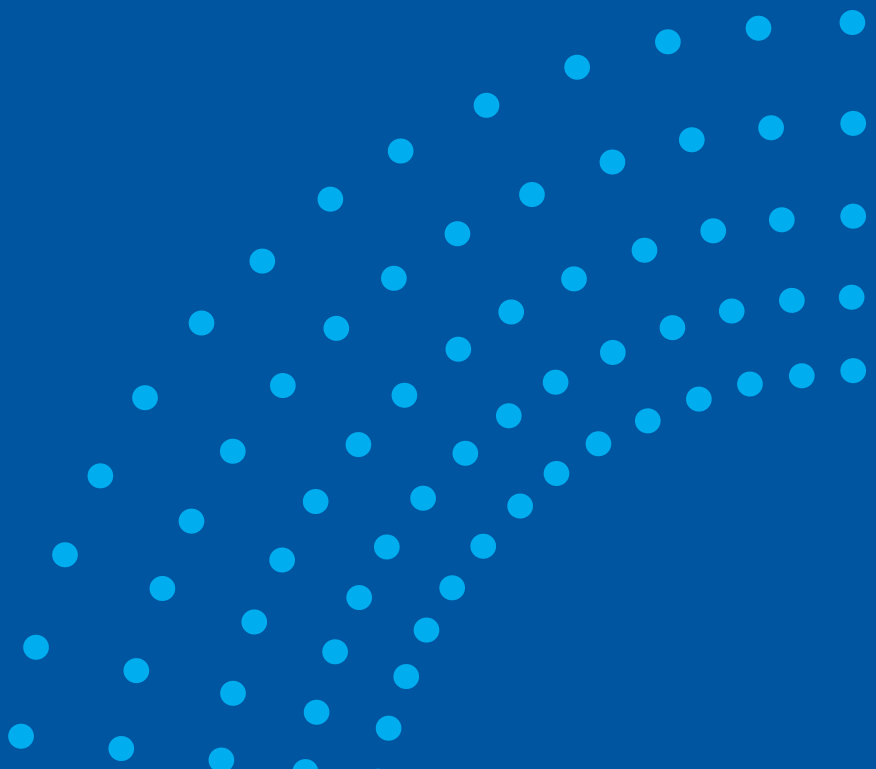
Master Thesis in Informatics

A Software Architecture Approach to Remote Vehicle Diagnostics

Ali Karimi, Johan Olsson & Johan Rydell
Gothenburg, Sweden 2004



REPORT NO. 2004 : 59

A Software Architecture Approach to Remote Vehicle Diagnostics

An Architectural Design Intended for the Commercial Vehicle Aftermarket

Ali Karimi, Johan Olsson & Johan Rydell

Supervisor: Jonas Kuschel

Department of Informatics
IT UNIVERSITY OF GÖTEBORG
GÖTEBORG UNIVERSITY AND CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2004

A Software Architecture Approach to Remote Vehicle Diagnostics
An Architectural Design Intended for the Commercial Vehicle Aftermarket

Ali Karimi, Johan Olsson & Johan Rydell

Report no 2004 : 59
ISSN: 1651-4769
Department of Informatics
IT University of Göteborg
Göteborg University and Chalmers University of Technology
P O Box 8718
SE – 402 75 Göteborg
Sweden
Telephone + 46 (0)31-772 4895

Chalmers Repro
Göteborg, Sweden 2004

A Software Architecture Approach to Remote Vehicle Diagnostics

An Architectural Design Intended for the Commercial Vehicle Aftermarket

Ali Karimi, Johan Olsson & Johan Rydell

Department of Informatics

IT University of Göteborg

Göteborg University and Chalmers University of Technology

SUMMARY

Over the past decades the technological development of vehicles has evolved rapidly and today almost everything in a vehicle is controlled by electronic systems. At present time there are no signs of any weakening regarding this trend. With the maturity of wireless communication technologies it is possible to provide new kind of telematics services. A category of telematics concern vehicle maintenance and these services depend and take advantage of electronic systems in vehicles. Remote vehicle diagnostics is such a service, which provides opportunities to conduct vehicle diagnostics work remotely. For a while now, there has been optimistic expectations on the potential of providing such services to customers among the automotive industry. These expectations have largely been based on the assumption that remote vehicle diagnostics could prevent breakdowns by detecting vehicle problems at an early stage. However, the true potential have yet to be revealed and manufacturers struggle in finding profitable business models. This master thesis studies remote vehicle diagnostics intended for the commercial vehicle aftermarket. Reasons to why remote vehicle diagnostics services have failed to reach a strong market penetration are outlined and explained. We have found that such services are often based on a weak architectural design. Therefore, this study examines how a suitable software architecture, for an aftermarket remote vehicle diagnostics system, should be designed. We argue that such software architecture should focus on a well-prepared flexible design and cost-effectiveness. This study has used an explorative approach towards remote vehicle diagnostics with a clear connection to the thoughts of "Practical Informatics". The conclusion of our study is presented as several design principles that together contribute to software architecture suitable for the aftermarket. Further, we describe how we have implemented our architecture using prototyping, to validate the architecture in real environment and derivate the accomplishment of "proof of concept".

Keywords: Telematics, Aftermarket, Remote Vehicle Diagnostics, Software Architecture, Design Principles, Proof-of-concept Prototyping.

Acknowledgements

We would like to thank our supervisor Jonas Kuschel, at IT University of Göteborg, for extensive support and continuous guidance throughout this master thesis work. His expertise and knowledge have truly been a great source of inspiration. Further, we would like to thank everyone at Newmad Technologies, most specifically our industrial supervisor Henrik Fagrell for provided assistance, contacts and resources. And finally Volvo Parts for giving us the opportunity to carry through such outstanding master thesis.

Once again, a great thanks to everyone involved!

Ali Karimi

I would like to thank my beloved family, especially my parents, for believing in me. I owe you great gratitude for every step and direction in my life. I wouldn't have come so far without your support. Finally, I would also like to thank my wonderful girlfriend for her inexhaustible love and support during this journey.

Johan Olsson

I would like to thank my family and friends for their support during this project. I can honestly say that this master thesis work has been very challenging and I couldn't dream that we would reach so far with our work. I would also like to thank my colleagues, it has been a pleasure to work with you and hopefully we will have the chance to join forces in the future.

Johan Rydell

I would like to thank my beloved family for their never-ending love and support.
I owe you so much.

TABLE OF CONTENTS

1 Introduction and Background

The purpose with this initial chapter is to provide the reader with an introduction and background to areas immediately attached to this study. We describe the characteristics of some important fields to understand and consider when conducting this kind of studies. First we provide the reader with a short introduction to vehicle electronics and how these electronic systems have evolved during the last decades. Secondly, we give a brief summary of the telematics domain, and describe how telematics services most often utilize and depend on the electronics inside vehicles. We describe the characteristics of the vehicle aftermarket and what to consider when introducing telematics in this domain. Conclusively, we try to position the research field where this study is conducted. This is done to provide the reader with a feel for this study.

1.1 History of Vehicle Electronics

Much has happened in the evolution of the car since the beginning of its history some hundred years ago. This certainly applies to the technical development of the vehicle itself. A lot of this development is derived from the introduction of more and more sophisticated electronic systems inside the vehicle. Today, electronics in a vehicle amount a big share of the total manufacturing costs, but equally most of the automotive innovations originate from new electronics. According to vehicle manufacturers, safety, efficiency, and enjoyment are the fundamental reasons to the implementation of computing and new electronics in vehicles (Walker et al., 2001). Electronic systems in a modern vehicle have several fields of application, e.g. to control safety systems, optimizing braking effect, managing engine operation and emission levels. Further reasons to why vehicle manufacturers recognize the long term potential of electronic systems, are the ability to improve their product in many areas and to use them as powerful sales arguments against other vehicle manufacturers.

The first common field of application for vehicle electronics can be traced back to the beginning of the seventies when electronic ignition was introduced in the vehicle market. Since then, a great deal has happened when it comes to the amount and complexity of vehicle electronics (figure 1.1). Consider for example that in 1955 an average vehicle contained 45 meters of electrical wiring, whereas today an average vehicle can contain several kilometers of electrical wiring. Similar to that development, the amount of electrical circuits has increased rapidly in vehicles. Back in 1969 when Apollo 11 made its historical moon landing, it contained 150 Kbytes of computer memory. A vehicle of today uses a lot more memory only to keep the CD-player running smoothly. These examples illustrate, just how extensive the growth of vehicle electronics has been during the last decades, and yet there are no signs of any weakening regarding this trend.

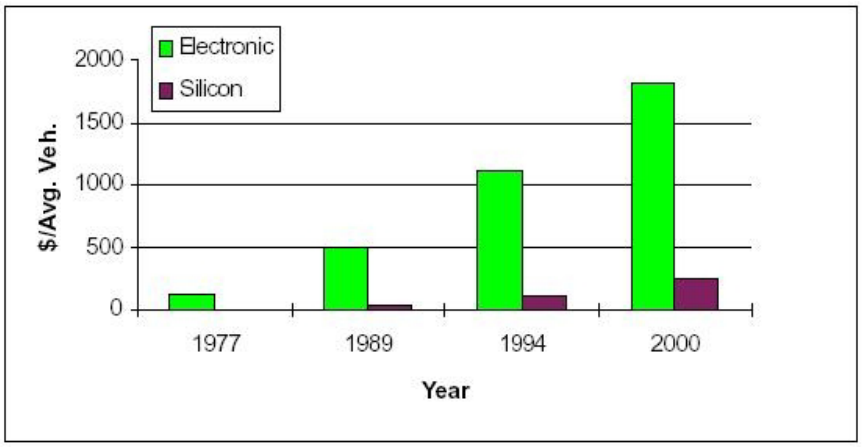


***Figure 1.1**: Electronic and silicon content per average vehicle* (Miller et al., 1998).

Much of the technical development of vehicles over the past decades, can be traced back to the necessity to complying with government requirements on exhaust emission control and reduced fuel consumption, as well as increased customer demands regarding safety, comfort and convenience. Faced with these external demands, the manufacturers began to investigate the possibilities to use electronic systems, as an assist to manage and control these functions, e.g. electronic ignition which improved fuel efficiency. This led to the introduction of new electronic systems which replaced entire mechanical and hydraulic applications, and to the adoption of computerized control systems inside the vehicles, referred to as electronic control units in the automotive industry. The number of such computerized control units soon started to increase, i.e. the engine, transmission and brakes etc. got an own dedicated control unit to manage that part of the vehicle. In the beginning these control units were quite independent from each other, but it did not take long before the automotive industry realized that advanced functions required a close cooperation between several units. Using sensor data from several control units enabled new electronic systems to be implemented, e.g. traction control, a system that process and manipulate sensor data from the vehicles wheels and engine (figure 1.2).

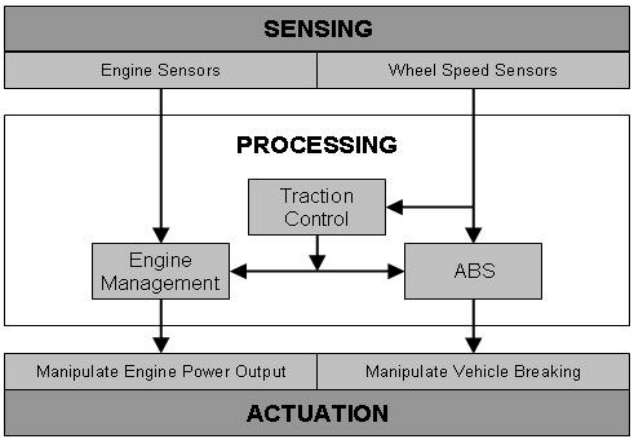


***Figure 1.2**: Example of traction control which uses sensor data from several control units.*

The standard solution to enable communication between control units was to use wiring, connecting the different units with each other. But adding more and more wiring led to many problems, such as increased vehicle weight, space consumption and weakened performance. These problems meant that fuel- and electrical power consumption increased in vehicles, and in the end the wiring problem hit a technological wall (Leen & Hefferman, 2002). The solution to that problem was to implement centralized and then distributed networks that connected each control unit through a small number of distribution buses based on serial protocols and thus replacing the need for point-to-point wiring. These networks enabled sharing of information and resources among electronic equipment. Today's vehicles contain several communication networks dedicated to different fields of application, e.g. high-speed networks aimed for critical systems (see chapter 4.2). The implementation of communication networks has had a positive effect on the wiring problem, even though new electronic systems are added continuously, increasing the need for additional wiring. For example, these networks made it possible for BMW to reduce the wiring in one of their models by 15 kilograms while enhancing functionality (ibid.).

1.2 Telematics

Concurrently with progress in automotive electronics, as described, various wireless communication technologies have emerged on the market. The recent development of such technologies has made it possible to remotely connect and exchange information with a vehicle, e.g. by using GPRS or UMTS. This opportunity to remotely communicate with a vehicle and its electronic systems has introduced interesting possibilities for new vehicle-related services to be developed. These services can for example collect information about the surrounding environment and then use this information to assist the driver or gather data from the vehicle electronic systems and transmit this information to a remote location for analysis. Such services, developed in this context most commonly are categorized under the generic term telematics. Although the term telematics lacks a generally stated definition, it is commonly referred to as the convergence of telecommunication and informatics (Nationalencyklopedin, 2004).

The early telematics applications primarily focused on services related to route guidance and safety, but lately we have seen that telematics has come to include a broad range of services with many different application areas, e.g. internet-enabling services, fleet management and vehicle diagnostics. Common to all emerging telematics services is that success highly depends on the ability among participants to sense the needs and desires of customers and development of products wrapped around customer-expressed values and not merely from the view of pushing telematics technology (Ford, 2002). Henfridsson et al. (2003) roughly divide the telematics domain into the following categories:

- Navigation and accessibility
- Safety and security
- Productivity
- Infotainment/entertainment
- Vehicle maintenance

Telematics services can also be divided according to their main application area, i.e. if they are intended to be used in the vehicle back seat, front seat or under the hood (Ford, 2002). By doing this rough division, entertainment and infotainment services would represent back seat, while front seat would be navigation and accessibility together with safety and security. Under the hood would also be safety and security together with productivity and vehicle maintenance services. Important to consider is that splitting telematics services in this manner is not always a straight forward task, often a service includes a combination of several fields of application. Yet another method used to group services is to divide them into generic and location-based services (Jameel et al., 1998). Generic services are not directly car-related but are of interest to drivers and passengers, e.g. entertainment services. Location-based services on the other hand are directly related to the vehicle-driving experience.

The first area of telematics to reach popularity among customers was navigation and accessibility services, providing route guidance to drivers using maps and position-tracking technologies. These systems used for route guidance, suggest what route to take by using turn-by-turn directions to a specified destination. Since the first navigation services, much has been done to improve the exactness and deliverance of the turn-by-turn instructions, i.e. that the instruction is correct and delivered at the right moment. This has been possible by the advancement of the GPS technology, which nowadays gives a much more accurate position then it did a few years back. However, route guidance has more opportunities to offer than just turn-by-turn instructions. Services recently developed, reveal an effort on/towards finding the optimal route (i.e. the fastest route) to a predefined destination rather than just the shortest route. An optimal route is calculated based on many variables other than just distance, e.g. ongoing road constructions, accidents and traffic jam situations. The system can thus suggest an optimal route based on these criteria's or notice the driver about the current traffic situation and present several alternative routes and then let the driver decide the best route from her preferences. To implement and collect required information the system must connect to external resources, in order to get information about the current traffic situation.

Services developed by vehicle manufacturers in order to enhance safety and security, incorporate sensing capabilities inside the vehicle that discovers accidental events. A safety service like that is an automatic crash notification, generated and sent by the vehicle automatically. For example, in case of airbag deployment or breakdown an automatic crash notification is sent to, e.g. a call center which can take proper actions, i.e. direct emergency- or roadside assistance to the location of the vehicle. The benefits of such a service would be a shorter response time to an emergency event, i.e. rescue personal get the emergency message immediately. In some rare cases, no one can call for assistance when an accident has occurred. Then an automatic crash notification is the only way to send for help. This aspect highlights another important feature with such services, namely the ability to automatically transfer location and other crash data collected from electronic systems, to emergency personal that can use this information to deal with the emergency in a more efficient manner.

Telematics services aiming to enhance productivity are usually connected to the area of fleet management. These services allow fleet managers to monitor and control the operations of their vehicles remotely. This is done to improve organizational performance by a more efficient transport planning and distribution of workforce in companies with large vehicle fleets, e.g. haulage-, logistics and delivery companies. Managers can monitor every vehicle at all times to check, e.g. position, fuel consumption, driving distance. This information is used to facilitate optimal routing strategies, service needs and analyze driver performance (Ford, 2002). Managers can discover differences in fuel consumption, which can indicate unnecessary hard driving styles. Fleet management systems use vehicle-tracking technologies, do basic data readings (e.g. current speed and total distance) from vehicle electronic systems, and transmit and present this data to fleet managers.

