Present and Future Requirements in Developing Industrial Embedded Real-Time Systems - Interviews with Designers in the Vehicle Domain

Kaj Hänninen
[1]*Mälardalen Research and Technology Centre (MRTC)*
[2]*Arcticus Systems*
Sweden
kaj.hanninen@mdh.se

Jukka Mäki-Turja
Mälardalen Research and Technology Centre (MRTC)
Sweden
jukka.maki-turja@mdh.se

Mikael Nolin
Mälardalen Research and Technology Centre (MRTC)
Sweden
mikael.nolin@mdh.se

Abstract

In this paper, we aim at capturing the industrial viewpoint of todays and future requirements in development of embedded real-time systems. We do this by interviewing ten senior designers at four Swedish companies, developing embedded applications in the vehicle domain.

This study shows that reliability and safety are the main properties in focus during development. It also shows that the amount of functionality has been increasing in the examined systems. Still the present requirements are fulfilled using considerably homogenous development methods.

The study also shows that, in the future, there will be even stronger requirements on dependability and control performance at the same time as requirements on more softer and resource demanding functionality will continue to increase. Consequently, the complexity will increase, and with diverging requirements, more heterogeneous development methods are called for to fulfil all application specific requirements.

1. Introduction

There is an increasing trend towards software solutions in embedded systems. Replacing mechanical functionality with computer-controlled solutions gives opportunities for more advanced and more flexible functionality, e.g., anti-lock braking, traction control etc.

Over the years, a large number of publications, e.g., [5][6][8][10][11][12][13][18] has addressed design issues, embedded application trends or requirements in development of industrial embedded systems. Möller *et al* [10] present the industrial requirements, both technical as well as process related requirements, on component technologies in the heavy vehicle domain. Åkerholm *et al* [18] presents an investigation concerning classification of quality attributes for component technologies in the vehicle industry. The investigation show that dependability characteristics (safety, reliability and predictability) are considered as the most important ones. Koopman [8] presents attributes of four different types of embedded systems (signal processing systems, mission critical and distributed control systems and consumer electronic systems). Koopman addresses requirements, life-cycle support and business models in development of embedded systems. Graaf *et al* [5] presents an industrial inventory of seven companies developing embedded software products. Their inventory of state of practice addresses requirements engineering and architectural issues such as design and analysis. The inventory covers companies from many different domains, e.g., developers of mobile phones and consumer electronics, distributed data management solutions etc.

In this paper, we investigate the industrial requirements in the vehicle domain, especially requirements related to real-time issues on a high overall level, such as safety and reliability requirements of embedded application/products, as well as on a lower technical level, such as choice of operating system (OS) and execution models. The study was performed as a series of interviews with ten senior designers at four Swedish companies.

Specifically, we address the following questions:
Q1. What characterise the embedded applications?
Q2. What are the designers concern on application properties such as safety, maintainability, testability, reliability, portability and reusability?

Q3. How are the applications verified/analysed?
Q4. What are the considerations in choosing an OS, and execution model?
Q5. What resources are considered as constrained in the systems, and to what degree?
Q6. What kind of tool support is needed in the development of future systems?
Q7. What are the designers experiences of software components, i.e., component based development?

The aim of this work is foremost to explore and describe the current and future industrial requirements as perceived by the senior designers.

The paper is organised as follows. In section 2, we describe the framework used in the study of the requirements. In section 3, we describe the results of the conducted interviews. In section 4, we address the main questions of the study and discuss our observations of the interviews. In section 5, we conclude our investigation. The paper ends, in section 6, with a discussion concerning verification of the presented results.

2. Investigation setup

For this study, we adopted the investigation framework described by Robson [15]. According to the framework, both the purpose of a study and the theory guiding the study should form as guidance when developing the actual research questions. The substance and the form of the research questions form the basis when deciding on suitable investigation method and sampling strategy.

Purpose: The main objective of this study was to investigate the typical set of industrial requirements in development within the embedded control community in the vehicle domain. The results are expected to form a foundation for further research on tool support, design, analysis and synthesis of embedded real-time systems using multiple execution models.

