-responsible person with several hundred of pages of information with some small changes. This seems to be a neglected problem that is becoming an overwhelming issue for developers and system architects.

P2. *Obsolete documentation.* Documentation is perceived as hard to maintain, requiring a lot of effort and time. As a direct consequence of this, correct and up-to-date documentation is lagging behind. One individual person or a group of persons can be responsible for updating reported changes in documentation. However it is hard to do this in parallel with development and this often introduces a delay until the change is reflected in the documentation.

If a company has documentation, it is versioned and there is also some kind of template specifying the how this should be done. However in all cases, how this is done in practice is highly dependent on the individual person managing the documentation. This has in one company lead to a special template used as a simple speedy possibility to go around their own rules. One way companies do this is to let everybody change according to their needs and freeze a version of the documentation regularly. COMP3 does not have this problem since a developer has to request a change beforehand.

P3. *Stale data.* Poor preservation of knowledge and inadequate documentation techniques often lead to stale signals in systems that the companies are or are not aware of. An issue with this is that these stale data items, except from adding to memory, bandwidth and CPU usage, may cause failures or unwanted system behavior. Unknown effects such as these are addressed in new, more stringent regulations such as IEC61508.

P4. *Inadequate ECU data documentation.* One thing correspond for all of the involved companies. There is a difference in how they treat data and signals on the network compared to internal data on ECUs. The network is documented using various tools and techniques whereas internal ECU data in most cases are not. The lack of efficient tools and techniques have made individual developers responsible for much of the knowledge about data items and functionality inside an ECU.

P5. *Dependency on individual developers.* Internal knowledge of an ECU is in several of the involved companies left to a single individual or a group of developers. This is an important issue since companies could lose valuable information due to poor, or non-existent, documentation. As an example, an individual developer in COMP1 can have all information about a certain part of the system or functionality. When other developers need a signal or information regarding that system or function, they have to ask the developer for it. When asked how this would influence the company if a staff member would leave the company, they say that it would not be a disaster but it would mean a lot of effort for someone else to get up to date.

The systems that COMP1 are developing have so far been quite small since large parts of the product have been mechanically controlled. The current trend is to introduce more and more computer-controlled parts, thus rapidly increasing the system complexity. The small size and amount of data in the system made it possible for persons to keep track of most things. This worked up until now. New platforms are being released with more computer controlled systems that are too complex for a single developer to handle. The new systems are redundant, safety-critical, contain more diagnostics, more signals, human-machine interface (HMI), and other functionalities.

Tool support

P6. *Lack of efficient tool support.* More efficient documentation, tools and processes are needed and could in the end reduce development costs. Companies themselves indicate that within a few years they will need to use a small set of tools or one single flexible tool to limit the amount of text describing models today. Since systems and functions require a lot of effort and are costly to develop, companies reuse as much of the system as possible. This puts high demands on documentation in order for developers to be able to understand how a function will work if it is reused in another setting with other dependencies. This is especially true if it is a safety-critical function which often is rigorously verified and tested.

P7. *Lack of visualization.* As systems are getting more heterogeneous and more complex, in the sense of more signals, increasing number of ECUs and more distributed data items, developers have raised the question of a need for a graphical view of the whole development chain to aid developers and system architects.

Important aspects to visualize are;

- how functions are connected
- how data is shared between functions
- how ECUs are connected
- where the nodes are physically placed

Routines

P8. *Poor support for adding data.* Routines for adding data to the system differ a lot between the companies. A problem here is that there is a lot of manual work done by individual developers, or just open discussions to verify how additional data affects the system and there is no effective tool support for this matter.

P9. *Difficult to search for data.* As long as a data item is distributed on the network, it is in most cases possible to search for a data item. However the possibility to search for an internal ECU data item is in most cases limited.

P10. *No support for removal of data.* Despite the fact that some of these systems are resource-constrained and available resources are sparse, a lot of unnecessary data items remain unused in the system. In an effort to reduce the number of unused data items, some of the companies try to remove old data when starting a new project but they are careful about doing so because they lack knowledge about system dependencies such as, who are producing and who are consuming this data. Instead they either try to hide data, leave it as it is, or mark them as occupied so there will be no new users. It seems that the overall problem here is a lack of feedback from the development tools. There is no way to automatically see dependencies for internal data.