Services aimed towards infotainment and entertainment constitutes a category of telematics services, which in the long perspective has great potential. These services are often characterized by the intention to provide vehicles with internet access, targeted towards passengers. Internet-enabled cars offer delivery of entertainment services through multimedia devices, e.g. computer gaming, downloads of music and digital video. Internet connection also allows information services (i.e. infotainment), such as access to e-mail, news, sports, weather reporting, bank errands and stock quotes. A good example of an infotainment service providing benefits for drivers would be the ability to get information about upcoming gas stations (e.g. the driver receives information about sales offers and the current petrol price) when traveling on roads where he/she missing local knowledge.

Telematics services aimed towards improving vehicle maintenance have been predicted to have a great impact on the automotive market, e.g. services that try to predict and prevent vehicle breakdowns. In contrast, to entertainment services that are not directly related to the vehicle itself, these services are closely attached to the actual vehicle and its in-built electronic systems. The driver and passengers do not need to be aware of the system which operates in the background, much like a crash notification service does. Vehicle maintenance services, in a telematics context are often associated with the concept of remote vehicle diagnostics (RVD), covered in detail chapter 3.3. RVD is concerned about diagnosing and solving vehicle problems from a remote geographical location, e.g. a central service center or a local repair shop. The ability to wirelessly connect with a vehicle provides experts or service technicians with on-board data that can be examined to analyze its condition. Vehicle problems can be analyzed and maintenance operations such as software updates can be carried out remotely and thus prevent the need of an appointment in a repair shop. If the problem cannot be dealt with remotely the service technician can inform the driver of a service appointment or send roadside assistance to the vehicle in place. The field of RVD, and its related characteristics and possibilities, is the main focus throughout this study.

1.3 Aftermarket

Success of new telematics services depends, as mentioned in the previous section, exceedingly on the ability among participants to sense customer needs and the promotion of services that can satisfy those need and desires. To do this one must recognize and understand the characteristics of the market and customer, for which the service is intended. RVD services can be designed and offered to several different customers, depending on the context, in which the service is intended to be used. A vehicle's life cycle can be divided into three separate phases, namely research and development (R&D), production, and aftermarket. Customers of RVD services, in R&D and production phases, are vehicle manufacturer or companies tightly linked with the development of a vehicle. The customer in the aftermarket is the end-user of a vehicle, i.e. the individual or company that purchases or operates the vehicle. This study focuses on RVD intended for the aftermarket.

The vehicle aftermarket can be described as: the life phase that follows after a vehicle is sold or put into operational use, and it continues right until the vehicle is made obsolete. Consequently, the aftermarket makes up most of a vehicle's life. Aftermarket and its customers can be divided into the personal- and commercial vehicle market, i.e. private car owners and commercial companies. Vehicle aftermarket concerns maintenance, reparations and the spare parts market. Products and telematics services (e.g. RVD services) aimed towards vehicle end-users are also considered to be part of the aftermarket domain.

In recent years the aftermarket has come to grow in importance. For a vehicle manufacturer the maintenance work (i.e. service, reparations and spare parts) is highly profitable compared to other areas. While the maintenance and spare parts only cover 10 to 25 percent of sales it can comprise up to 50 percent of the manufacturers profits (Wright & Pugh, 2003). With this information in/on hand it comes without saying why manufacturers place more attention to the aftermarket business, and at the same time try to find new business models to control and protect their strong position in/on the market. This has been accentuated even more by a new EU-legislation that e.g. allows dealers and authorized repairers to use whatever service parts they choose, manufactured from any source that holds appropriate standards. This has forced vehicle manufacturers to act and find new business models, like establishing partnerships with dealers and repair shops to deliver repair and maintenance services (ibid.).

The characteristics of aftermarket business are e.g. lots of available customers. Anyone who owns a vehicle is an aftermarket customer. In contrast, a RVD service aimed towards R&D or production would only have one single customer. Because of the aftermarket diversity, different actors have the ability to offer RVD, e.g. vehicle manufacturers, dealer's repair shops and independent service providers. The aftermarket of RVD can therefore be described as having a many-to-many relationship (figure 1.3), i.e. several providers have large numbers of possible customers and customers have several manufacturers and repair shops etc. to choose from. This condition means that RVD solutions intended for the aftermarket must be delivered in a manner to be sold in large quantities, i.e. due to the large number of customers. Large number of produced

units implies lower unit price and increased maintainability. This depends on that there are different economic prerequisites in the aftermarket, e.g. compared to R&D, where only single units are produced. Everything that adds to the production cost of a vehicle (e.g. hardware associated to a RVD solutions), becomes exceedingly more expensive to the end customer. Vehicle manufacturers that could promote RVD are therefore cautious, in adding in-built equipment aimed for aftermarket use, which increasingly raises the purchase price to customers, thus manufacturers must keep the price of the vehicle to a minimum. Further, aside from the actual vehicle cost the aftermarket customers rarely have the capacity neither the wish to pay more than necessary for services such as RVD. Customers often choose the product or service that have the lowest price or are the easiest to maintain. Developers and service providers must consider these economic aspects when designing and introducing new RVD technology, e.g. if a manufacturer has a very expensive solution, customers will probably choose another manufacturers solution instead. For success of RVD in the aftermarket there is also a need to provide solutions, which in the end saves money or resources to the customer. This particularly applies to the commercial vehicle aftermarket where there is a much bigger focus on profitability and efficiency compared to the personal vehicle market.

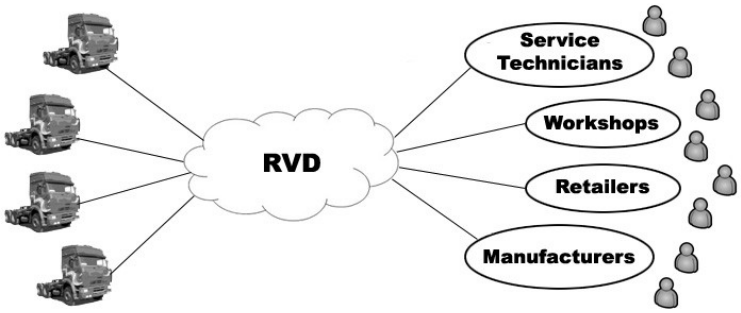


Figure 1.3*: Illustrates the business relationships in the aftermarket concerning RVD.*

In the vehicle aftermarket new business models have emerged, concerning the ownership of a vehicle. Traditionally a vehicle has been sold to the customer, i.e. the customer posses their own vehicles. Nowadays it has become increasingly common to lease a vehicle, i.e. instead of buying; the customer pays a periodic fee for using the vehicle. This business model mostly applies to the heavy commercial vehicle market, e.g. trucks and construction equipment. In the commercial vehicle market the "uptime" of a vehicle is very important, i.e. the period of time when a vehicle is available for use. The "downtime", i.e. the time when a vehicle is not available for use (e.g. due to a breakdown), is costly to companies and therefore the emphasis is to reduce the downtime. Another reason to why the commercial vehicle market suits this business model is because there is a focus on the core activities today, amongst companies like logistics- and haulage firms. For example, the core activity for a logistic company is to offer the service of delivering goods not to own and maintain a vehicle. Such companies want to leave issues like financing and insurance, maintenance and repairing in the hands of specialist (e.g. vehicle manufacturers) and instead concentrate on their own core activities. Advantages to vehicle manufacturers would be the ability to offer improved after-sales maintenance and support to customers. For several years vehicle

manufacturers have tried to control the aftermarket, including maintenance and spare parts sales. This becomes a lot easier to do for manufacturers with this business model, because they are the ones controlling maintenance work.

Vehicle manufacturers consider RVD as an opportunity to further increase their presence on the vehicle aftermarket and as a possibility to make maintenance and repair work more efficient. By offering RVD to customers, as a part of a service deal package, the manufacturer gains a better control on the use of spare parts, i.e. this will lead to a higher employment of vehicle manufacturer's spare parts. To reach a good market penetration of RVD, manufacturers must prove obvious benefits to their customers, e.g. economical benefits. RVD has the potential to promote efficiency and reduce downtime, and in the end save money to the companies. This would be done using RVD to discover problems that otherwise could have remained hidden and the prospect of detecting serious problems or failing components at an early stage, and avoid them from evolving to a breakdown when the vehicle is in operational use (Campos et al., 2002). The downtime of a vehicle can also be reduced by using RVD to improve maintenance scheduling and an improved repair- and spare parts planning. If the repair shop knows which parts they need in order to repair a vehicle, they can plan ahead and have the right parts in stock when repair work begins (Dennis & Kambil, 2003).

1.4 Research Positioning

With this introduction to the advancement of vehicle electronics and how much these electronic systems shape the performance of a modern vehicle, as well as all opportunities that vehicle electronics offers in terms of telematics services to vehicle manufacturers, service technicians, telematics service providers and end-customers, it is suitable to try and identify the research field where this study is conducted.

When dealing with telematics services one must be aware of the important role that wireless communication technologies have in these solutions. For telematics services to be able to exchange information, forth and back with a vehicle, it is necessary to utilize such technologies. Consequently, we deal with wireless communications and its theoretical framework in this study. We have to consider characteristics, such as complexity, performance, bandwidth and range of different technologies.

In the telematics section we outlined that this study focuses foremost on RVD and the vehicle maintenance part of the telematics domain. In contrast to many other telematics services, e.g. entertainment services, RVD solutions are closely attached to electronic systems inside vehicles. RVD means that diagnostics data must be requested from the vehicle electronic systems, i.e. a RVD solution has to communicate with a vehicle's control units. To apply RVD one must understand how to communicate with vehicle electronic systems and how to deal with diagnostics data, i.e. how to do parameter readings and interpret the requested data. As mentioned earlier our focus during this study is on RVD, thus we need to deal with vehicle electronics and associated software.

As outlined earlier (see chapter 1.3), a vehicle's life cycle can be divided into three separate phases, namely research and development (R&D), production, and aftermarket. Most of the vehicle's life concerns the aftermarket. To vehicle manufacturers the aftermarket is highly profitable and there is an ongoing process in finding new business models to protect and increase aftermarket business. Yet, the question still remains for vehicle manufacturers whether or not RVD can be an opportunity to succeed and find new business models in the aftermarket. Can RVD be useful to customers and especially to the commercial vehicle market as a way of reducing vehicle downtime?

In this study we focus on the opportunities of RVD in the commercial vehicle aftermarket. To do this we have to recognize the characteristics of the aftermarket and what to consider when developing telematics services aimed towards the aftermarket. With this presentation of the areas covered in this study, it is possible to position our research work, which we illustrate below in figure 1.4.

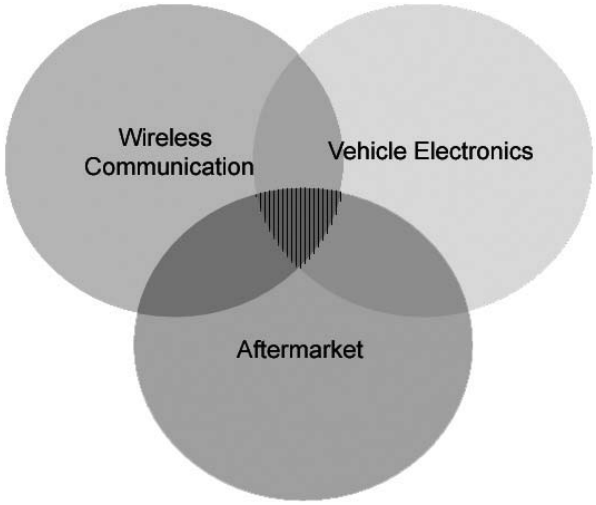


Figure 1.4*: Illustrates in which research area this study's is conducted.*

Our study is conducted where the fields of wireless communications, vehicle electronics and aftermarket business converges. This delimitated field of research concerns telematics and in particular RVD, but could also be described as applied information technology.

2 Research Question

Vehicle electronics has evolved quickly during the past decades and as stated before there are no reasons to believe that this trend would decelerate in the nearest future. The large amount and sophistication of electronics inside a modern vehicle brings new challenges as well as new opportunities to the automotive market, concerning manufacturers, repair shops, service technicians and customers. Vehicle manufacturers' situation is affected by the development of electronic systems, not only because they are responsible for implementing them into modern vehicles, but also because they have to assure that the systems are reliable and free from any kind of hidden defects. If such issues are overlooked, the manufacturer's reputation will be damaged and there will be increased expenses due to unscheduled service costs. A worst case scenario would be that the manufacturer needs to immediately recall large amount of vehicles to change or upgrade a critical electronic system.

The occurrence of complex electronic systems has undoubtedly had an effect on how the diagnosis of a vehicle is carried out by the service technician. In the past the service technician only had the mechanical parts of a vehicle to consider when diagnosing the vehicle. Today, service technicians not only have to master the mechanical parts, but they also need to manage the electronic systems of a vehicle in order to place a correct diagnose. This circumstance has made it necessary to provide service technicians with proper diagnostics methods and tools. These tools, connected to the vehicle's electronics, provide service technicians with diagnostics data, which identifies the cause of a problem or provides important information for further trouble-shooting. Computerized diagnostic tools in combination with telematics offers the possibility to evolve vehicle diagnostics as currently undertaken into RVD.

There has been an optimistic forecast on the future use and application of RVD among the automotive industry, and there still is even though the full potential has yet to be exposed (Ford, 2002, Bisdikian et al., 2002, Campos et al., 2002). These expectations have mainly been based on the assumption that RVD would reduce the need of local service technicians, and that many vehicle related problems instead could be solved by an expert, working from a central service center (Kuschel & Ljungberg, 2004). Despite all forecasts, there is an ongoing struggle amongst vehicle manufacturers in finding profitable models for the delivery of RVD solutions, although some examples of commercial solutions exist like GM´s OnStar and Networkcar (Hansen & Wolfe, 2004).

Bisdikian et al. (2002) mention two reasons for the lack of success of telematics services in general. First, they argue that many telematics services are developed for a specific communication technology and that they are based on closed platforms, i.e. isolated solutions that are unable to communicate with each other. This leads to a limited market penetration and that no services are developed on the existing solution. When it comes to RVD services it is important to understand that diagnostics systems inside vehicles are closely controlled by vehicle manufacturers and that these systems differ between manufacturers. This circumstance means that it is complicated to develop independent RVD solutions, although attempts are made with independent platforms, e.g. Open

Services Gateway initiative (OSGi Alliance, 2004). Consequently, many vehicle brands have developed their own solution, see GM´s OnStar. The other reason mentioned, by Bisdikian et al. (2002), is the absence of a mutual approach among developers, on how to deal with issues like wireless connections, how mobile platforms should be designed and how to design services for mobile users, e.g. vehicle passengers.

Besides, the more general guidelines stated above, one must consider three major issues when designing a RVD solution; first the wireless communication technologies, second the system architecture, and third the user interface design (Jameel et al., 1998). Wireless technologies are common to mobile services, not only in the telematics domain. The distinction between telematics services and many other mobile services is that vehicles are highly mobile. Therefore such solutions must deal with many different wireless technologies and be able to frequently alter between them, in order to function as supposed to at all time. The most important issue to consider when developing RVD solutions is the system architectural design, e.g. architecture and components distribution, and hardware requirements. An essential architecture decision that has to be made is whether to use a thin or a thick vehicle client. User interface design concerns, among several things the selection of desirable diagnostics data and the decision on how to present this data to service technicians, e.g. through a web interface.

When designing the system architecture it is important to identify the market and customer for which the RVD solution is intended. Further, it is important to understand the needs, desires and characteristics of market and customers. This latter aspect, highlights a reason to the lack of success of aftermarket RVD solutions in general, namely the proved tendency of such, to contain costly and advanced hardware, e.g. like solutions used in the R&D phase (Fagrell & Kuschel, 2003). The same solutions are usually based on a weak architectural design, i.e. the design of the architecture is not considered and is only put together in a rush to manage some critical R&D requirements. This performance is maybe acceptable in the R&D phase, where space and hardware is not an issue but certainly not in the vehicle aftermarket. Unfortunately, it is often similar system architectures, like the ones used in R&D that is also proposed for solutions intended for the aftermarket. This means that such RVD services would become expensive to implement and both costly to buy and maintain for customers. This strives against the conception among manufacturers, not to implement expensive additional hardware aimed towards the aftermarket, which raises the manufacturing cost and purchase price (ibid.). To the manufacturer this could mean that vehicle sales become tougher due to a higher price compared to that of competitors. This also collides with the fundamental characteristics of the commercial vehicle aftermarket, where low- costs and maintainability (e.g. low purchase price and no need for expensive upgrades) are important sales arguments. These circumstances can not be overlooked when dealing with aftermarket RVD solutions.

Based on the characteristics and requirements of RVD intended for the commercial aftermarket, we formulate the following research question, which we try to answer in this study:

What would be a suitable software architecture for a remote diagnostics system that meets the requirements of the commercial vehicle market?