Theories: Our experience is that there has been little effort done to encapsulate novel theories by supporting development tools and techniques in the industrial domain. Traditionally, the development of these systems tends to focus on the safety critical parts, which constitutes a small fraction of the total system functionality. Homogenous development methods are often used for both the safety-critical and the non-critical functionality in the systems. This results in unnecessary complex designs and over utilised systems, where valuable resources such as processing time and memory resources are wasted. We believe that there is a need for more sophisticated development support compromising of additional tool support and more domain and application specific development platforms, resulting in a more heterogeneous development environment and resource efficient run-time structure. The development platform should aim at handling complexity by relieving the developer of too low details while preserving predictability for core functionality as well as flexibility for less critical functionality in the run-time structure.

Questions: We compromised upon a set of quantitative and qualitative, closed and open-ended questions. The main purpose of the quantitative questions was to facilitate analysis of importance among application properties.

Data collection: Due to the substance and the form of the research questions, the study was conducted as face-to-face interviews using a questionnaire.

A pilot study was performed at an OS and development tool vendor. The purpose of the pilot study was to refine the data collection plans and to evaluate the feasibility of the chosen data collection method. The structure of the questionnaire was refined and additional questions were added, as a result of the pilot study.

Sampling: In this study, we use purposive non-probability samples, i.e., the samples are selected as to interest (we do not make generalisation to any population beyond the samples). Four successful and renowned companies in the Swedish vehicle domain were participating in the study. The samples represent both subcontractors as well as own equipment manufacturers. Moreover, the selection is a representative subset of both off-road and road vehicles. The companies range from small and medium-sized enterprises to large corporate groups. The thorough examination of the applications and development processes require us for secrecy reasons to refer the companies as A, B, C and D.

Ten software designers with several years of experience from development of control systems for embedded real-time systems participated in the study. For preparation reasons, the questionnaire was mailed in advance to each interviewee.

Analysis: Upon agreement with the interviewees, each interview was tape-recorded. The recordings and notes taken during the interviews were interpreted and analysed both individually and at group basis. We did however not use any specific software package to interpret or analyse the collected data.

Biases: Several factors may introduce unwanted biases in a real-world study. For example, recording interviews may affect the respondent, welcoming or sharing the respondents' views may affect the interview, and so on. To avoid or at least minimise possible biases, we followed recommendations given in [14][15][17] about how to construct questionnaires

and conduct face-to-face interviews in real-world situations.

It is our experience that the research questions were easily understood and similarly interpreted by the interviewees, and that the recording had no or very little effect on the respondents and the outcome of the interviews.

3. Investigation results

In the following section, we describe: (i) Real-time and functional characteristics of the examined applications. (ii) The interviewees concern on selected application properties. (iii) The currently used resource management policies and available execution models of the examined applications. (iv) The actual resource situation in the examined systems i.e., availability of computing resources such as CPU time and memory. (v) Some desired support in development tools, as expressed by the interviewees.

3.1. Application characteristics

The product volumes of the investigated applications are typically less than 1000 products per year. The applications are mainly used as control applications for various types of vehicles. In addition to control functionality, the applications typically contain functionality for information handling such as logging for diagnostic purposes and presentation of data i.e., visual interaction with the system operators.

The architectures of the examined systems are of distributed character where several nodes, Electronic Control Units (ECUs), perform computations and communicate with each other mainly via CAN buses. Each ECU is usually dedicated to handle specific type of functionality e.g., an engine controller is mainly responsible for controlling engine specific functionality such as fuel injection, ignition etc. The number of ECUs in the systems has typically been increasing over the years. For example, Table 1 shows the amount of software and the number of control units in evolution of a single product at one of the investigated companies.

Table 1. An example of the amount of software and the number of ECUs in a single vehicle, at company A.

Year	**1991**	**1997**	**2002**
Lines of code	20,000	55,000	140,000
Files (.c, .h)	50	400	700
ECUs	1	2	3

Current characteristics: The examined applications are realized by hard and soft real-time tasks. In several systems, hard real-time tasks are used to model the majority of all functionality. In extreme cases, as much as 95% of the functionality is modelled by hard real-time tasks. In addition, functionality with requirements that are neither hard nor soft, but somewhere in-between, is often modelled as hard. In context to this, the designers stress that development of hard application tasks is considered as more controllable and simpler than development of soft application tasks. In addition, several interviewees consider time-triggered systems to be the most convenient way to model hard real-time functionality.