5 Remedies and Vision for Future Directions

In this section we elaborate, based on the problems (P1-P10), observations (O1-O4) from the study, and future standards and regulations, on possible improvements in data management tools and processes for embedded real-time system's development.

To improve data management we propose to lift data to a higher level during development. A more data centric development is needed, where data is considered early in the development phase and seen as its own entity. To substantially elevate existing data management and documentation towards a more data centric development, we propose six remedies;

R1. *A unified development environment.* To successfully be able to manage the problems stated in P1, P2, P5 and P6, scattered information needs to be gathered in one development environment. As seen in the study, some companies successfully use a signal database for bus messages. By extending this to also include internal signal and state data, an integrated data management environment supporting the entire development chain including requirements, modeling, design, implementation, and testing is achieved. This data management environment could aid developers by filtering out only the relevant documentation for each development activity. Correctly implemented this environment should provide an easy interface for developers from different sub-systems can share to update and manage documentation.

R2. *Global data warehousing and data-flow graphs.* Data warehousing is an effective technique, providing means to store, analyze and retrieve data. By introducing global data warehousing, and data-flow graphs to the development environment a company-common documentation base that development projects of different sub-system can access and share is provided. It also gives developers the possibility to identify and visualize data providers and subscribers and thereby aiding designers when adding, managing and removing data. This gives developers the means to solve problems identified as P3, P4, P8-P10.

R3. *Automated tools and techniques.* To additionally aid developers solving P2, P4-P5 and maximize the impact of a unified development environment, automated tools and techniques must be introduced to link design-time documentation against run-time mechanisms.

R4. *Physical visualization.* By introducing physical visualization, showing the physical layout and data streams of the system, identified as P7, we solve a problem that was explicitly pointed out by some interviewees in the study.

R5. *Meta-data information.* To aid in solving P2 and P6, a natural coupling between system requirements and data properties meta-data information such as resolution, real-time properties, priorities, criticality, etc. needs to be included into the development environment.

R6. *Integrated data modeling tool.* During our previous case-study [19] it became obvious that using internal data structures for internal data storage lead to difficulties to keep track of data and to perform memory optimization. A integrated data modeling tool can provide developers with means to organize and structure all system data, thereby aiding in solving P5. Within the database community several data modeling techniques, such as entity-relationship modeling,[5] exist.

Introducing these remedies and forming a uniform development environment give developers the prerequisite needed for effectively managing their system development and maintenance. Figure 2 illustrates how the problems are linked with the proposed remedies.

	Remedies					
Problems	R1	R2	R3	R4	R5	R6
P1	X					
P2	X	X	X		X	
P3		X				
P4		X	X			
P5	X		X			X
P6	X				X	
P7				X		
P8		X				
P9		X				
P10		X				

Figure 2. Problem areas with associated remedy or remedies.

6 Conclusions

In this paper, we show that due to the increasing system complexity, current state of practice in data management is not adequate. There are many important issues observed in this case-study. From these, we have identified ten problem areas and formulated four key observations, based on current practice and future needs. These problem areas and observations set the path for future research and improvement.

It is confirmed by all involved companies that new processes and techniques for achieving a satisfactory documentation on a software system are needed to be able to handle the needs of today and tomorrow. This is something that could be required to meet the upcoming safety regulations, eg. as specified by IEC 61508, and will be a complex and difficult transition for these companies.

The study shows that there is much to be done within the area, especially documentation of data internally on ECUs. Inefficient, or lack of, routines for adding, removing or searching for data or data properties has in some cases made companies completely dependent on individual experts instead of thorough documentation. As the systems grow, this approach is no longer feasible.

Another more unwanted effect of inadequate data management is that there are also data included which no one knows exists. These stale signals is an important safety issue since they could have unknown consequences to the system. An important fact is that these systems are in many cases resource-constrained and stale data waste resources. This could be a major cost factor for mass producing companies with high demands of cost-efficiency.