2.1 Purpose

To answer our research question many subjects need to be considered and examined. When working with RVD one need to get a basic understanding on how vehicle diagnostics work is applied in reality and recognize the context in which RVD can be useful for customers and service providers. It is also important to identify, all actors RVD concerns, i.e. vehicles, vehicle manufacturers, service technicians, experts, customers and how these interact with each other, e.g. in a required maintenance situation. The field of RVD is also highly technical, and areas like vehicle electronic systems, interpretation of diagnostics data and wireless communications are essential to master in order to answer the research question, e.g. how to communicate with a vehicle's electronic systems and how to transmit diagnostics data wirelessly.

The main purpose of this study is, as the research question implies, to design a suitable software architecture for a RVD system, aimed for the commercial vehicle market. Our ambition is to examine what parts and components such software architecture should hold and how these components should interact with each other to meet the described characteristics of RVD. Based on that analysis, we intend to design a software architecture that supports several application scenarios of RVD. The proposed system architecture is based on a couple of findings, which we present as our design principles for a RVD system, aimed for the commercial vehicle aftermarket.

Another predefined objective of this study is to give a practical- as well as a theoretical contribution. The aim is to implement the proposed architectural design, to demonstrate several applications of RVD and validate the concept and its reliability in a real environment. The validation of the prototype system is done to ensure that its intended purpose is fulfilled.

2.2 Delimitations

In this study we make some delimitation to our research work. The architecture for RVD that is designed and implemented targets the commercial vehicle market and not the private vehicle market. This delimitation to the field of application of our architecture is done, based on the discussion in previous sections about the characteristics of RVD. This means that we also choose to adapt our architecture to the aftermarket and thus leave out the R&D and production stages.

We believe that there are better probabilities to utilize the enabling properties of RVD in the commercial vehicle market than in the personal vehicle market, e.g. by reducing downtime, a benefit which is not equally important to the single individual car owner. The commercial vehicle market should be of more interest in RVD solutions. We also extend this delimitation to concern heavy commercial vehicles, e.g. trucks and construction equipment. This is done because we believe that RVD can be particularly

useful to this market segment compared to others, e.g. reduction of downtime is more crucial for specialized construction equipment.

It is a technically complex task to enable RVD. There are many issues to consider when designing and implementing this kind of architecture, e.g. wireless technologies. The wireless communication which RVD so heavily depends on is a highly technical and complicated field. There are many problems, still to be solved concerning wireless communication technologies, e.g. unreliable connections and data security. In this area we make the delimitation, not to concern about data security and environmental disturbance when dealing with wireless communications.

3 Related Work

This chapter presents related work within the field of vehicle diagnostics and maintenance. A summary of different diagnostics methods and techniques is given together with an explanation of associated concepts and terms. Conclusively, we describe RVD and what opportunities this technology brings for both diagnostics work practice and aftermarket business solutions.

3.1 Diagnostics

Vehicle maintenance work has been transformed over the years. The rising share of electronics in modern vehicles requires improved diagnostics methods and equipment in order to accurately locate and diagnose any malfunctions. Service technicians can no longer merely rely on visual and physical inspections alone to resolve vehicle problems. Computerized diagnostics equipment and knowledge about the use of those systems are therefore important assets to the contemporary service technician.

The challenge for developers of diagnostics tools is to support the service technicians in their way of working, by providing relevant data that can be used to detect problems or tracing the problem cause. One dilemma with diagnostics data is that it often identifies the effect of a problem and not the cause of the problem (Kuschel & Ljungberg, 2004). In these cases the guidelines for service technicians are not enough to make the correct conclusion of the problem. The service technician then needs to use his/her collected experience of similar problems to find the real problem and the correct solution (ibid.).

Hamilton (2002) presents a more organized perspective and description of diagnostics and its signification. He claims that diagnostics, from a general point of view, should answer three main questions: Is there any problems? What is the problem? What possible actions can be taken? Hamilton also presents four different steps that are acquired in a diagnostics cycle (FDDR):

1. *Failure* – Deviating behavior in a system component.
2. *Detection* – Internal, embedded diagnostics systems that monitor deviations. This can be either hardware or software solutions.
3. *Diagnosis* – Analysis of the collected data from the embedded system. Information that indicates "what" is wrong but not the "cause".
4. *Recovery* – The necessary actions to be taken based on collected diagnostics data.

Improved vehicle diagnostics is imperative in order to find occurring failures in a timely and cost-effective manner. This is especially important when dealing with maintenance of commercial vehicles, where reduction of downtime is a major factor for the customer. A properly performed diagnostics routine reduces time spent by the service technician on troubleshooting procedures. Vehicle manufacturers and suppliers thus share a common interest in minimizing expensive maintenance work, recall procedures and warranty claims.

The traditional usage of diagnostics functions concerns reading error messages from the vehicle's on-board system memory, read-outs of vehicle parametric data and reprogramming of electronic control unit software. These types of diagnostics functions are done using an on-board system, off-board system, or a combination of the two.

3.1.1 On-board/Off-board

Vehicle diagnostics has not always been performed according to a standardized framework. Manufacturers followed their own system with a customized set of signals. It was not until the year 1988 Society of Automotive Engineers (SAE) initiated standardization of test signals. Further revision by Environmental Protection Agency (EPA) has led to mature standards such as OBD (On-Board Diagnostics).

OBD is an on-board diagnostics system integrated into the majority of all newly produced vehicles. An on-board system provides incorporated diagnostics capabilities that monitor virtually every component that can affect vehicle performance (Carr, 2004). An OBD system can deal with environmental issues related to pollution and emission levels by monitoring and adjusting the necessary system components. The system is responsible for performing diagnostics routines on each component to make sure that it is functioning properly. If a problem is detected, the on-board system illuminates a warning light on the dashboard, alerting the driver.

Diagnostics can also be carried out "off-board", i.e. retrieving information for external analysis, so-called off-board diagnostics. Such systems are normally used to carry through complex diagnostics functions whenever time, space and processing power is of importance. Volvo VCADS Pro is an example of such an off-board diagnostics system where data can be exported to spreadsheets or visualized in graphing applications for further analysis. The off-board diagnostics tool usually runs on a laptop computer or a handheld device connected to the diagnostics outlet. These tools are commonly used by service technicians in workshops to isolate and pinpoint the nature of the problem.

***Figure 3.1**: Examples of the two types of diagnostics systems, where the dashboard represents the on-board system and the laptop computer, with VCADS Pro software, represent the off-board system.*

However, most often it is the combination of on-board and off-board diagnostics that provides the most desirable results (Maintenance Council´s (TMC), 1998). For instance,

self-correcting and data acquisition operations should be performed in place, i.e. on-board, while time-consuming operations such as data processing and manipulation operations should be carried out by an off-board diagnostics system. Off-board diagnostics should be able to define, store and visualize accessible snapshot data but also offer continuous surveillance of status information. It should be able to interpret and clear the occurrence of failures (ibid.).

3.2 Prognostics

Prognostics as well as diagnostics are processes that evaluate a system's condition. Diagnostics deals with the current condition of a system based on observed symptoms, whereas the aim of prognostics is to make evaluations of the future condition. Prognostic assesses the current health of a system and at the same time predicts its remaining lifetime based on an estimation of the gradual degradation in the operational capabilities of the system (Luo et al., 2003). The following working definition of prognostics is suggested by Hess *et al.* (cited in Mathur et al., 2001):

"The capability to provide early detection and isolation of a precursor and/or incipient fault condition to a component or sub-element failure condition, and to have the technology and means to manage and predict the progression of this fault condition to component failure."

The introductory part of this definition deals with detection and isolation of fault conditions and could actually be considered to be a diagnostics process. The close relationship between the two processes makes it reasonable to look at them in correlation to each other, although the results from these processes have different impact on decision-making operations. Results gained from the diagnostics process are useful in making reactive decisions, for instance if a component should be repaired or replaced. Prognostic examination results aim to prevent and avoid incipient faults in a proactive manner, with the ambition to maximize the service life of the component while minimizing operational risk (Mathur et al., 2001).

The driving force of prognostics stems from manufacturers desire to increase their share on the service market, by offering their customers novel and attractive service contracts. In some of these new service offerings, the parts and billing model is replaced by a guaranteed uptime model. Thus, manufacturers have an increased motivation to maintain their customers' equipment in operational order (Vachtsevanos & Wang, 1999). A heavy vehicle such as a truck contains several pieces of equipment that experience performance degradation during operation, due to erosion, friction, and internal damage and so fourth. The industries of today are dealing with significant issues such as prolonging the lifetime of its critical processes and providing maintenance on an "as-needed" basis. Important parts of these ideas include the ability to diagnose impending breakdowns, prognosis of the remaining functional lifetime of the process and as a result improve safety, maintenance operations scheduling, lower costs of maintenance, and minimize downtime.

3.3 Remote Vehicle Diagnostics

The technological progress in the areas of automotive electronics and wireless communications brings new possibilities to the process of diagnosing vehicle failure situations. Previously, diagnostics work was restricted to the workshop where the vehicle and service technicians had to be co-located, but this is not necessarily the case with RVD. Using wireless technology and connecting to the electronic system of a vehicle, it is possible to perform diagnostics work from a distant location. Currently, there is no general definition of RVD but a universally applicable description is provided by market analysts Frost & Sullivan:

"*The ability to access the vehicle's performance parameters and trouble codes in case of malfunction using a wireless network, and provide necessary support services."* (Valsan, 2002)

The prospects of RVD are lucrative, particularly for the commercial vehicle market where the availability of the vehicle is especially important. For example collection and analysis of diagnostics data concerning vehicle performance during operation could assist vehicle manufacturers in discovering incipient quality problems, such as failure with mechanical parts and/or electronic components. RVD enables to adjust the performance of a vehicle without the need of physical intervention, i.e. mending mechanical parts performance, since a vehicle and its operation to a large extent is supervised and controlled by computerized components (see chapter 1.1 & 4.3). Examples of companies that have made substantial investments in RVD are General Motors, Bosch and Reynolds & Reynolds. GM's commercial solution, called OnStar, provides the driver with the opportunity to connect to a call center where an operator can retrieve diagnostics data from the vehicle. Based on the accumulated data the operator might suggest awaiting roadside assistance or schedule a service appointment. Yet, another example is the Reynolds & Reynolds' solution Networkcar. It is a RVD/location service serving both consumers and fleets. Networkcar automatically downloads trouble codes, location and similar vehicle data from the vehicle for analysis (Hansen & Wolfe, 2004).

Vehicle manufacturers assume that the introduction of RVD, in the manner described above, will reduce the need of local service technicians. Instead, many vehicle related problems can be solved by experts working from a central service center. Vehicle manufacturers believe that such infrastructure would effectively improve vehicle maintenance as well as complying with the economical prerequisites of the commercial service market. This centralized view on how RVD should be applied has been the dominating one among the automotive industry (Kuschel & Ljungberg, 2004). However, there are alternative approaches, compared to the centralized model of RVD. Kuschel & Ljungberg (2004) characterize diagnostic work practice into three main findings: co-location, collaboration practice and reliance of local knowledge. Based on these assumptions a decentralized model is suggested. They argue that RVD should be introduced on a repair shop level, supporting local service technicians instead of central experts. They consider that service technicians' local knowledge and expertise, and their relationship with both vehicle and customer, are important requirements in order to have successful usage of RVD.

Regardless of the type of diagnostic model (i.e. either using the centralized or decentralized model) and/or any kind of promising opportunity provided by RVD, there are still some contradictory opinions about what RVD actually bring about. For instance, Janet Howells-Tierny (2002) states that there are still many obstacles left to overcome before RVD and solid business cases can be implemented successfully. Further, she argues that the main problems concern cost-efficiency and availability of wireless communication infrastructure. Howells-Tierny also states that the reception of error codes only provides clues to fixing vehicle problems and requires the service technician to conduct thorough investigations in order to find the root problem (ibid.).

4 Technical Framework

This chapter introduces some technical concepts of vehicle diagnostics. We give an overview of the technology in modern vehicles that enables diagnostics examination using electronic equipment. First we describe the internal networks and the communication protocols used for information exchange with the electronic control units, and then we explain/detail diagnostics data in the form of a diagnostics trouble code.

4.1 Mechatronics

Mechatronics can briefly be described as design of computer controlled electromechanical systems. The focus in this domain resides in integrating electronics in mechanical components. An example of a mechatronic system is Anti-lock Braking System (ABS), which is basically a mechanical system but requires an integrated design of electrical computerized systems in order to function properly (University of Waterloo, 2004). Mechatronics also concerns software and programming techniques. The need for reliable communication, security and synchronization between the integrated components in a vehicle requires a high degree of failsafe programming. For example, a microprocessor must not reside in a non-functional state despite a malfunctioning measuring component. Further, the electronic complexity in a vehicle increases the need for diagnostics of its functions and components, e.g. an occurred error should be found and isolated at an early stage to minimize interference with the rest of the system.

In response to these requirements researchers have developed in-vehicle networks, also referred to as multiplexing (Intel Corporation, 2004). It is a method for transferring data among distributed electronic modules via a serial data bus. The number of dedicated point-to-point wire connections can be reduced by combining the signals on a single wire through time division multiplexing. The distributed architecture of multiplexing networks also facilitates modifications of the system. Changes made through software updates only apply to the concerned units. The application of these in-vehicle networks, instead of dedicated point-to-point wiring, significantly improves cost-efficiency, reliability and serviceability.

For varying purposes, different types of networks exist in vehicles today, e.g.:

1. *LIN (Local Interconnect Network)* – A serial bus system for networking between distributed electronic systems in vehicles. LIN provides a cost-effective alternative for bus communication, where the bandwidth and versatility of CAN is not required (Bosch, 2004).
2. *CAN (Control Area Network)* – CAN is a high-integrity serial data communications bus for real-time applications (Bosch, 2004) It is widely acknowledged for its excellent error detection and confinement capabilities. Common areas of usage are industrial automation and control applications.
3. *MOST (Media Oriented System Transport)* – MOST is a network technology based on synchronous data communication. It is the primary choice for in-vehicle infotainment and entertainment services (MOST Cooperation, 2004).

4. *FlexRay* – A new communication protocol designed for advanced control systems that demands high bit-rate and link capacity. FlexRay is used in "X-by-Wire" systems (AD&P, 2004).

Conclusively, the networks have different areas of usage; LIN is used for functions requiring low bandwidth, CAN for medium bandwidth, MOST for high bandwidth and FlexRay for functions requiring high security, such as steering- and brake systems.

Dealing with heavy commercial vehicles in conjunction with diagnostics, the CAN network and its specified communication protocols are the most suitable and widely used technologies for communication and management of different kind of electronic control units.

4.2 CAN and Heavy Commercial Vehicles

CAN is a high-integrity serial data communications bus supporting distributed real-time systems (Waern, 2003). It was originally developed for the automotive industry but because of its well-proven robustness and security, it has also gained success in many other industrial control applications, such as in the automation industry.

CAN is based on the so-called broadcast communication mechanism. The broadcast communication is achieved by using a message oriented transmission protocol. This means that messages sent over the network do not define stations or station addresses. Instead each message is recognized by a specific identifier which is within the whole network. Due to the broadcast property all messages sent over the network are visible to all nodes, but mechanisms are provided by the protocol (e.g. local filtering) forcing each node to only response to self-concerned information. Further, each message has its own priority, thus solving the problem of several stations competing for bus access simultaneously.

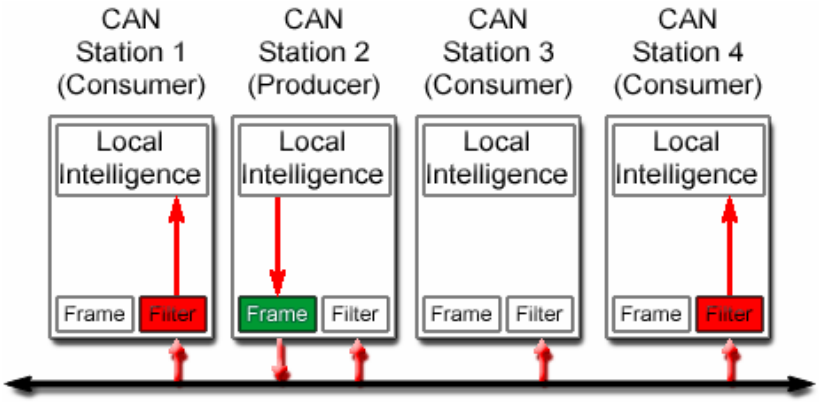


***Figure 4.1**: Illustration of data exchange between different nodes.*

The fact that CAN is message-oriented assures high flexibility, i.e. it is easy to add and remove nodes on the network. There is no CAN related limitation on the number of nodes on the network. The message-oriented architecture also contributes to increased

reliability on data passed between nodes, as a consequence of the fact that all nodes are referencing the same data. The architecture also prevents a situation where multiple sources generate the same data (Henfridsson et al., 2002).