Typical technical requirements in the examined applications include; jitter requirements and precedence relations among tasks. The timing constraints, e.g., deadlines on different functionality, can vary as much as three orders of magnitude in a single application, typically from milliseconds to several seconds.

The amount of safety critical functionality varies in the investigated applications. In all of the examined applications, the control functionality is considered as being most safety critical and developed mainly using the time-triggered paradigm.

Several interviewees consider their systems being I/O intensive. In some systems, as much as 30% of the available processing time and hundreds of I/O pins is used to handle I/O functionality. The I/O functionality is realised by both time and event-triggered execution models. However, it is most commonly realised using the time-triggered model, i.e., through polling.

The information intensity in the investigated applications varies. In some applications, the information originates from logging and diagnostics of the systems operational conditions, whereas other applications receive and process external information that is presented to the users during operation.

Future characteristics: The interviewees believe that the information intensity and number of control functionality will increase in the future. They state that in the future both legislation and insurance reasons will force development of more sophisticated control algorithms and require an increasing amount of information to be saved for diagnostic reasons. In addition, designers from one company predicts that legislations, especially non-pollution laws, and future trends in development of vehicle engines will require better control precision. This will result in an increased transformation from open to closed loop controlling. Furthermore, some interviewees predict that functionality interacting with the environment will be developed using fewer sensors in the future and that certain conditions/states of the environment will be derived using the remaining set of sensors. Classification of functionality in Safety Integrity

Levels (SIL) [7] is also believed to be an important activity in the future.

3.2. Functional application properties

In this section, we present the interviewees concern on the following application properties: safety, maintainability, testability, reliability, portability and reusability.

Safety: Safety is considered as a derived property originating foremost from analysis and testing. In some of the examined systems, redundancy and certain safety properties are solved outside the actual software implementation, by physical cabling etc. The software in these systems can be overridden by mechanics in case the software malfunctions and a safety critical situation occurs.

Maintainability: Some interviewees state that the developers consider and try to facilitate future maintainability of applications. Some interviewees also state that they have very strong requirements (economical and quality) on applications being error free since withdrawing an erroneous application would be very costly due to the product volumes. There seems to be an agreement on that maintainability will have to be considered as a more important property in the future, specifically in the context of upgradeability. The lifespan of the examined systems can be several decades and customers put demand on new features and require hardware replacement parts to be available during the entire lifespan of a system. This requires applications to be well structured and easy to understand for future developers (maintainers).

Testability: Testability is stated as an important and necessary property to achieve reliability and safety. Today testing is the main technique to verify functional requirements.

Reliability: Several interviewees state that a company's reputation is very much dependent on the reliability of the delivered systems; i.e., it is considered as being of utmost importance to develop systems that actually are, and perceived by customers as, reliable. Failure in producing reliable systems is often stated to origin from erroneous requirement specifications, i.e., not from the implementation itself.

Portability: Some interviewees do not consider portability during development, simply because they seldom change hardware or OSs. Other respondents claim that portability is an increasing concern and that it is mainly facilitated by separation of hardware and software dependent functionality.

Reusability: Reusability of both soft- and hardware is an ongoing activity in all of the examined systems. However, the amount of reusable software varies in the examined systems. Some interviewees' state that reusability of architectures is not achieved until they have undergone several modifications, hence it may takes years before certain parts of architectures are actually reusable. To facilitate reusability among different systems, some of the companies have developed common software platforms. The platforms contain all common functionality and have standardised interfaces. General software components are also mentioned as reusable entities. The components are general in the sense that they are, to a large degree, application independent.

Additional properties: When asked for additional properties that are considered as important for their applications, the interviewees mentioned *robustness*, *scalability* and *usability*. Robustness is defined by the respondents as 'the absence of unexpected behaviour' or as 'an additional degree of reliability'. Scalability is considered in the context of development as the ability to scale systems using the available development tools. Usability of architectures is mentioned as a process related issue. In that context, the usability of architectures is said to be dependent on whether it facilitates understanding and communication between developers. All of the respondents stress the importance of architectural descriptions as means of communication between people i.e., not only as logical or structural system description.