Currently, adequate tools to manage distributed data exist, resulting in a much better data management for distributed data compared to internal ECU data. In this paper, suggestions for improved tool-support for internal data, as well as overall system data management is presented. It is our belief that a novel tool that incorporate adequate data documentation, management and design views, both for design and run-time would significantly improve current data management practices.

7 Future Work

From this case-study we could also see an emerging need for more flexible and efficient run-time data management. Several interviewees indicated that there is an increasing need to manage both hard and soft real-time requirements within their systems. There are also indications that a more secure handling of data is needed since there is a desire to connect to the system at run-time for maintenance, upgrades and infotainment purposes. This issue is seems especially important when using telematics to access these safety critical systems. Another issue is the coming standards and regulation which will put higher demand on data management. These are some of the important issues still to investigate based on the outcome from this case-study.

References

[1] A. Albert. Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems. pages 235–252, 2004.

[2] S. F. Andler, J. Hansson, J. Eriksson, J. Mellin, M. Berndtsson, and B. Eftring. DeeDS Towards a Distributed and Active Real-Time Database System. *ACM SIGMOD Record*, 25(1):38–40, 1996.

[3] Arcticus Systems. http://www.arcticus.se.

[4] L. Casparsson, A. Rajnak, K. Tindell, and P. Malmberg. Volcano - a Revolution in On-Board Communications. Technical report, Volvo Technology Report, 1998.

[5] P. P.-S. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1), 1976.

[6] D. Dolezilek. Iec 61850: What you need to know about functionality and practical implementation. *Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources*, March 2006.

[7] L. Gabriel and H. Donal. Expanding Automotive Electronic Systems. *Computer*, 35(1):88–93, Jan 2002.

[8] R. B. GmbH. *CAN Specification*. Bosch, Postfach 30 02 40 Stuttgart, version 2.0 edition, 1991.

[9] K. Hänninen, J. Mäki-Turja, and M. Nolin. Present and Future Requirements in Developing Industrial Embedded Real-Time Systems - Interviews with Designers in the Vehicle Domain. In *13th Annual IEEE Int. Conf, Engineering of Computer Based Systems (ECBS)*, Germany, 2006.

[10] IBM Rational software. New York, USA. http://www-306.ibm.com/software/rational.

[11] International Electrotechnical Commission IEC. Standard: IEC61508, Functional Safety of Electrical/Electronic Programmable Safety Related Systems. Technical report.

[12] A. S. Jochen Krebs. *IBM Rational Unified Process Reference and Certification Guide : Solutions Designer (RUP)*. IBM Press, December 2007.

[13] R. K.Yin. *Case Study Research Design and Methods*. Sage Publications, Inc, third edition edition, 2003.

[14] J. Lindstrom, T. Niklander, P. Porkka, and K. Raatikainen. A Distributed Real-Time Main-Memory Database for Telecommunication. In *Proceedings of the Workshop on Databases in Telecommunications*. Springer, 1999.

[15] Local Interconnect Network. http://www.lin-subbus.org.

[16] Media Oriented Systems Transport. http://www.mostcooperation.com/home/index.html.

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

[18] D. Nyström, A. Tešanović, M. Nolin, C. Norström, and J. Hansson. COMET: A Component-Based Real-Time Database for Automotive Systems. In *Proceedings of the Workshop on Software Engineering for Automotive Systems*, pages 1–8. The IEE, June 2004.

[19] D. Nyström, A. Tešanović, C. Norström, J. Hansson, and N.-E. Bånkestad. Data Management Issues in Vehicle Control Systems: a Case Study. In *Proceedings of the 14th Euromicro Conference on Real-Time Systems*, pages 249–256. IEEE Computer Society, June 2002.

[20] W. Royce. Managing the development of large software systems: concepts and techniques. In *ICSE: Proceedings of the 9th international conference on Software Engineering*, pages 328–338. IEEE Computer Society Press, 1987.

[21] C. B. Seaman. Qualitative methods in empirical studies of software engineering. *Software Engineering*, 25(4):557–572, 1999.

[22] Serena Dimensions. http://www.serena.com/products/.

[23] Sparx Systems Ltd. http://www.sparxsystems.eu/.

[24] Vector Informatics, CANdb Admin. http://www.vector-worldwide.com.