The committee for heavy vehicles at SAE began the development of a framework for CAN-based applications and released the first specification in 1988. SAE also produced the J1708 and J1939 protocols for communication between each node within the network (SAE, 2004). The CAN bus based on J1939 is the faster of the two, and is used for transmission of the system's control signals. As a result of the high speed data transfer rate (up to 1 Mbits/sec) a flexible system, with the capability to manage various alterations, is created. System information and diagnostic data is provided using the other CAN bus, i.e. the one based on J1708/J1587. The picture below illustrates how the standardization of the protocol is structured according to the Open System Interconnection (OSI) model.

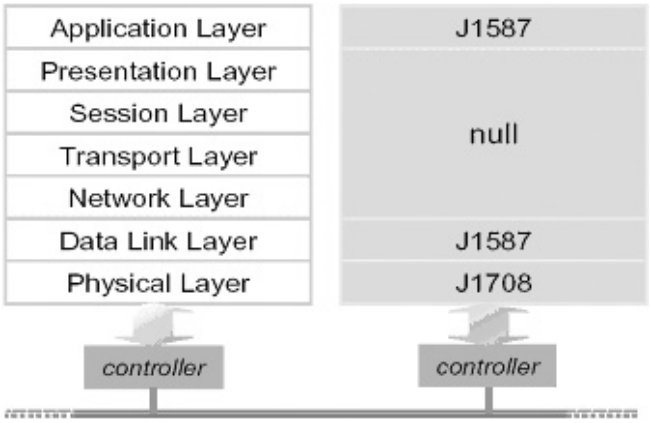


***Figure 4.2**: Relation between J1708/J1587 and the OSI-model.*

J1708 defines the physical layer of the OSI-model, while J1587 defines the remaining layers. The advantage of utilizing standards is primarily the facilitation of compatibility between different products from one or many different manufacturers.

For distribution of system information between different control units and the possibility to read diagnostics data, the communication channel based on J1708/J1587 is used. This is done by broadcasting messages on the bus. J1708 specifies a message format which is used for communication on the bus. According to the protocol specification a message consists of the following parts:

- *Message identification* – Abbreviated to MID (Message Identifier). This field is one byte long and stretches from 0 to 255. Each MID identifies a unit or a transmitter, and is unique across the system.
- *Data* – The actual information related to the specific node on the network.
- *Checksum* – Used by the receiver for validation of the message.

J1587 belongs to a higher layer than J1708 in relation to the OSI-model. It needs a similar format on each message but with the addition of the data field space. J1587 defines a message type called PID (Parameter Identifier). To be able to request specific information, e.g. the engine speed, both MID and the corresponding PID must be supplied. The illustration below shows the valid message format for each protocol.

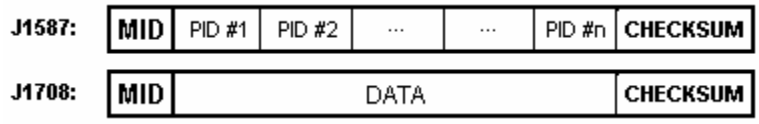


***Figure 4.3**: Illustration of the message format.*

4.3 Electronic Control Unit – ECU

Electronic control units are sophisticated microprocessor-based systems that perform numerous real-time control functions. Some of these functions concern, e.g. ABS, transmission, air-conditioning, engine and emission control. The supervision of these systems is possible with data collected from a variety of sensors attached to the components inside the vehicle. The sensors measure values and states of vehicle parameters during operation. Collected data is continuously transferred to each control unit, which is responsible for executing the necessary adjustments and settings needed for optimal system performance. For example, the engine ECU can control the amount of fuel injected in to each cylinder, keeping the performance of the engine optimized. It can also provide engine protection if an abnormal value of a parameter is encountered. For instance, if the oil pressure drops below a critical level, a pre-programmed instruction in the ECU software may shut the engine off or decrease the power output.

Abnormal data values from parameters and sensors generate Diagnostics Trouble Codes (DTCs) that are stored in the ECUs. With a diagnostics tool connected to the ECUs, large amounts of diagnostics data can be requested from the system, providing vital information to a service technician. Having the ability to communicate with each ECU on the network the service technician can read and erase DTCs, display parameter conditions and update the ECUs with new software patches.

A truck consists of several different types of electronic control units, e.g.:

- *EECU* – Engine Electronic Control Unit
- *TECU* – Transmission Electronic Control Unit
- *VECU* – Vehicle Electronic Control Unit

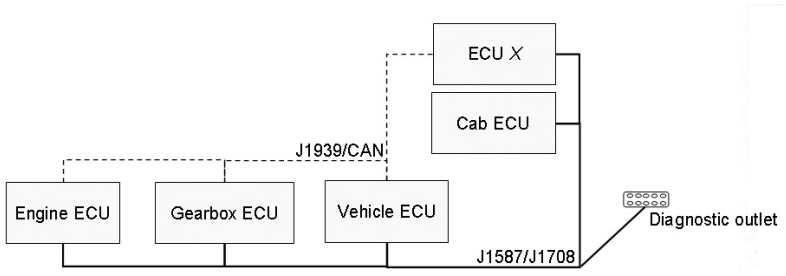


***Figure 4.4**: A schematic overview of the ECUs and the protocols used in the CAN network.*

4.4 Diagnostic Trouble Code - DTC

As previously mentioned, ECUs are highly computerized and contain built-in self-diagnostics capabilities in order to detect problems that affect vehicle performance. When a failure is detected, a DTC is stored in the ECU memory, which holds information related to the specific problem. Certain DTCs are managed by the existing on-board diagnostics system, e.g. illuminate a lamp on the dashboard notifying the driver about occurrence of deviating behavior, while most DTCs require off-board diagnostics systems using the diagnostics connector interface located in vehicle, for further analysis, interpretation, and determination of the kind of action needed for managing the emergence of the problem.

As shown below, a DTC contains detailed information about the occurred failure.

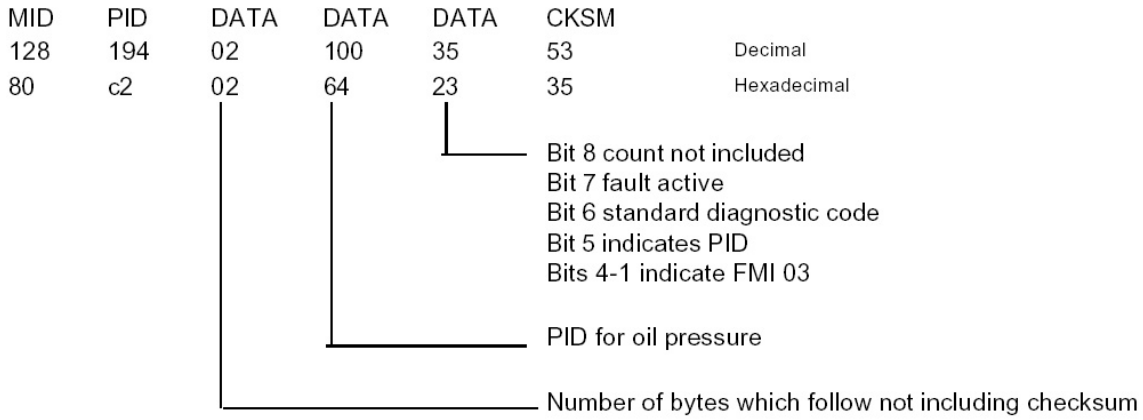


***Figure 4.5**: Example of a DTC data message* (SAE, 1998).

In order to interpret a DTC, it has to be decoded according to the J1708/J1587 protocol specifications (SAE, 1998).

There are two starting frames in a DTC message indicating the related MID and PID. The following data frames contain information associated with the particular fault code whereof the first frame describes the number of following bytes (excluding checksum), the second frame indicates the related PID (Parameter Identifier) or SID (Subsystem Identifier) and the last data frame includes information that further describes the

characteristics of the fault code. Below, we provide a short description of what each bit represents.

- *Bit 8* – Occurrence count included – true or false. Informs whether there is a value included in the DTC which determines the number of times a DTC has occurred.
- *Bit 7* – Current status of the fault code – active or inactive.
- *Bit 6* – Type of DTC – standard or expansion DTC. Indicates whether the value is between 256 and 512, in such case the value needs to be added with 255.
- *Bit 5* – Low character identifier for a standard DTC - PID or SID.
- *Bit 4-1* – Failure mode identifier (FMI) for a standard DTC. Description of the type of failure detected in the subsystem identified by the PID or SID. The description of the type of failure detected is always between the values 0-15, where each value has its own corresponding meaning e.g. value 1 - "Data valid but above normal operational range, i.e. the engine could be overheating."

Using the MID, FMI, PID or SID associated with a DTC, the control system containing the fault and the affected subsystem within the control system can be determined. However, this diagnostic data is not always sufficient to make a correct diagnosis of a vehicle. Sometimes the DTCs may point to the effect and not the actual cause of the problem. It is therefore important to question and further examine the outcome of a DTC (Kuschel & Ljungberg, 2004).

5 Method

Here we provide a brief overview on the methods used and how these have been applied in our study. A method includes everything that has to do with the execution of the study. Which method that is used to collect the data and how the study is executed, depends on the research problem and the purpose of the study (Ejvegård, 1993). Methods works as a means of assistance in dealing with the research area and in a structured way find an answer to the research question. The method that is chosen also depends on what gives the best answer to the research question in relation to the time and the means at disposal.

Methods can generally be divided into quantitative or qualitative methods. A quantitative method implies scientific methods where statistical processing and analytical methods are applied (Patel & Davidsson, 1991). Quantitative methods express the collected data as measurable numbers and tables and the analysis is performed with statistical methods. A qualitative method aims to examine of which character a phenomenon is and how it should be identified (Wallén, 1996). These methods aim to create a deeper understanding compared to quantitative methods. The qualitative method is easily recognized because of its closeness to the research objects (Holme & Solvang, 1997).

The ambition of a study usually depends on the level of knowledge in the research area (Wallén, 1996). The research work is typically classified according to the nature of the research objectives or type of research. In this master thesis work we have adopted an explorative approach towards the research area. An explorative study aims to obtain/reveal basic knowledge about the research problem (ibid.). The research design is characterized by flexibility in order to be sensitive to the unexpected and to discover insight not previously recognized. Exploratory research is appropriate in situations of problem recognition and definition. Once the problem has been clearly defined, exploratory research can be useful in identifying alternative course of action (Patel & Davidsson, 1991).

There are some things that characterize research work dealing with vehicle- electronics and diagnostics. Manufacturers are not keen to let people from the outside get detailed information about the electronic systems in their vehicles. To access the control units inside a vehicle you need special hard- and software. This means that some sort of collaboration with a vehicle manufacturer is almost necessary to conduct this kind of research work. In our case any kind of required hard- and software has been available due to close collaboration with Newmad Technologies and Volvo Parts. At the same time it is also difficult to get access to manufacturers R&D departments and specialist repair shops, e.g. to conduct field studies and interviews. Field studies at these locations help the researcher to get a good understanding for the work practice and requirements of vehicle diagnostics. Such an example is the field study, by Kuschel and Ljungberg (2004) studying the work practice of local service technicians at several repair shops. With this approach they gained an understanding for the main characteristics of diagnostics work. Restrictions to empirical data have been confining our research work and therefore it has been necessary to choose other available methods to collect research data. In this study

we have chosen literature reviews and workshops to collect data and obtain the desired level of knowledge in our research domain.

However, the main focus in our research work has not been to get a deep understanding for diagnostics work practice. Our focus has instead been to design a software architecture based on design principles that enables RVD. The architecture should support several application scenarios of RVD and be suitable for the aftermarket of heavy commercial vehicles. Concurrently with the design each proposed solution has been closely elaborated and implemented to validate the concept and its reliability in real environment. This validation has been performed to get a "proof of concept" ensuring that the architecture works under different circumstances. If "proof of concept" was achieved the architecture could be considered successful, i.e. the architecture fulfilled its intended purpose.

This way of conducting research has a clear connection to the thoughts of "Practical Informatics" (Ljungberg, 1999). Practical Informatics is a practical, design and programming oriented approach to Informatics. In Practical Informatics there is a focus on design, the use and enabling properties of computer technology and less focus on the more abstract and descriptive parts of Informatics (ibid.). In practice, this means that research projects dealing with software development, like this study, should put in more effort on design and development of new services, and evaluation.

The data collection in this study has been done by literature reviews and by organizing workshops with experts in the area. The knowledge obtained, helped us to get an understanding of vehicle diagnostics. In conjunction with the knowledge gained from brainstorming sessions we produced a number of scenarios that exemplify RVD usage (see Appendix A). These scenarios helped us to identify requirements and functionality in order to design and implement our architecture for RVD.

To design and implement our architecture we have used a software engineering process. The explorative nature of this study suited the choice of an evolutionary approach to software engineering. The evolutionary development method of "prototyping" has been used to specify, design, implement and validate our architecture (figure 5.1). Prototyping is based on the idea of developing an initial implementation and then refining it through many versions until an adequate system is achieved (Sommerville, 2001). This iterative process means that several prototypes have been developed, each one with its own separate objective. A common factor has been to derivate the accomplishment of "proof of concept" by carrying through validation of each built prototype. Validation made it possible to analyze the outcome of each iteration. By doing so we have been able to determine what parts to be kept, thrown away or modified in our prototypes in order to meet the requirements of our architectural design.

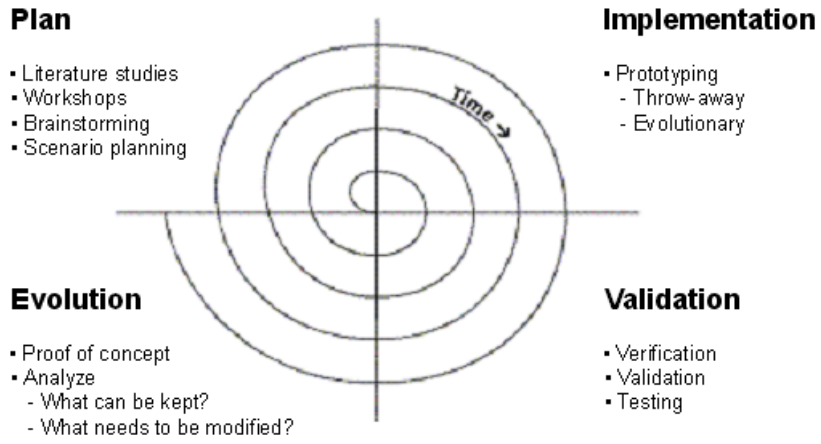


***Figure 5.1**: Evolutionary development using prototyping with activities.*

5.1 Literature Studies

To get an initial understanding for the problem domain available literature has to be reviewed. The literature provides an overview of the research area, knowledge about what has been done and the different views that exist in the research area. Literature studies help to point out lack of information within a certain area of knowledge; thus giving relevance to the research problem and providing knowledge for defining a meaningful scientific research question. Literature studies also describe how prior studies have been planned, how data has been processed, and how the results of different methods are interpreted.

The domain of RVD is an area where the literature is somewhat limited. This depends on RVD being a relatively new application domain where the body of knowledge is narrow. Fortunately, RVD has several aspects in common with other areas such as vehicle electronics and wireless communications. In these areas the body of knowledge is much greater and therefore we have included this literature in our review in order to obtain knowledge about vehicle electronics, and wireless communication technologies. The iterative process used in this study means that requirements have changed between iterations. Those changes in requirements have meant that complementary information has been required continually during this study, i.e. literature review has been an important part throughout the study.

5.2 Workshops

To further strengthen the knowledge in our research area, recurrent workshops with specialists (in the field) have been conducted. This has been done to fill gaps in the literature, to discuss different ideas, problems and solutions and not least to regularly analyze and discuss the outcome of each prototype. The participants in the workshops have been the master thesis students, personnel from Newmad Technologies and Volvo Parts, as well as a researcher from the IT University of Göteborg. The personnel involved in each workshop depended on the specific purpose of the workshop, e.g. if it was a

strategic, theoretical or practical workshop. The number of participants in the workshops varied from five to seven persons.

The first workshop to be held was a strategic one, where the participating parts together outlined the overall purpose and aim of this master thesis work. In that workshop all the above parties participated including the person in charge of strategic development and a system product owner from Volvo Parts. Newmad Technologies contributed with a specialist in the RVD field, who naturally would act as the head of development. The researcher from the IT University played an important part of the strategic planning, providing theoretical and practical knowledge about RVD and research work in general. This knowledge was of great value, when planning the continued course of action. The outcome from the strategic workshop was a detailed plan on how to conduct and pursue with this study.

In the initial part of this master thesis work we held a couple of theoretical workshops to obtain essential knowledge in the research domain. In these workshops we met with the head of development and the researcher to discuss technical aspects, existing RVD solutions and the characteristics of the commercial vehicle aftermarket. A theoretical workshop was also used to discuss the scenarios of RVD constructed in this study. Discussions were held concerning which scenarios to implement in the prototypes as well as required changes and additions to the proposed scenarios.