3.3. Temporal application properties

This section describes the interviewees view on the temporal analysability of the applications and verification of functional/temporal behaviour. It also addresses the verification of resource utilisation in the examined applications.

Analysability and verification: Analysis of real-time properties such as response-times, jitter, and precedence relations, are commonly performed in development of the examined applications. In this context, some interviewees stress the desire of better analysis support in development tools and state that analysing a whole system with respect to temporal and spatial attributes is very difficult, sometimes even intractable. Due to the difficulties in analysing a complete system, and for upgradeability reasons, some of the examined systems are intentionally over-dimensioned with respect to processing power and memory resources.

The emphasis on verification is foremost on the functional behaviour. Our experience is that the temporal attributes are not serving as direct guiding factors (albeit they are more or less considered) during development.

Functional behaviour: All of the respondents had a unanimous opinion that analysis and verification of the

functional behaviour was the most important activity in the verification and analysis processes (more important than analysis and verification of temporal behaviour). The functional behaviour is mainly verified by manual and automatic module and systems tests. Failure mode and effect analysis (FMEA) are commonly performed both during development and on complete systems. Several interviewees state that source code inspection is performed among the developers and that it serves as analysis/verification of functional behaviour.

Temporal behaviour: The verification of temporal behaviour was said to have lower importance than of functional behaviour. The temporal verification of the examined systems commonly involves verification of precedence relations among functions and verifying that deadlines are met i.e., that estimated worst-case execution times holds and that calculated worst case response-times are met.

Verification of resource utilisation: Many of the examined systems have been evolving for several years. The amount of resources, e.g., the number of control units, has been increasing over the years. Currently, all of the examined systems have more than enough processing time and available memory to perform the intended computations. Hence, verification of resource utilisation, such as memory consumption, is considered of lower importance. However, some interviewees desire possibilities to analyse memory consumption, mainly to be used when the available resources are running low, i.e., before additional resources (ECUs) have to be added to the system.

3.4. Operating systems

In this section, we describe the issues involved in choosing operating system and the execution models used in the examined applications. We describe the main motivations to why these operating systems were chosen and the interviewees expressed experience of the used execution models.

When investigating the type of technical considerations that has bearing on the choice of OS for the embedded applications, we discovered several non-technical considerations that are strong motivators to the choice of a specific OS, e.g., requirements on coordination to use a common OS at different departments of a company. These requirements do not directly reflect the technical need in development. The technical requirements are commonly considered later on. However, the requirements on simplicity, i.e., ease of use, is a motivator both when choosing OS and among available execution models

The commercial operating systems that are used, or have been used, in the embedded applications by the investigated companies are Rubus [1], VxWorks [16], OSE [4], O'Tool [1], RTX [2] and WinCE [9]. In addition, one of the investigated companies develops their own operating systems, used in a majority of their applications. The main motivation for this is that their own operating systems are claimed to be simpler, more robust and have less run-time footprint (timing and memory overhead) than the commercial OSes.

The interviewees' state that the main considerations when choosing a commercial operating system include:

- Cost (royalties, licenses).
- Availability of supported development tools related to the OS .
- The supported execution models in the OS, i.e., its suitability for the application domain.
- Coordination within a corporate group or subsidiaries to use a common OS.
- The availability of fast and skilful technical support.
- Recommendations originating from other companies evaluating the OS.
- The popularity of the OS, i.e., to what extent is the OS used by other companies.
- The OSs internal timing and memory overhead.
- Safety classification issues.

3.5. Execution models

Both time- and event-triggered execution models are used in all of the examined applications. The time-triggered model is commonly used for control functionality whereas the even-triggered model is used mainly for information handling for diagnostic reasons. The interviewees state that the choice of execution model in development is mainly dependent on: (i) Verification possibilities, both functional and temporal. (ii) Flexibility of adding new functionality. (iii) Required response-time on functionality. (iv) Simplicity of use in development.

3.6. Resource limitations

This section describes the current resource situation in the examined systems, as expressed by the interviewees. We investigated whether and to what degree, the amount of processing time, RAM, ROM and communication bandwidth, were considered constrained in the systems.

As described in section 3.3, many of the examined systems are intentionally over-dimensioned; hence, the interviewees did not consider any of the resources as being particularly constrained during software development. However, in case the systems would run out of resources, the interviewees' state that they would most probably consider installing additional hardware resources rather than redesigning the way the applications utilises the resources. This is however,

said to be dependent on the urgency of system delivery. In extreme cases, functionality has been removed from the examined systems, when the available resources have been fully utilised.

3.7. Desired tool support

In this section, we present the interviewees expressed desire concerning support in development tools and their experiences of software components, i.e., component-based development [3].

The expressed wishes, concerning support in development tools, amplify the requirements on verification, safety and reliability aspects. The concise picture seems to be requirements on simulation and verification possibilities of applications on PCs. Moreover, an integrated possibility for model-based development with Matlab and Simulink together with automated code generation is another common desire expressed by the interviewees.

The following is a list of desired tool support, as expressed by the interviewees. The desired support addresses both technical and process related issues. The interviewees would like to see:

- Simulation of the embedded applications on PCs.
- Replacing of text based user interfaces with graphical user interfaces.
- Support for model based development with possibilities to exchange information between tools from different vendors.
- Abstractions of graphical models i.e., visualisation of architectures at different levels and from different views.
- Automatic code generation e.g., from models to source code.
- Support for formal verification of source code.
- Support for execution time analysis.
- Possibilities to identify or trace the requirement specifications from the source code, and vice versa.

The current support in development tools varies at the companies. For example, one company has extensive support for simulation of embedded applications on a PC, whereas others do not have simulation possibilities at all. However, none of the examined companies has all of the listed support in their development tools.

3.8. Software components

Only one of the examined companies explicitly state that they use software components in the development of their applications. The company uses both in-house as well as third party developed components. The reasons to why the other investigated companies do not use software components are related to facts such as difficulties in understanding the concept of component-based development. Furthermore, issues such as modifiability of functionality are stated as a restricting factor for use of software components.

However, all of the interviewees' state that the abstraction possibilities that components provide, is one of the main motivators of component based development, simply because it facilitates understanding and communication between developers.

4. Discussion - our observations

In this section, we address the main questions of the study and present our own observations and conclusions of the interviews.

Q1. What characterise the embedded applications?

The fact that more and more mechanical solutions are replaced with software, results in an increasing complexity both in size and in diversity. The applications are evolving and contain more heterogeneous functionality that before. In the future, this requires abilities to cope with (i) increasing data handling and (ii) increasing complexity in control functionality. It is common that applications contain a mix of hard and soft real-time tasks. We observed that a surprisingly small fraction (e.g., ~25% at company A) of the requirements reflects need of hard real-time tasks. Still, the use of hard real-time tasks is very high (~75% at company A). We believe that the high utilisation of hard tasks is mainly related to three reasons:

(i) simplicity in development
(ii) for verifications/reproducibility reasons
(iii) tradition in development.

The simplicity in development originates from years of evolving support in development tools that to large extents is intended for development of safety critical real-time systems. There is also a tradition in using hard real-time tasks for the majority of functionality, simply because developers tend to rely on designs from previous projects, instead of scrutinizing and considering the designs appropriateness for the diverging type of functionality found in today's and in future applications.

Hence, the predicted increase in information intensity and diversity of functionality, require use of more suitable development models, i.e., models for diverging strategies that can handle both safety critical functionality as well as more flexible and resource efficient functionality in the same system.

Q2. What are the designers concern on application properties such as safety, maintainability, testability, reliability, portability and reusability?

The future classification of functionality in Safety Integrity Levels (SIL) implies that reliability, safety, analysability and testability will continue to be very important application properties in the future. Moreover, we believe that facilitating maintainability of the applications will be a more important activity to consider due to the increasing complexity, long product life cycles and demand on upgradeability of the applications. However, moving into the area of more maintainable systems, through, e.g., raising the level of abstraction and introducing reusable frameworks, introduces challenges since it must be done without compromising the systems safety or reliability.

Q3. How are the applications verified/analysed?

Functional behaviour is typically verified through testing on the target platform, whereas properties such as temporal behaviour are mainly verified with support of software analysis tools. Worst-case execution times are commonly estimated during development, and later on, verified through measurements on the target platform.