Later workshops had a more practical orientation, where different ideas, technical issues, problems and solutions were discussed. The participants in these workshops were the same as those in the theoretical workshops. In these workshops the master thesis group presented the development work carried out since the last workshop. The expert from Newmad Technologies and the researcher gave their view on the recent work, which led to discussions on how to pursue the work in the best possible way to meet the upcoming thesis objectives. The upcoming objectives were often linked to the purpose of the next prototype to be implemented and validated. Practical workshops were planned regularly during this study and were combined with a pre-determined objective, e.g. validation of a specific prototype.

The limited number of participating parties could have affected the outcome of the workshops. With a small number of participants the danger exists to get a narrow-minded view on RVD and surrounding fields, i.e. other relevant people's opinions are excluded. Further, the participants represented Newmad Technologies and Volvo Parts. Therefore, their view on RVD can be the one that serves their companies interests. It has been important to be aware of this risk factor since our intention has not been to design RVD architecture to some specific brand's vehicles/aftermarket. Literature reviews together with ability to nuance the industrial participant's view on RVD with the researchers´ view on the same area, worked as a method to overcome problems with limited number of participants.

5.3 Scenario Planning

Scenario planning can be important as a means of assistance to support the learning process for groups or individuals, by providing a framework for open-ended discussions about the problem domain (Ringland, 1998). Scenario planning means describing possible future fields of application, i.e. in our context for RVD, which are related to the existing reality. Porter (1985) defines scenarios as:

"An internally consistent view of what the future might turn out to be – not a forecast, but one possible future outcome."

In this study we have applied the method of scenario planning by describing different scenarios where RVD can be used and are beneficial to the commercial vehicle aftermarket. To do this we have used our collected data through literature reviews and workshops in combination with brainstorming to prepare several scenarios of RVD. Brainstorming is a method used to quickly come up with new ideas and proposals.

Constructing scenarios is difficult since the human view of the future is limited by the human mindset (Hodgson, 2003). Mindset is our perception on the outside world and this perception is sustained through our belief system. This pattern is very difficult to break and a process is needed to help us escape the domination of our mindset. As mentioned earlier this is accomplished using literature reviews and workshops. With an open mind towards our research area it has been possible to write "stories" of RVD.

As mentioned above we prepared several different scenarios of RVD (see chapter 6). In order to check the significance and integrity of each scenario we used further workshops to discuss these matters. These workshops lead to modifications and adjustments to some scenarios whereas others were discarded. In these workshops we also grouped the scenarios by similarity, i.e. scenarios which had common aspects and similar field of appliance were grouped together. This process helped us to select which scenarios to design for and implement. This selection was also made from an importance and time consuming point of view.

5.4 Design and Implementation

To design and implement our architecture for RVD we have used a software engineering process. There are many different software engineering processes to choose from but there are some activities which are common to all software processes. According to Sommerville (2001) these are:

1. *Software specification* – The functionality of the software and constraints on its operation must be defined.
2. *Software design and implementation* – The software to meet the specification must be produced.
3. *Software validation* – The software must be validated to ensure that it does what the customer wants.
4. *Software evolution* – The software must evolve to meet changing customer needs.

In this study we have chosen an evolutionary development process. The evolutionary approach to software development is widely applied to system development (Sommerville, 2001), which we are dealing with in this study. The characteristics of this approach are that the activities of specification, development and validation are interleaved. An initial version of the system is quickly developed and is then refined through valuable input to produce a system that satisfies the customer. The main benefits of an evolutionary approach are the effective way of producing a system that meets the immediate needs, and the ability to incrementally develop the system specification. As the development work advances a better understanding of the problem is established, thereby contributing to an increased quality of the prototype. These characteristics are important reasons to why we have chosen this approach. As we have developed a better understanding for our research domain, we have easily been able to implement this knowledge into our prototypes.

5.4.1 Prototyping

One approach to evolutionary development is to use the method of prototyping. A prototype is used to demonstrate concepts and try out different design options (Sommerville, 2001). Prototyping is also used to find areas of strength and weakness in the software. Using a prototype it is easier to reveal errors and omissions in the requirements and then modify these to reflect this new understanding of the requirements. The use of a prototype usually leads to improvements in the system specification (Ince & Hekmatpour, 1987).

The objectives of the prototype should be made explicit already when the prototyping process is started (Sommerville, 2001). When the objectives are set you need to decide what to put into and what to leave out of the prototype system. In our case that was represented by the planned scenarios, which of those to include and which to leave out of the prototype. Prototyping is an iterative process and as mentioned in the beginning of this chapter several prototypes have been developed, each one with its own set of objectives. The first two prototypes were mainly developed to find out more about the requirements and to get an understanding for those requirements that were not well understood. From the third iteration and onward we began the implementation of our scenarios. The last sets of prototypes were mainly used to validate our architecture and thereafter make necessary changes to the prototypes.

There are two different approaches to the development of a software system using prototyping (figure 5.2). These approaches are the evolutionary- and the “throw-away” approach to prototyping. There is an important difference regarding the objective of these two approaches. In the evolutionary approach the objective is to deliver a working system. The objective of “throw-away” prototyping is instead to validate the prototype in order to improve the system specification.

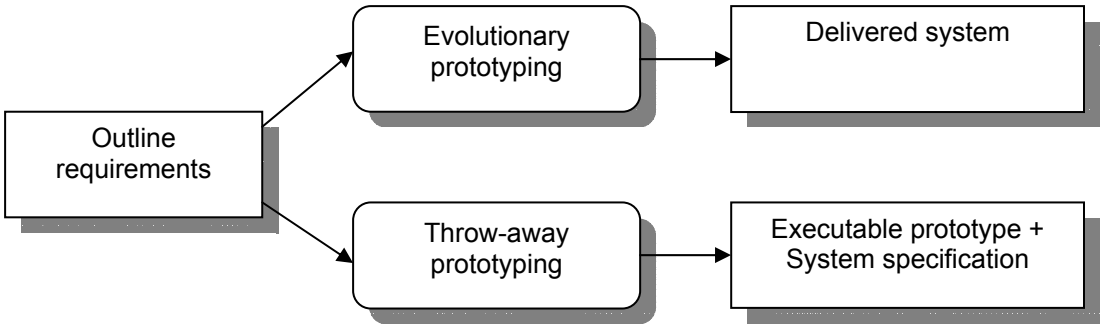


***Figure 5.2**: Approaches and objectives of prototype development* (Sommerville, 2001).

In system development, there is often a need to use different approaches for different parts of the system. This has also been the case in our appliance of the prototyping process; the selected approach depended on the aim and purpose of the iteration.

5.4.1.1 Throw-away Prototyping

A "throw-away" prototype is written, evaluated and modified in order to refine and clarify the system specification. The evaluation of the prototype helps the developers to modify the system specification until its functionality is satisfactory. Once the evaluation of the prototype is finished, the prototype is "thrown away" and will not be used as a basis for further system development. When the system specification is complete it is derived from the prototype and the system is then implemented to a final version using a phased software process model (Sommerville, 2001).

When developing a "throw-away" prototype the first requirements to be implemented are the ones that are not fully understood. This is because you need to find out more about those requirements. Requirements that are well understood may be left out and never implemented. Characteristics as poor system performance and reliability may be acceptable because the prototype is discarded after evaluation (Sommerville, 2001). The important thing is that the prototype fulfils its principal function of helping the developers with understanding the requirements.

This approach to prototyping was used during the first two iterations, i.e. the prototypes built in these iterations were discarded. As mentioned earlier the main objectives with these prototypes were to get an understanding for the requirements that were not fully understood. This means that the first prototype was built to get an understanding for how to read parameter data and DTCs from a simulated vehicle. These requirements were essential and therefore very important to understand properly from the beginning of this study. The second prototype was built to test the wireless transmission of data, back and forth from the vehicle. These were also important requirements that needed to be understood from the beginning in order to eventually implement a full scale prototype system. The expression "throw-away" is a bit misleading because some components of a "throw-away" prototype can be reused. This was done by reusing some key components in later developed prototypes. With this new understanding for requirements that

(previously) were poorly understood, we were able to move on to implement our scenarios, which were made using an evolutionary prototyping approach.

5.4.1.2 Evolutionary Prototyping

Evolutionary prototyping is based on the idea of developing an initial system, exposing it to user comment and refining it through multiple stages as the requirements become clearer until a final deliverable system is developed (Sommerville, 2001). When using this approach it is suitable to start with a simple system containing the requirements which are best understood and which are given highest priority. This initial prototype is then augmented and modified as requirements are changed or new ones are discovered. The aim of evolutionary prototyping is to reach a point where the prototype system becomes the final product.

From the third iteration and onwards we used this approach when building our prototype system, i.e. we developed an initial prototype which was validated and modified through a couple of stages to finally become the requested prototype. The initial evolutionary prototype was the first one that was built to support our planned scenarios of RVD. In this prototype we implemented all scenarios that had been selected for implementation. During this phase the knowledge gained from the two primary "throw-away" prototypes was used in order to implement the scenarios. The functionality supported in this prototype was reading of parameter data, logging of parameter data and notification about any occurred DTCs in a vehicle. This prototype was built to function in conjunction with our simulator environment.

In the remaining iterations we modified the initial prototype as some requirements become clearer. The main aim of the latter iterations was to refine the prototype system and make it work in a test bed environment and later on in real environment. The final prototype was therefore a prototype which supported our scenarios of RVD and worked in real environment, i.e. the prototype supported RVD on a vehicle in operation independent of wireless communication technology.

A problem with the evolutionary approach is that continual change can lead to a corrupt structure of the prototype system, which in turn means a relatively short lifetime to the system due to increasing maintenance problems. However, the intention of this study has never been to deliver a final and complete RVD system. The intention has been to design, implement and validate the architecture to derivate the accomplishment of "proof of concept". Therefore, the final RVD prototype can be regarded as a "throw-away" prototype to be evaluated and be used as a mean(s) of assistance in the process of creating a detailed specification of a RVD system intended for the commercial vehicle aftermarket. This final prototype can be regarded as the practical contribution regarding our work.

5.5 Validation

Software validation or, more generally, verification and validation are performed to ensure that the software follows its specification and meets the customer's expectation of the system (Sommerville, 2001). Verification of software is performed to check for conformity to its specification and validation to prove that the software is suitable for its intended purpose. The process of verification and validation is carried out by doing reviews, code inspections and software testing.

Validation of an evolutionary prototype is difficult since there is no detailed system specification prior to implementation. When using evolutionary prototyping the verification and validation is mostly concerned with checking if the system is adequate, i.e. if it is good enough for its intended purpose (Sommerville, 2001). In this study the validation has mainly been carried out by performing different testing on the developed prototypes. We have used three different test environments in this study, these are:

- *Simulator environment* – Hard- and software to simulate a vehicle and WLAN to communicate between architecture components.
- *Test bed environment* – Using a test-rig at Volvo Parts to simulate a vehicle and GPRS to communicate between architecture components.
- *Real environment* – Using a real operational vehicle to provide test data (e.g. generating DTCs, parameter readings, etc.) and communication independent of wireless technology.

Testing can be performed on different levels of the software, spanning from single unit testing to final acceptance testing. The first levels of testing are usually performed by the programmer responsible for the component, i.e. the programmers test the code as it is developed. This has been adopted in this study where the programmers had the responsibility to review and test each component. The later stages of testing included the task of integrate several developed components. This has been done using system testing, where our sub-systems have been integrated to make up the whole architecture for RVD. The system testing was done in simulator- and test bed environments. This type of testing is mostly concerned with finding errors due to unanticipated interaction between sub-systems and sub-system interface problems. The outcome from this testing was a validation, that the prototype fulfilled its intended purpose in a test bed environment.

Acceptance testing is the final stage in the testing process and include customer supplied test data rather than simulated test data. In our case acceptance testing can be seen as performing the final test in real environment using an operational heavy vehicle. Acceptance testing can help to reveal problem and errors in the requirements or unacceptable performance of the prototype. This can be done because a real test environment "exercises" the system in a different way and can therefore reveal previously hidden problems. With the final acceptance testing performed we could validate, that the prototype worked as supposed in real environment and that it fulfilled its intended purpose of supporting different scenarios of RVD.

6 Scenarios

During the master thesis work we produced a number of scenarios that exemplify different use-cases for RVD. These scenarios also identify the actors, the information exchange between them and what role they play in each scenario. With this information in hand, one also gets a deeper knowledge of what kind of technical demands that the software architecture must be able to deal with. Furthermore, the scenarios illustrate important key features of the proposed software architecture.

The scenarios are divided into three main categories:

- Notification
- Periodic
- Real-time

By having constant or periodic control over vehicle parameters, and the ability to receive notifications whenever vehicle abnormalities occur, the service technician can take proper actions based on the collected data. This might involve informing the driver about scheduled maintenance or sending collected vehicle data to the nearest repair shop in range of the vehicle, and prepare for the vehicle's arrival in an efficient way. If it is a critical vehicle malfunction the service technician can inform the driver to await roadside assistance in place.

6.1 Notification

This category is based on the notion that the service technician responsible for a vehicle receives a notification whenever the vehicle shows a deviating behavior. Notifications can either be initiated by the driver or by a monitoring system inside the vehicle. The system runs in the background, transparent to the driver, responsible for observing vehicle performance and the discovery of any potential deficiencies. The driver should not need to be aware of any ongoing activity performed by the monitoring system. It is also desirable that all deficiencies are discovered before they become apparent to the driver. A driver can of course discover abnormal vehicle behavior and take actions, but by then the problem most often has already advanced into a serious problem, that would require maintenance of the vehicle in a repair shop.

In order to promote a closer relationship with the customer (e.g. the driver), the driver should always be given the opportunity to make a phone call to the repair shop whenever he or she feels that something is wrong with the vehicle, even though the monitoring system has not detected any abnormalities. A problem experienced by the customer might in fact escape detection by any diagnostics application. This takes in consideration the importance of a person-to-person correspondence, which further could strengthen customer relations and facilitate diagnostics examination.

Notifications sent from a vehicle should appear on the end-user application window alerting the service technician that an event has occurred. The notification will not

disappear until a service technician responsible for the specific vehicle has acknowledged the alerting message.

The monitoring system inside the vehicle generates three different types of notifications:

- *Status* – Informs the remote server of connection availability and existing communication technologies.
- *Critical* – DTC transmitted from the vehicle.
- *Semi-critical* – Generated by a parameter values, that are outside predefined limits or intervals, explicitly specified by a service technician. Can be considered as an indication of a potential DTC.

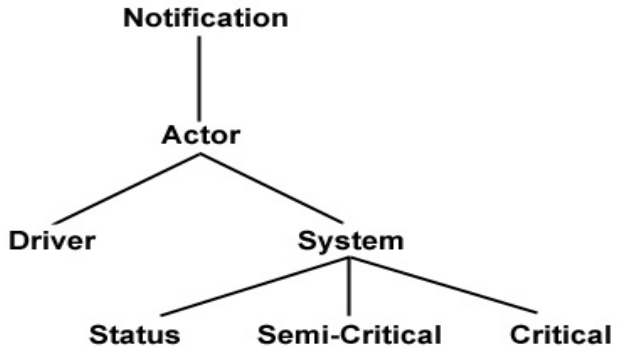


***Figure 6.1**: The relation between different cases of notification.*

6.2 Periodic

These scenarios describe the supervision, or logging, of vehicle parameter data, during a specified amount of time.

Periodic logs are initiated by a service technician responsible for supervision of the vehicles. A service technician can, based on professional knowledge, determine the amount of log data needed for analysis. For instance, the service technician might be interested in performing a log during 20 minutes, or store snapshot values triggered by certain parameter values in between certain limits. The interval or frequency of a periodic log could also be defined by the customer. Then the periodic logging could be offered as a service, where the cost is based on frequency and amount of periodic log data.

The reason for initiating a periodic logging of vehicle parameters might be the recurring notifications of DTCs, or if the driver informs the repair shop about suspect vehicle behavior. If the provided information indicates flaws within a certain part of the engine, the service technician may want to start a periodic logging of one or several parameters possibly associated with the problem. Given the results from the periodic log, the service technician can see how the parameters operate in correlation to each other and find abnormal patterns. This information could give further explanation to the problem and what needs to be done. Periodic supervision most often means that data is stored for analysis at a later occasion.

A service technician should be able to specify the periodic log with properties such as:

- *Total time* – Duration of the periodic log.
- *Interval* – How often the vehicle data should be sent to the client application.
- *Frequency* – Is by which frequency the monitoring system should perform readings of the on-board ECU.
- *Rules* – The monitoring system will only transmit parameter data which is in between the range of the specified limits, or in the specified state.
    - *Limit* – Parameter data in between certain values.
    - *State* – Indicates whether a parameter is in "on" or "off" mode.

By applying rules to the periodic log, such as the predefinition of limits, a service technician can use his or hers individual experience to specify semi-critical parameter values. This can be seen as an ability to utilize prognostic examination, in order to prevent incipient faults.

6.3 Real-time

Supervision in real-time provide interactive management of vehicle parameters. A service technician's modification of vehicle parameter settings should bring instant feedback of the resulting effect and analysis of potential problems takes place instantaneously. The real-time scenarios might also concern remote update of ECU software.

The service technician might perform real-time parameter readings when the supervision does not require extensive logging in order to investigate a problem. For example if the service technician receive notifications regarding problems with the engine cooling system, a real-time supervision of the relevant parameters associated with that system, might be sufficient in order to find the cause. The reading could detect abnormal data values of the investigated parameters indicating the defective components. If the provided information fails to describe the cause of the problem, the expert/service technician might decide to initiate a periodic log for further examination.

7 Design Principles

Literature reviews and workshops with experts helped us obtain the desired level of knowledge in the research area. Collected knowledge together with brainstorming made it possible to prepare several application scenarios of RVD. These scenarios describe situations where RVD can be applied in the heavy commercial vehicle aftermarket. The scenarios helped us to identify important features and several requirements of the architecture. This provided us with a good understanding of the requirements placed on a suitable RVD architecture intended for the commercial vehicle aftermarket. With this understanding, we could design an architecture that meets the technical requirements of RVD as well as the identified characteristics of the commercial vehicle aftermarket, e.g. we have thoroughly examined the distribution of system components and considered the economic prerequisites in the aftermarket area. From our architectural design we introduce a couple of findings, which we present as our design principles. These design principles has been selected because they highlight features and requirements placed on the architectural design. They are considered as central to RVD in combination with the commercial vehicle aftermarket. This means that some design decisions made, common to software architecture design in general are not introduced as design principles, even if they are an important part of the architecture. In this study, we argue for the importance to follow these design principles to achieve a suitable RVD architecture intended for the commercial vehicle aftermarket. The design principles are regarded as the theoretical contribution of this study and this chapter defines and explains our design principles.

When designing a system one needs to concern about both internal as well as external software quality factors (Fitzpatrick, 2003). External quality factors are those that can be observed by the user (e.g. latency, reliability, correctness, etc.) whereas internal quality factors are those which lead to a qualitative design from a technical perspective. To achieve these qualities one need to have a good set of carefully prepared design concepts and principles. These concepts and principles should help developers to answer questions like what criteria should be used to partition software into components, and how to separate function or data structure from conceptual representation. One of the fundamental software design concepts that have an important role during this project, especially to provide information and to help isolating key factors of the problem domain, is the software architecture design concept. There are many different definitions of what software architecture really means and what it stands for. The definition given here is the one normally used by software engineers (Bass et al., 2003):

> "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them."

An architecture model defines the way in which the components of a system interact with one another and the way in which they are mapped onto an underlying network of computers. The overall purpose of our proposed model is to ensure that the structure will meet current and future demands. It is important to realize the significance of software architecture and its impact on the whole system. The chosen architecture will definitely

determine many of the qualities of the system. Dealing with software architecture we focus on a set of properties that we believe are important for RVD systems and should be specified during the architecture design. Shaw & Garlan (1995) defines these set of properties as:

- *Structural properties* – This aspect of the architecture design defines the components of the system (e.g. modules, objects, filters, processes, databases, etc.). The divergent components will be separated but similar ones will be located into belonging packages. The structural properties will also determine the manner in which the interaction between components should take place.

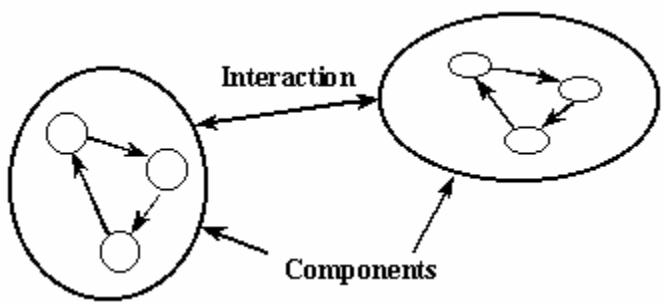


Figure 7.1*: Component separation and interaction.*

- *Extra-functional properties* – Software architecture must not only concern about functional but also extra functional properties. While former handles the specification, adaptation, analysis and the connection between components, the latter describes how the architecture should address the problem of achieving requirements for performance, reliability, scalability, security and other system characteristics.
- *Families of related systems* – The architecture design should follow well known design patterns which are commonly encountered in the design of the architectural building blocks.

As presented earlier (see chapter 5) the method used during this project, i.e. explorative prototyping, is based on iterations. This property is not overlooked when dealing with design processes. Based on the outcome from each of the iterations in conjunction with the facts presented here, and the important influence of dealing with "aftermarket" business solutions in conjunction with wireless technologies, we managed to isolate some key factors of the delimited problem domain which compose our design principles:

- Distributed Architecture
- Data Replication
- Vehicle Client Data Cashing
- Carrier Independency
- Filtering
- Resource-Effectiveness

These design principles have laid the foundation of the final result of the project, e.g. handling the complexity of wireless communication technologies, the system model and the logical as well as the physical separation of each part of the software architecture.

7.1 Distributed Architecture

As previously mentioned the chosen architecture has an important impact on the outcome of the final system. Therefore it is important to realize, understand and consider not only current but also the future requirements of the system and its adaptation to flexibility, scalability and maintenance. As a system developer one need to be aware of the fact that there is an increased expectation and need to develop more and more complex systems that will meet different requirements, such as the ability to address ever-changing business requirements, capable of delivering timely information, consider the economical perspectives and being able to cope with multiple users and different kind of work-practices. Consequently, the architecture selection and the type of solution will be the "supporting pillar" of the final outcome and the quality of the system.

The selection of the most suitable architecture for our project was made by analyzing different kind of models and at the same time trying to answer some important questions that needed to be under consideration.

- What are the limitations of a specific architecture?
- Will the business logic and business data be available everywhere at all time?
- Will it be cost-effective when upgrading software?
- How will it affect the maintainability?
- What are the prospects of architectures adaptability to different kind of work-practice?

The result of the research and analysis, which mainly were surrounding the questions given above, debouched into selection of distributed architecture based on layered and multi-tier client/server architecture. These are well known architecture properties used by many organizations to meet similar kind of requirements such as those introduced for RVD software architecture.

Distributed architecture as design principle fulfills the properties of software architecture that are introduced earlier as suggestions made by Shaw & Garlan (1995). The layered architecture follows the structural property required for software architecture by partitioning the software into independent components and determining the interaction between them, thus providing modularity, facilitating maintenance, and keeping the system flexible towards new requirements such as subtracting or adding new components to the system. This is of major importance for a RVD software architecture where new ideas and services are still to arise in a "state of the art" domain such as RVD. New customer services might be recognized as time goes by and the system itself should not be an impediment for any kind of new additions to the system, such as new components or even subsystems.

Multi-tier client/server (properties) provides extra functional properties by not only managing the partitioning of the software but also focus on the physical location of each partition, the ability to replicate system resources (hardware as well as software), providing security in the sense of abstraction, and of course scalability by allowing the distribution of system loads across multiple servers. The importance of multi-tier client/server properties for RVD system is related to the (extreme) mobility of each vehicle and the different work practice of each service technician and even experts. For example the different work practice of each service technician can be described as different kind of mobility, such as *wandering* and *visiting* (Kristoffersen & Ljungberg, 1999). Regardless of the type of work practice information need to be made available to authorized users.

7.1.1 Layered Architecture

Layered architecture is desirable when dealing with complex systems. By using layered architecture we managed to divide the complexity of the system into separate components and order each one in a layered manner. Each component has been carefully adapted to a specific responsibility and also delimitated in a desired manner from the rest of the system. Due to the chosen solution, interfaces have to be provided for communication between components. The structural properties provided by the layered architecture facilitate the subversion of complex problems into smaller and more apprehensible fractions and smoothes the progress of additions and modifications to the system. Consequently, new applications or diagnostic services can easily be added to the current system since the layered architecture facilitates integration of new components and subsystems.

We base our solution on a layered architecture, mainly because of its great capability to support both present and future requirements of RVD. It meets requirements such as the differences between a centralized and decentralized work practice, additions of new components and subsystems, and the possibility to distribute complexity, both in regards to system logic and resource-effectiveness. During the formulation of our research question we described current problems with similar telematics systems. We manage to overcome many of these problems by providing some important features (outlined with a brief description below) that together contribute to a suitable RVD architecture.

- *Modularity* – We partition the software into different components where each satisfies a particular problem requirement, e.g. the client application, application server, the vehicle client application, etc. The information belonging to a module will not be available to others more than the provided information through specific interfaces.
- *Scalability* – The modularity of each layer makes it easy to add new components to the current system without any major problems or side-effects to the whole system. New components need only to provide interfaces to other components and be aware of the interfaces provided by others. This property of layered architecture implies the ease of development of new subsystems and the ability to interact with external systems. For example there would be no problem to

integrate new kind of services where close communication would be needed with application server and database server. The rest of the system would not be affected and will most certainly be able to coexist with the new added services.

- *Distribution* – Our architecture provide the possibility to locate each layer at different physical places. This is an important property especially for a distributed system such as RVD where vehicles are extremely mobile and the service technicians and the experts' work practice are normally pretty variable.
- *Portability* – The layers can be used to provide portability for software systems that need to be able to run on other hardware platforms, e.g. the application server layer can be run on UNIX whereas the client application can be implemented on different platform such as Microsoft Windows.

7.1.2 Multi-tier Client/Server Architecture

The multi-tier architecture is a mature and well accepted architecture which has been in use both for commercial and military purpose since the 1990s (SEI, 2004); thus following into the category described by Shaw & Garlan (1995) as families of related systems. Most importantly it fulfills the demands of typical distributed software architecture requirement as the one needed for RVD system.

A multi-tier architecture is primarily defined by three component layers; the front-end component which provides portable presentation logic. A client application layer corresponds to the front-end component. The back-end component which provides access to dedicated services. A Database Server layer corresponds to the back-end component, and finally the middle-tier component which provides sharing of resources and controls business logic by isolating it from the actual application. An application Server layer corresponds to the middle-tier component. The middle-tier is normally divided into several units with different responsibly. This is desirable when the gap between the front-end component and the middle-tier is too big. Figure 7.2 illustrates the multi-tier architecture.

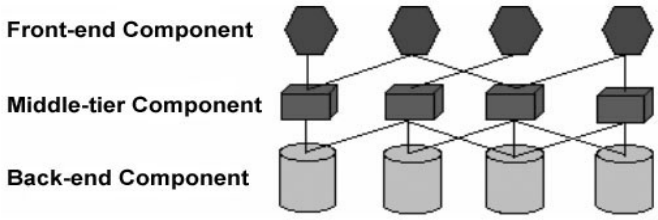


***Figure 7.2**: Tiers of the Multilayer client/server architecture.*

Disregarded from the front-end, middle-tier, and back-end components described here there is also a vehicle component in the proposed RVD system which is responsible for managing any kind of communication with the vehicle itself.

The following is some important features gained using multi-tier architecture:

- High Encapsulation
- Performance
- Scalability
- Availability
- Thin Clients
- Thin Vehicle Client

The high encapsulation property forces each client to only being able to invoke services and/or methods, thus also facilitates software development by allowing each tier to be built on separate platform if needed. This property implies that different tiers can be developed in different languages, e.g. in the RVD system the front-end component can be developed in Flash MX, the middle-tier in Java and/or Visual C++, etc.

The performance property is achieved by dealing with both network and processing performance. The network bottleneck is kept to a minimum by only sending request and response over the network – which also is an implication of cost-effectiveness over wireless communication when payment is done per sent packets. This is valid not only for communication between the client application and the application server but also communication between application server and vehicle client. The processing performance is achieved by dismissing complex execution from client application and vehicle client to application server.

The nature of multi-tier architecture allows distribution of loads across multiple servers. This scalability factor of multi-tier architecture is valid both for middle-tier component as well as the back-end component. This is an important property which makes it possible to provide availability. One will be able to replicate both application servers as well as database servers for keeping integrity and consistency and provide information to users no matter location (centralized, decentralized, etc.).

The possibility to provide thin clients as well as thin vehicle clients for RVD system is an important prospect. The thin client property will only be responsible of providing an interface between the end user and the rest of the system. The application logic will be separated from business logic and functional rules. For example, the thin client can be effectively combined with a web application, thus the application client can be executed regardless of the kind of operative system or platform (e.g. PC, laptop, PDA, etc.) as long as it has an integrated web browser. Following this manner, the client application will meet different kind of work-practices. The use of thin client will in some cases also implicate cost-effectiveness. Imagine having hundreds of “rich” clients and there is a need for upgrading software to new version. Worst case scenario can be the need of mass production of CDs with the upgraded software that needs to be distributed all over places. These kinds of problem are all solved by the use of thin clients, where all changes are made at application server. The importance of thin vehicle client is based on being able to reduce any requirement for addition and/or expensive hardware. Using thin vehicle client any kind of heavy execution and calculation need to be managed externally, e.g. the application server being responsible for such processing.

One possible disadvantage with this approach might be increased network traffic, e.g. Vehicle Client transmitting huge amount of redundant or irrelevant information. Yet, a compromise has to be made between using thin or rich client, where the importance of a thin cost-effective vehicle client (which is appropriate for the commercial vehicle aftermarket) surpasses the benefits gained from having a rich client with high performance and computing power.

7.2 Data Replication

Data replication is the process of making copy of actual data. The data available at one place will be replicated to a location separated from the origin position. In a mobile environment such as the one related to vehicles, replication is absolutely required in order for non-connected users to access common information. This type of mobility, i.e. a vehicle moving from one location to another, is denominated as "traveling" in Kristoffersens & Ljungbergs conceptual framework for mobility (1999). Normally data replication is based on scenarios where mobile users need to be able to access common information available on for example a mainframe server. However, in our case the situation is reversed, since the mobile vehicles are the source of common information and need to be accessed by service technicians.

In research articles, such as Agrawal & Abbadi (1990) and Hsiao & DeWitt (1991), data replication is described as the solution for data availability. In our case the point of view regarding data replication is not restricted to only affecting data availability. We believe that using data replication addition important benefits, such as performance, cost-effectiveness and data preservation, will be gained, especially when conducting software architecture for a RVD system.

7.2.1 Data Availability

Our definition of data availability is the process of making vital vehicle information (such as DTCs) accessible to authorized users, even though the vehicle is out of network coverage. This is an important aspect to consider due to a variety of contributing factors preventing a peer-to-peer communication with the vehicle. The importance of supporting data availability is based on resolving connection interruption difficulties introduced by the usage of wireless communication and its related shortcomings. Connection interruption is a major problem with wireless communication (Watson, 1994). Due to the common emergence of such problem in wireless environment, connection interruption should not be thought of as a failure but as a challenge that always needs to be under consideration when designing for wireless system. It might be caused by network coverage/scope, network load, discovery of new "colliding" wireless communication bearer, and the occurrence of downtime (e.g. the RVD application server or RVD vehicle client crashing or vehicle being switched off). In some cases the connection might not be totally interrupted, i.e. communication is still possible but not at optimal level. This kind of connection issue is normally referred to as "weak connectivity" and is the result of intermittent communication (i.e. loosing and regaining network connectivity), low-bandwidth or high latency (Conan et al., 2002).

The problem introduced by connection interruption can be resolved using data replication to provide data availability. By replicating data to locations with solid and reliable communication and storage capacity, data will be reachable for authorized users regardless of their location and/or work-practices. For example vital vehicle information, such as DTCs, need to be replicated in such way that it will be available at all time, and preferably at any location, to users such as service technicians and experts. This can be accomplished using data replication, thus making vital information available at all time to authorized users. Consider for example a situation where an expert needs to retrieve all occurred DTCs on a specific vehicle or a model of vehicles. Further, imagine that the vehicle is out of reach due to either absence of network coverage, vehicle shutdown, or intermittent connection (figure 7.3).

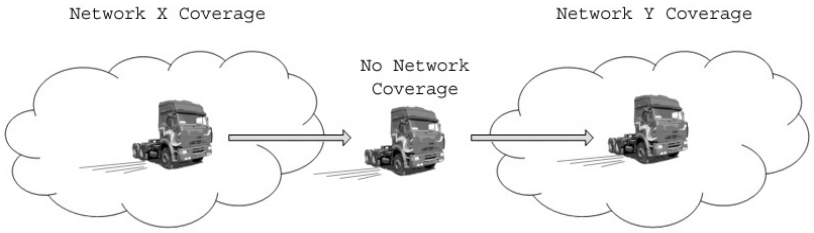


***Figure 7.3**: Vehicle shutdown or loss of connection.*

Instead of trying to retrieve all DTCs directly from vehicle, which might be unavailable for communication, the expert queries the diagnostic database where the data is permanently stored and always accessible. All DTCs stored in the database will correspond to a historical as well as currently active DTCs related to the specific vehicle. Even though vehicles might be out of reach for the moment, most recent data is still available to users (this principle is based on notification properties described earlier in chapter 6.1).