The interviewees desire tools for verification of both functional and temporal behaviour of embedded applications on PCs. We believe that for the large fraction of future functionality, predictable and flexible execution models, where combinations of different analysis techniques that focus more on average case behaviour and quality of service rather than on worst-case behaviour, will be significant.

Q4. What are the considerations in choosing an OS, and execution model?

Politics and non-technical aspects are strong motivators in choosing OS. It is obvious that such issues could motivate the use of an OS that is more or less suitable to fulfil the technical issues in an application domain or specific needs within a corporate group. We believe that the increasing complexity in the examined application domain require more focus on technical issues, such as availability of novel tool support related to the OS and possibility to utilise more suitable execution models in the OS. For example, with increasing demand on safety classification such as SIL, the OS must be able to support the trade off between technical aspects such as verifiability and efficiency. For example, the small core of safety critical functionality should be allowed to use more resources if it must fulfil the SIL classification and be verifiable (testable and analysable), whereas the rest of the functionality (non-safety-critical) should utilise more resource efficient run-time mechanism to implement the functionality.

Q5. What resources are constrained in the systems, and to what degree?

Our investigation revealed that the computational resources are not considered as constrained during software development. We believe there are two possible reasons for this.

(i) The investigated companies are already using resource efficient development methods (legacy methods), originating from times when all functionality was homogenously implemented.
(ii) The systems are over-dimensioned at the same time as the developers put most effort in implementing complex functionality without having tool support to analyse resource consumption, e.g., memory usage.

The increasing number of ECUs reveal that the computational resources are highly utilised from time to time, i.e., before addition of hardware. With an increase in diverging functionality, the current situation where a static schedule is used for the majority of all functionality, will either be intractable or overly resource demanding (ending up in new ECUs being added) in the future. Instead, the future development tools need to support an efficient and verifiable way to allocate resources, so that the developers either can:

(i) Continue their efficient way of developing with efficient tool support adapted to the diverging functionality in the application domain, or
(ii) Have novel tool support that allows them to begin developing systems using efficient and resource saving models.

Some interviewees experience that the quality of software increases when developers do not have to worry about resource consumption. Hence, future support for resource efficient development needs to be automated to as large extents as possible.

Q6. What kind of tool support is needed in the development of future systems?

The view on future requirements is that safety critical functionality needs to be certifiable and the emphasis on less critical functionality will be on more efficient resource usage (e.g., average resource utilisation rather than worst case utilisation). This

requires system integration tools with possibilities to take domain specific models that support efficient automatic code generation, reproducibility for the safety critical functionality and efficient resource usage for the rest. In addition, to cope with the increasing complexity developers need tools that lift the level of abstraction, i.e., tools that provide both different levels of abstraction as well as different views (e.g., temporal and functional) at each level of abstraction. It is imperative that the tools relieve some burden of developers (our study show that simplicity is a strong motivator in development) for example by letting synthesis tools provide details (such as assigning temporal attributes, priorities etc.) so that requirements are met.

Q7. What are the designers experiences of software components, i.e., component based development

There is an ongoing activity at one of the investigated companies concerning reusability of general type (application independent) software components. We believe that general components facilitate development and may increase the software quality since they are often adapted in several applications and being subject to extensive testing. However, to be resource efficient, or predictable for safety critical parts, these type of components need to be efficiently, and/or predictably, synthesised, i.e., become application specific in the run time system. Hence, the components should be general and execution model independent during development, and then mapped to an application specific run-time structure.

5. Conclusions

In this paper, we presented some requirements in development of industrial embedded systems in the vehicle domain. The requirements were collected by a number of interviews with ten senior designers at four companies in Sweden.

Many of the investigated applications are developed using methods that are adequate for the (relatively small) parts that are safety critical. Less critical parts are adapted to fit into the framework of the critical parts. With the increasing size and complexity of software, this homogenous way of developing applications will, we believe, be inadequate. In the future software development strategies, methods and tools must be able to capture the different diverse requirements of the applications and trends in the application domains. Ranging from a small core part of the application that is safety critical to a larger part of the system focused on, for example, quality of service and average case behaviour. The characteristics of the examined systems and the predicted increase in information intensity and higher precision on control functionality, would allow for more suitable execution models, i.e., resource saving and quality enhancing, to be introduced (one company even expressed their interest in execution models addressing variable quality of service levels).