7.2.2 Performance

Replicating data to location with solid storage capacity, reliable communication link and high availability will prominently improve the performance. Performance is normally degraded by "latency" which is most often a side-effect of the perpetual connection interruption or the so called weak connectivity problem. Latency is normally described as the round-trip time of a data packet (Watson, 1994). The round-trip is dependent of the so called elapsed time, i.e. the sum of connection time and transmission time. The former indicates the time consumed for the Vehicle Client to connect to the Remote Server and receive the first byte, whereas the latter means the time consumed between receiving the fist byte and the last one. Latency can also be measured based on only the time it takes for a packet of data to travel from source to destination.

The problem with latency, thus also performance degradation, does not only affect the system itself but also the user of the system, e.g. the users' perception of the systems availability and performance. Using data replication a prominently improved system

performance will reduce and/or hide latency problem, thus making communication issues totally transparent to the user.

7.2.3 Cost-effectiveness

Data replication will also provide cost-effectiveness which is obtained by data being sent (replicated) only once from the vehicle to a solid and persistent storage location. Information obtained from vehicles will/can be used for future incoming inquiry initiated by service technician and/or experts. Using such behavior the application server will be totally relieved from any kind of preparation of new inquiries intended for vehicle Client. Most recent information will always be available, at a local persistent storage, to any authorized user. Avoiding redundancy in such manner implicates avoidance of unnecessary use of the billing system belonging to the currently used wireless communication system between the application server and the vehicle client. This unnecessary cost would have been added to the total expenses if redundant inquiries where obligated. Yet again, these issues can be avoided by replicating data to a solid and persistent location with high availability and reliable communication properties.

7.2.4 Data Preservation

Data preservation property enables statistical evaluation as well as historical prospect of a specific vehicle or model of vehicles. This property can for example be used in conjunction with data warehouse and data-mining to maintain, discover and provide new kind of services based on future needs and/or to meet ever-changing business rules. Analysis of large amounts of preserved vehicle data might reveal frequently occurring problems with specific vehicle models and/or flawing components. Such data/information can for example be directly forwarded to the vehicle manufacturer's production or R&D-departments.

7.3 Vehicle Client Data Caching

There are many issues that need to be under consideration when working with wireless communication. As previously mentioned the most common shortcoming problem with wireless network is either connection interruption or the so called intermittent communication. Irrespective of shortcomings related to wireless communication some additional impediments are introduced due to vehicles importance, as actors, in the overall system. One major problem might be the sudden shutdown of a vehicle during request execution (i.e. a request generated by the application server and obtained and scheduled by the vehicle client), or during response transmission (i.e. the vehicle client transmitting result of a request generated earlier by the application server). The former problem does not only introduce additional latency but, even more importantly, it will be the contributory cause to the loss of vital information gathered and adapted for the application server by the vehicle client. Imagine the vehicle client processing a scheduled periodic log request. The total time of the periodic log might be 5 hours. During this period the vehicle might be switched off several times. If no mechanisms are introduced to take care of such problem the information about the request itself with its

corresponding response prepared by the Remote Agent, during the period until the vehicle is switched off, will be permanently lost. The latter problem, i.e. a vehicle being switched of during response transmission, will be affected in the same way as the one described earlier. The additional problem here concerns the "corrupted" data received by the application server.

Our suggested solution to these problems is to use caching. The problem with lost information about the actual request and the corresponding data is resolved by temporary caching at the vehicle client. No matter when or how the vehicle shuts down, information about the task itself (i.e. the request) and any eventual corresponding response will be available when the vehicle is switched on again. Following this approach any request that has not been scheduled or executed, or even was interrupted in an unfinished state during execution, will be able to move on from the "checkpoint" just before the interruption. This principle will also reduce or hide the latency problem (Watson, 1994). Caching will not solve the unstable wireless communication problems but it will reduce the latency by having information ready to be sent again if needed. Latency is reduced by preventing the need of processing and preparation of the same task all over again. Consequently the vehicle client reveals an absence of redundant execution. If no acknowledgement is received from the application server, data will be kept until the proper packet has arrived or the time-out expires.

7.4 Carrier Independency

The importance of mobile applications and their ability to access large amount of data, whenever and wherever in a secure and intelligent manner, increases along with the popularity and development of wireless technology. Many of the billing systems for wireless communication are based on the amount of data transmitted over the wireless carrier, thus the importance of choosing the most profitable solution for each situation, e.g. it might be the cheapest in sense of payment, or it might be the total amount of information that can be transmitted between two endpoints.

In contrast to Internet applications based on wired communication, which normally only need to support HTTP or TCP/IP, mobile applications are intended for many different carriers, e.g. SMS, GPRS, UMTS, WLAN and HTTP. Therefore an application based on a specific carrier most often needs to undergo essential modifications in order to be adjusted and adapted for another carrier, thus this kind of compulsory modifications might be both time consuming and expensive in an economical perspective. Another problem related to carrier differences is the bandwidth which is still limited and expensive. Therefore mobile applications also need to adapt their self to the capacity variation. Yet another problem is the coverage and availability of each carrier. The scope of each differs from one another and they are not all available at all places. A RVD system must be able to support the vehicle's mobility nationally as well as internationally, i.e. the vehicle's movement from one country to another, and at the same time providing solution for exchanges between different carriers as well as impermanent absence of wireless communication.

The bottom line here is that there are no wireless technologies today that fulfill all kind of requirements (i.e. one specific wireless technology that has the ability to fulfill the requirements related to RVD systems) and as described above developing an application supporting all kind of carrier properties integrated in the application is not to recommend. It will most definitely be too complex and hard to maintain. The solution for such problems is to make the software architecture independent of any kind of communication carrier. To achieve carrier independency mechanisms are needed for managing communication protocols and to separate/partition the protocols from the actual application logic. Such approach will relieve applications from any kind of responsibility or knowledge about the actual communication currently used to maintain transmission links between for example the application server and the vehicle client.

7.5 Filtering

Many of the telecommunication technologies' billing systems are either connection oriented, i.e. debiting for the total amount of time connected, or packet oriented, i.e. debiting for the total amount of data-packets sent over the wireless link. Example of connection oriented billing system is GSM, whereas packet oriented billing systems are GPRS, UMTS, etc. Irrespective of the type of wireless communication used, thus also the type of billing system provided, the important fact common to all communication bearer is that less diagnostics data implicates smaller expenses. For instance, using packet oriented systems the overall cost will be reduced due to less data-packet needs to be sent over the communication link. Correspondingly, using connection oriented systems less data implies less obligated transmission time.

Thereby we believe that reduction of diagnostic data is of major importance, not only for transmission optimization but also for reduction of expenses which is proportional to the diagnostics data quantity. The reduction of the diagnostics data quantity will also reduce and facilitate the management of the fundamental hazards related to wireless communication (e.g. minor risk to be affected by the connection interruption, reducing latency, require less bandwidth, etc).

The challenge of providing such behavior, i.e. the reduction of diagnostics data, is foremost providing a solution which will not eliminate or loose any kind of important diagnostics data that would jeopardize the interpretation and the actual mean of its "announcement". There is no use receiving corrupted diagnostics data that can not be analyzed.

Our approach for providing a solution for the presented aspect is based on *filtering* information. The filter property has actually a major role in the proposed architecture. Its primary responsibility is to keep data transmission "volume" to a minimum. By using "intelligent" algorithms the filter property will not only minimize the transmission volume, but it will also contribute to yet another alternative for managing the problem faced with memory constraints.

7.6 Resource-effectiveness

Unlike for example R&D the characteristics and relationship, between service technicians, experts and vehicles, of RVD for the aftermarket are many to many. Considering R&D there might be only few vehicles that need to contain the required hardware to fulfill whatever analysis that are meant to be realized. In such case the cost, and the physical size of the hardware for that matter, is not an issue. The main purpose is to be able to carry out analysis and being able to test the system in real environment without concerning about any kind of expenses. We believe that the solutions used for R&D will not fulfill the requirements of the aftermarket, mostly because of the absence of aftermarket's "many to many" characteristics. R&D solutions also collides with the fundamental characteristics of the commercial vehicle market, where low- costs and maintainability (e.g. low purchase prices and no need for expensive upgrades) are important sales arguments.

Applications developed for the aftermarket are normally not about carrying out analysis on the system itself, when integrated in some particular vehicle or model of vehicles, but it is normally designed and tested for being profitable both for suppliers as well as consumers. The benefit gained by manufacturers is foremost improved customer loyalty, whereas for customers the main focus is on safety, security and of course cost-effectiveness. Cost-effectiveness is not only important from a customers point of view but also from the manufactures point of view. Providing services to customers must also be profitable for manufactures. Therefore cost-effectiveness has an important "role" in the proposed architecture. Based on such circumstances we believe that for achieve such objective one need to focus on introduction of an "optimal" and resource-effective solution for the components localized at the vehicle side of the architecture. Thereby the proposed solution for the vehicle client, which is the component in our layered architecture responsible for any kind of communication with the vehicle itself, need to be as resource-effective as possible, that is producing a solution that minimizes all kind of "unnecessary" expenses for realization of the intended vehicle client.

8 Implementation

This section describes the implementation of the proposed software architecture and some of its characteristic features previously stated in the design principles chapter (see chapter 7). First, we explain the structure of the distributed architecture together with a description of its main components, and then we illustrate how we accomplished to provide the properties needed to meet the demands of our proposed architecture.

8.1 Implementation of Distributed Architecture

As previously described (see chapter 7.1) layered multi-tier architecture has been chosen to deal with the complexity of the system and to meet the RVD's current and future requirements. During development we have focused on partitioning/separating the software into different "independent" layers, with the primary focus on meeting the structural as well as extra functional properties required for software architecture. The result of the software partition encompasses the main components of the system.

The layered architecture provides properties such as modularity, the ability to distribute each layer physically, and portability, i.e. being able to process each partition on different kind of platforms. Although these are important properties and they do facilitate deployment, there is still a need to consider the complexity of making heterogeneous system interoperable, i.e. system interoperability in the sense of facilitating new systems, subsystems, as well as legacy systems to be able to share vital vehicle information. The multi-tier architecture in conjunction with XML, as a standardized data format and data carrier, provides the necessary means to support system interoperability.

8.1.1 Main Components

The result of the software distribution debouched into the following components of the layered architecture:

- Client Application
- Remote Server
- Database Server
- Remote Agent

Each layer has been provided careful and accurate separation boundaries with strongly affected responsibilities. Such action implicates provision of the important property that should be given in any kind of object oriented system development, i.e. weak coupling and strong cohesion between each layer. Figure 8.1 illustrates the main components of the system and the order in a layered manner.

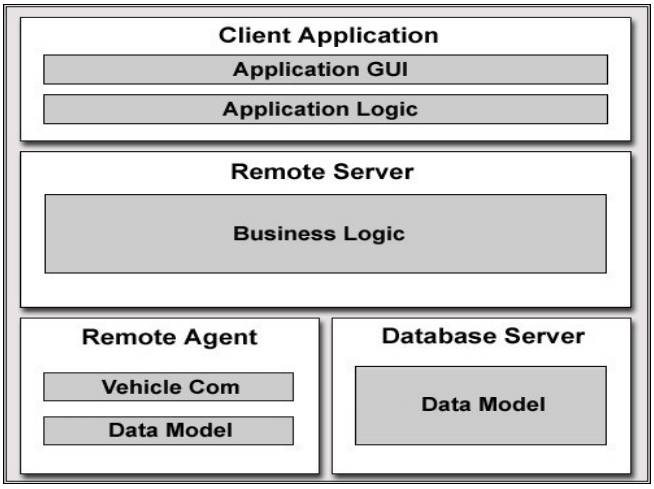


***Figure 8.1**: Layered architecture – Illustrating main component.*

8.1.1.1 Client Application

The client application, which is the interface between the system and the users, is a web based interactive application. This layer is responsible for presenting information in a correct manner to the user. Any request made by the user will be reported to the underlying component, i.e. the Remote Server. Internally the application layer consists of two layers, the application GUI and the application logic. The application GUI is responsible for presenting information to the user whereas the application logic evaluates any input generated by the user before notifying the server.

The client application is implemented with the development tool Flash MX 2004 Professional. By using Flash technology we attained a client that is accessible over the Internet from virtually any computer supporting execution of Flash movies, i.e. computers with a standard web browser and the Flash Player plug-in. This means that the client application can be accessed from all major platforms e.g. Windows, Macintosh or Linux machines running Internet Explorer or Netscape, as well as from Internet enabled devices such as PDAs and Smartphones. Another benefit gained from using Flash applications is the exclusion of any installation procedure, since the application is automatically downloaded to the client computer upon access. Flash applications are also very small in byte-size which assures rapid downloads using minimum amount of bandwidth. The alternative approach to develop the client web application would be to keep it as simple as possible using HTML. Obviously, we decided to go forth with Flash MX 2004. The reason for such decision was based on the lack of HTML's direct interactive experience that matches the capability of GUI client-server applications. Macromedia Flash provides this capability by allowing development of applications designed to give users desktop functionality in their browsers. The strong scripting capabilities of Flash MX 2004 enable the creation of interactive online functionality not possible with HTML.

The client application's main responsibility is to provide end-users (e.g. a service technicians and/or experts) the possibility to request vehicle diagnostics data. From the web interface the user can request run-time parameter readings or specify periodic logs to be performed by the Remote Agent in the vehicle. Any request made is instantly reported to the Remote Server which stores it as a new task in the Database Server.

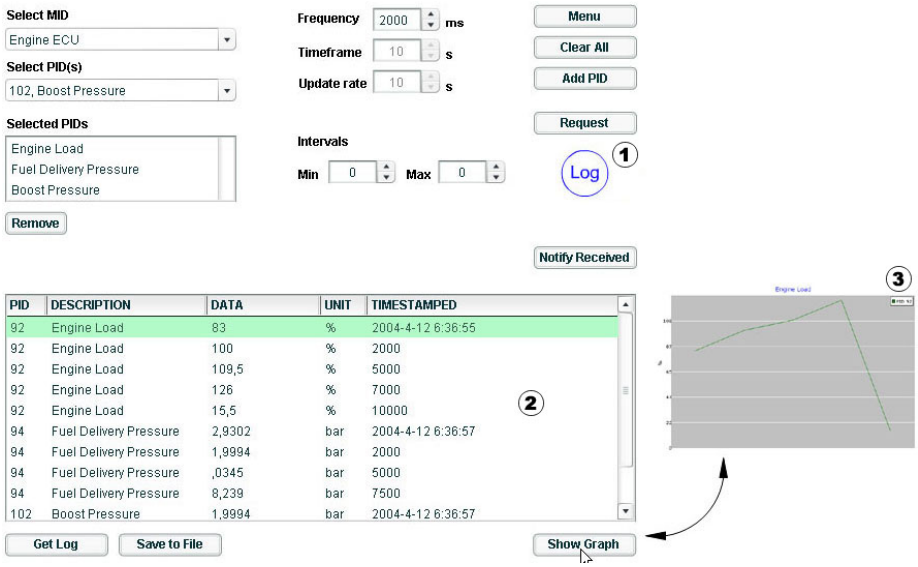


***Figure 8.2**: A screenshot of the web interface, with (1) notifying new periodic log, (2) Data about the periodic log and (3) graph over the selected parameter "Engine load".*

Response related to earlier initiated requests such as run-time parameter readings, periodic logs or occurring DTCs are notified as blinking symbols appearing in the application window. To achieve this notification feature a persistent socket connection is established between the client application and the Remote Server. By using a socket connection the client application will automatically receive notifications when new diagnostics data is added to the Database Server. The symbols will remain until a user has identified the notification and acknowledged the reception to the Remote Server.