A wide spectrum of different kind of tool support is desired in development of the applications. For example, tools for model-based development with simulation possibilities and automatic code generation are considered as highly desirable. Furthermore, the use of software components and CBSE in general, provides possibilities for architectural descriptions at a high level. The importance of architectural descriptions as means of communication between developers, i.e., not only as logical or structural system descriptions, implies that a strong motivator to use software components is their ability to serve as descriptive entities, i.e., not only as reusable entities.

6. Verification of the investigation results

According to Robson [15] there are no standardised means of assuring complete reliability in a study that use flexible design strategy. We did however follow recommendations in [15] to minimise threats to the reliability of the conducted study by:

- Studying and minimising possible sources of biases.
- Describing the application characteristics, properties etc. (section 3) based on information from notes and tape recordings taken during the interviews.
- Interpreting the respondents answers at a group basis when necessary.
- Verifying the observations (section 4 and 5), with the help of a senior designer with expertise in vehicular real-time systems.
- Verifying our observations (section 4) with representatives from two of the participating companies.

References

[1] Arcticus Systems. Home page. http://www.arcticus.se

[2] Ardence, Home page, http://www.vci.com

[3] I. Crnkovic, M. Larsson, *Building Reliable Component-Based Software Systems*, Artech House, 2002, ISBN 1-58053-327-2

[4] Enea Embedded Technology, Home page, http://www.ose.com

[5] B. Graaf, M. Lormans, H. Toetenel, “Embedded Software Engineering: The State of the Practice”, IEEE Software, Volume 20, Issue 6, 2003

[6] B. Graaf, M. Lormans, H. Toetenel, “Software Technologies for Embedded Systems: An Industry Inventory”, 4th International Conference on Product Focused Software Process Improvement, Rovaniemi, Finland, 2002

[7] International Electrotechnical Commission (IEC), “Functional Safety and IEC 61508”, May 2000, Geneva, Switzerland

[8] P. Koopman, “Embedded System Design Issues (the Rest of the Story)”, Proceedings of the International Conference on Computer Design (ICCD), Austin, October, 1996

[9] Microsoft, Win CE.NET Home page, http://msdn.microsoft.com/embedded/prevver/ce.net/

[10] A. Möller, J. Fröberg, M. Nolin, “Industrial Requirements on Component Technologies for Embedded Systems”, International Symposium on Component-based Software Engineering (CBSE7), Edinburgh, Scotland, May, 2004

[11] A. Möller, M. Åkerholm, J. Fröberg, M. Nolin, “Industrial Grading of Quality Requirements for Automotive Software Component Technologies”, Embedded Real-Time Systems Implementation Workshop in conjunction with the 26th IEEE International Real-Time Systems Symposium, Miami, USA, December, 2005

[12] C. Norström, M. Gustafsson, K. Sandström, J. Mäki-Turja, N-E,. Bånkestad, ”Experiences from Introducing State-of-the-art Real-Time Techniques in the Automotive Industry”, Eight Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Washington, US, April, 2001

[13] P.G. Paulin, C. Liem, M. Cornero, F. Nacabal, G. Goossens, “Embedded Software in Real-Time Signal Processing Systems: Application and Architecture Trends”, Proceedings of the IEEE, Volume 85, Issue 3, 1997

[14] M G.E. Peterson, “User satisfaction surveys, what the engineer should know”, Proceedings of the Ninth IEEE Symposium on Computer-Based Medical Systems, June 1996

[15] C. Robson, *Real World Research*, 2nd edition, Blackwell Publishing, 2002, ISBN 0-631-21305-8

[16] Wind River, Home page, http://www.windriver.com

[17] R. Yin, *Case Study Research*, 3rd edition, Sage Publications, 2003, ISBN 0-7619-2553-8

[18] M. Åkerholm, J. Fredriksson, K. Sandström, I. Crnkovic, “Quality Attribute Support in a Component Technology for Vehicular Software”, Fourth Conference on Software Engineering Research and Practice in Sweden, Linköping, Sweden, October, 2004