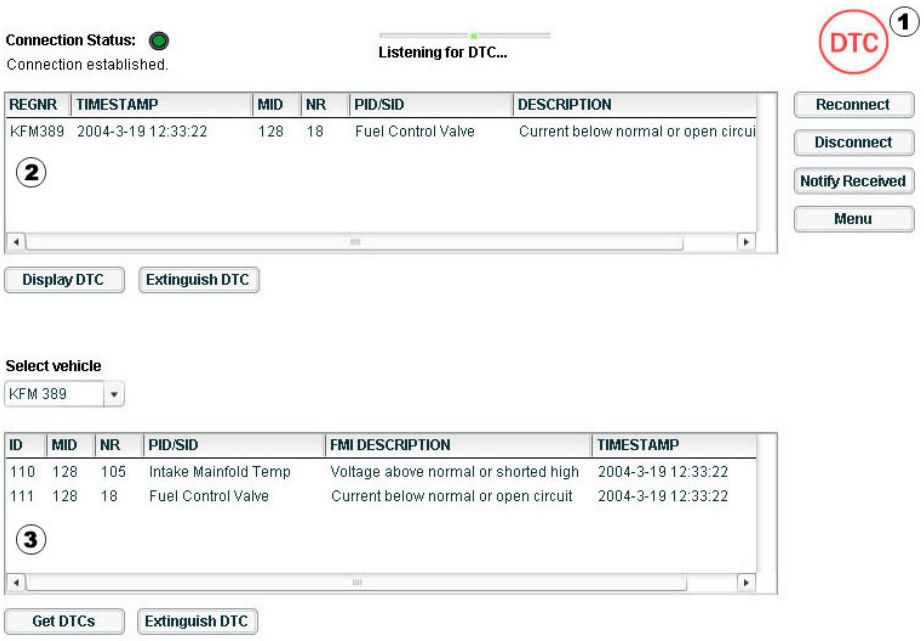


***Figure 8.3**: A screenshot of the web interface, with (1) a notification about the recently sent DTC, (2) Data about the DTC and (3) data about the previous DTC.*

8.1.1.2 Remote Server

The Remote Server works as the middleware in the proposed layered architecture. It acts like a middleware between the client application and the Database Server and also between the client application and the Remote Agent. The Remote Server is responsible for taking care of any request initiated by the client application, e.g. requested parameter readings or logging of diagnostic data. The Remote Server prepares the task for storage in the diagnostic database located at the Database Server. After the task has been stored, a request message is assembled and forwarded to corresponding Remote Agent. When the task has finished its execution on the Remote Agent, a response is sent back and received by the Remote Server, which interprets the enclosed diagnostic data. If the response complies with predetermined rules, the Remote Server notifies the client application about the awaiting response. The proposed architecture also supports transmission of diagnostic data initiated by the Remote Agent. This concerns the occurrence of newly discovered DTCs by the Remote Agent, which are automatically sent to the Remote Server. The incoming DTC is interpreted by the Remote Server and appropriate client applications are notified about the incoming DTC.

The Remote Sever is the component that contains the business logic in the proposed software architecture. The business logic provides mechanisms for interpretation of diagnostic data, i.e. how to process a DTC message and functions for decoding data. For example, an incoming DTC message contains a byte string that has to be analyzed on bit level and then interpreted with the help of descriptive information stored in the Database Server. Other functionalities provided by the business logic are methods for, how to deal with responses that are not acknowledged immediately as well as requests that never

reach their intended target. This includes timers that repeatedly try to reach specific client applications with response notifications. If a client application can not be reached in a predetermined period of time the awaiting response will be redirected to a related client. In the same way the client application will be notified if a request is unable to reach its intended target.

The Remote Server has, in accordance with the structural property of software architecture, several kinds of interfaces that communicate internally and externally. Foremost, the Remote Server has an internal interface, which contains several components that provide internal functionality, e.g. data interpretation, timers, packaging- and parsing of requests and responses. The Remote Server has separate external interfaces for communication with the client application, the Database Server and the Remote Agent. The interface towards the client application partly communicates through a socket connection. This enables the Remote Server to notify the client application about incoming messages, without the client application initiating the communication. Using Flash MX the web interface does not even need to be reloaded when the Remote Server notifies about received responses or communication problems. The communication between the Remote Server and the client application also concerns diagnostics data as response to earlier initiated request by the client application.

The database interface manages all communication with the Database Server. This includes storage of new requests in the diagnostic database as well as permanent storage of received diagnostics data. The Remote Server also communicates via the same interface when requesting data that describes how to interpret diagnostics data and identifying Remote Agents and client applications. The communication between Remote Agent and the Remote Server is always initiated by the Remote Agent.

8.1.1.3 Database Server

The Database Server contains the database management system which handles the data model. It provides persistency, consistency and integrity for the data model. Any request made from the client application or any response delivered from the Remote Agent will be forwarded via the Remote Server to be stored by the Database Server.

To increase performance and to manage several database operations, a modular type of development has been chosen, i.e. taking advantage of provided properties such as stored procedures and triggers. Some of the benefits gained from using these techniques are faster executions, reduced network traffic, and to enhance general types of integrity and preserve consistency. These kinds of properties are crucial for the system in view of the fact that diagnostics database, located at the Database Server, is a critical component of the proposed architecture. It requires an extensive infrastructure of its own to manage the large amount of data collected from a vehicle during its lifetime. Furthermore, the integrity of the database must be maintained due to the important “role” of collected diagnostics data, i.e. composing the basis for future development of diagnostics and/or prognostic applications, e.g. future diagnostics analysis toolsets. A complete description of the diagnostics database conceptual model is illustrated in Appendix B.

8.1.1.4 Remote Agent

The Remote Agent is the interface between the vehicle and the rest of the system. All kinds of requests are sent from the Remote Server to the Remote Agent and, on the contrary, any response or notification generated by the vehicle is sent from the Remote Agent to the Remote Server. At a higher level the Remote Agent can internally be divided into two major layers, the Vehicle Communication (Vehicle Com) layer and the Data model layer. The Vehicle Com is responsible for interpreting incoming requests, preparing outgoing responses, and managing any kind of communication with the vehicle, whereas the Data model is used as a temporary location for storing data.

Following the definition of the structural properties (see chapter 7) of the software architecture, the Remote Agent can be divided into three different kinds of interfaces; internal interface, Remote Server interface, and vehicle interface. The internal interface defines how the Remote Agent is composed from different distributed components and how communication is realized between them. These internal components together compose the intelligence behind the scene of the Remote Agent. As illustrated in figure 8.4 the internal interface consist of several components:

- *Translator* – Encapsulating objects responsible for parsing and interpreting incoming external as well as internal messages and requests. It is also responsible to make sure that information received is surely intended for the current vehicle and that the source of the information has the authorization to send messages and/or request to the Remote Agent.
- *Task* – Responsible for preparing "intelligent" request objects which will make sure to carry through the requested task such as collecting special kind of desired information or taking any necessary action to change the vehicle state or condition.
- *Scheduler* – Objects belonging to the scheduler component have the responsibility to make sure that tasks are scheduled and processed. It has the responsibility to make sure that right amount of concurrent tasks are processed, thus avoiding allocation and "abuse" of to many resources. It is also responsible for avoidance of starvation, i.e. making sure that no tasks remain unresolved and different kinds of tasks are executed concurrently to make sure that no specific task is violated by another type of task.
- *Response* – Encapsulating objects responsible for preparing diagnostics data, e.g. an occurred DTC or results from parameter reading or periodic log (see chapter 6), to be sent further to the Remote Server. It also makes sure that any previous data that has not yet been sent to the Remote Server (e.g. due to vehicle shutdown just before preparing data transmission) are prepared to be sent.
- *Database* – The main responsibility of these objects is to provide interfaces for managing any kind of temporary storage and retrieval operation to the integrated database management system controlling the Remote Agent data model.

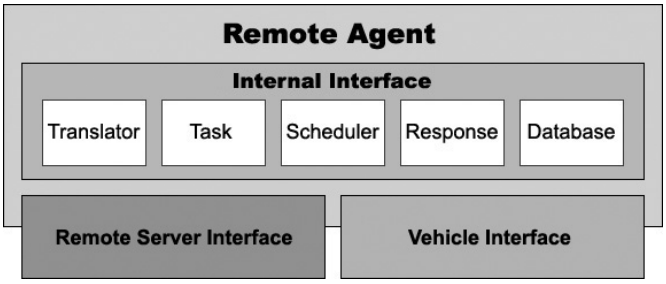


***Figure 8.4**: Remote Agent's Interfaces.*

The Remote Server interface, as obviously indicated by its name, is the interface provided for communication to external entities, i.e. the Remote Servers. It is via this interface that any kind of external messages and requests, together with the remote server's self identification, are sent to the Remote Agent. Using a design pattern such as Observer-Subscriber (Gamma, 1999) several Remote Servers would be able to register interest for receiving diagnostic data such as any kind of occurred DTCs.

The vehicle interface manages objects responsible for any kind of direct communication with the vehicle's internal network, e.g. communicating with individual/several ECUs to request information or update internal software.

8.1.2 XML Interoperability

Our proposed software architecture needs to be independent of any kind of platform, i.e. operative system and development languages/tools. The distributed architecture, which as described earlier is based on component-based layered architecture, enables to physically locate each layer at different locations and to be implemented using whatever development language. Although the properties of the distributed architecture introduces extremely important benefits for a RVD system, it does not automatically provide any solution for making heterogeneous systems understand each others messages and/or transmitted documents.

What we do provide here is the ability of new systems, subsystems, and even future legacy systems to have common interface for application documents and its content, thus making it possible for the systems to understand each other. The requirement for providing such behavior is supporting *interoperability*. Interoperability is the ability of any entity (enterprise, device, application) to communicate, exchange data and be understood by any other entity (Loesgen, 2000). To achieve interoperability between heterogeneous systems one or both of two different approaches can be used, namely adhering to published and well known interface standards or by making use of "brokers" of services that can convert one interface into another "on the fly" (e.g. Common Object Request Broker Architecture (CORBA)). The former approach is the one chosen during this project. In our proposed architecture any document with belonging content is generated and transmitted by the standard data carrier called *Extendible Markup Language* (XML). By using XML we remove barriers to data sharing and software integration. We enable system interoperability by allowing any new additional XML

enabled application to call the methods of current XML enabled applications in the proposed system, regardless of what language either application is written in or on what machine either application is running on. Figure 8.5 illustrates a communication/data exchange example between to different entities of the system, namely the Remote Server and the Remote Agent.

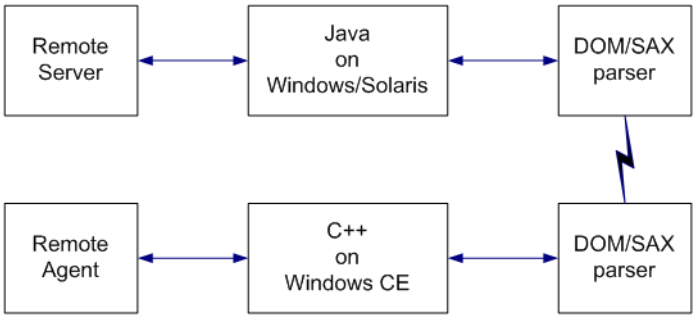


***Figure 8.5**: Heterogeneous Interoperability.*

The Remote server is written in Java and might for example be running on an OS such as Windows or Solaris, whereas the Remote Agent is implemented using Visual C++ on the Windows CE platform. Any method invocation, requests, responses, etc. between the Remote Server and the Remote Agent are made using generated XML documents. This is an important feature to the system due to the possibility of new additional functionality and/or subsystems to the current system, where for example the new subsystem requires enhanced rules and additional elements. Nevertheless, the changes will not affect the primary system's entity behavior. Any additional unrecognized rule or element is simply not transmitted or translated. Figure 8.6 illustrates the structure of XML documents transmitted between the Remote Server and the Remote Agent.

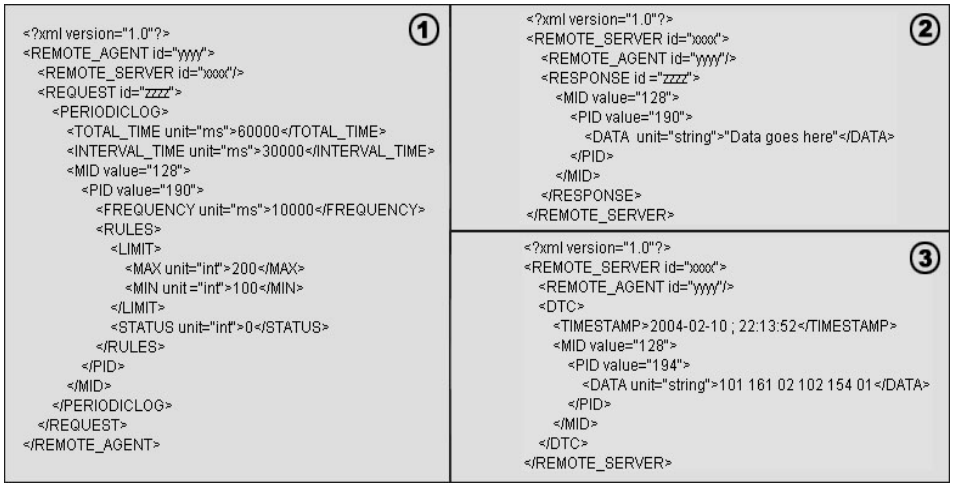


***Figure 8.6**: Example of XML document format transmitted between the Remote Server and the Remote Agent: (1) Periodic log request sent by the Remote Server, (2) Response to request sent from the Remote Agent to the Remote Server, (3) DTC notification sent by the Remote Agent to the Remote Server.*

8.1.3 Architecture Overview

Figure 8.7 illustrates an overview of the architecture based on the discussions and chosen solutions. Client application, Remote Server and Diagnostics database are all distributed at different locations and connected to the public Internet, whereas vehicles which are extremely mobile and each containing the Remote Agent are connected to available wireless communications. The figure also illustrates the data flow between each entity of the system which is based on transmission of vital information "wrapped" in XML documents. The following chapters will go deeper into different kinds of challenges that need to be resolved following realization of our proposed principles for RVD systems.

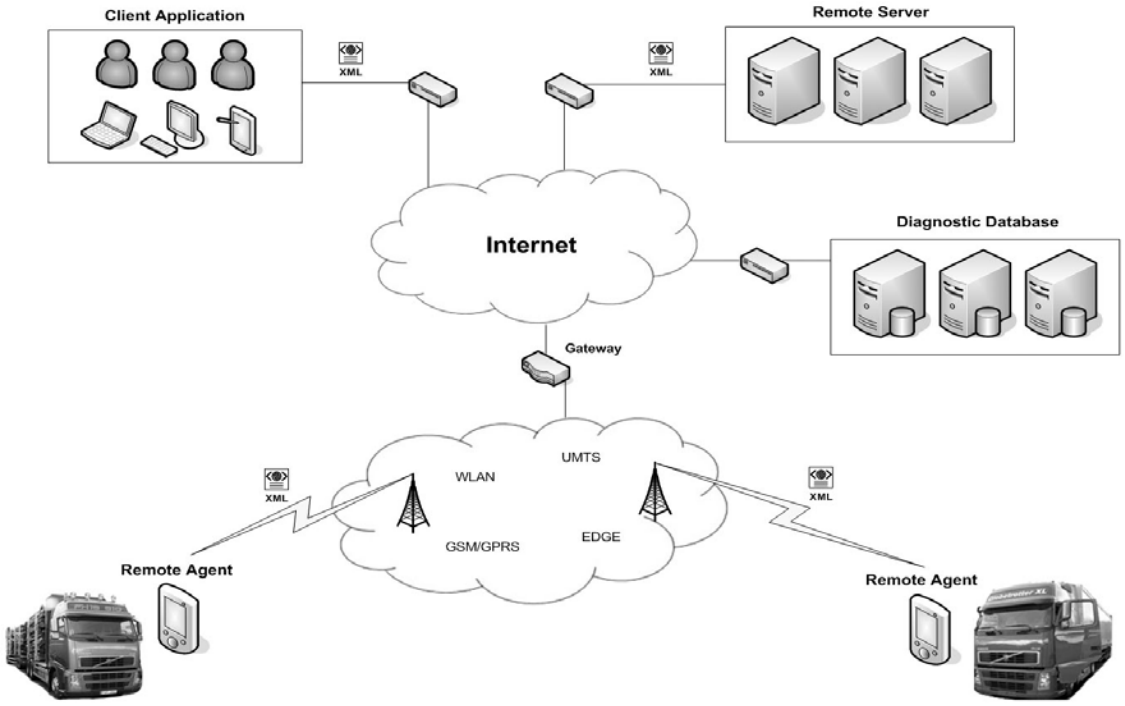


***Figure 8.7**: Architecture overview.*

8.2 Implementation of Data Replication

Data (foremost DTCs) are replicated from the Remote Agent to the Database Server managing the data model, i.e. the diagnostics database, responsible for permanent storage of vital diagnostics data. The partitioning property of the distributed architecture presented in our work (see chapter 7.1) enables the possibility to backup or even replicate the data model to different physical locations and thereby further improving availability and scalability of the proposed architecture.

Due to the mobility of vehicles and the absence of constant and reliable connection, we use pushing technique from the Remote Agent to the Remote Server for realization of data replication. The Remote Server, which normally takes actions on order of a service technician or an expert, does not need to manage the supervision of any vehicle experiencing any kind of state changes, e.g. occurrence of new unmanaged DTCs. If no

notification is sent from the Remote Agent, the Remote Server can be certain that the vehicle can be in no other than 2 different states:

1. There are no new occurred DTCs that need to be managed.
2. The vehicle is out of reach, e.g. there are no communication network available to the vehicle for the moment.

Reflecting on the two different states given above, one can surely conclude that any kind of diagnostics data, especially DTCs, will always be available to authorized users. If any DTCs occur, these will be pushed from the Remote Agent to the Remote Server which in turn will interpret the data, send data to the Database Server for permanent storage in the diagnostics database, and notify the Client application about the new incoming information. Figure 8.8 illustrates a simple sequence diagram of the brief scenario described here (see Appendix A: scenario 2 & 4 for more thorough description).

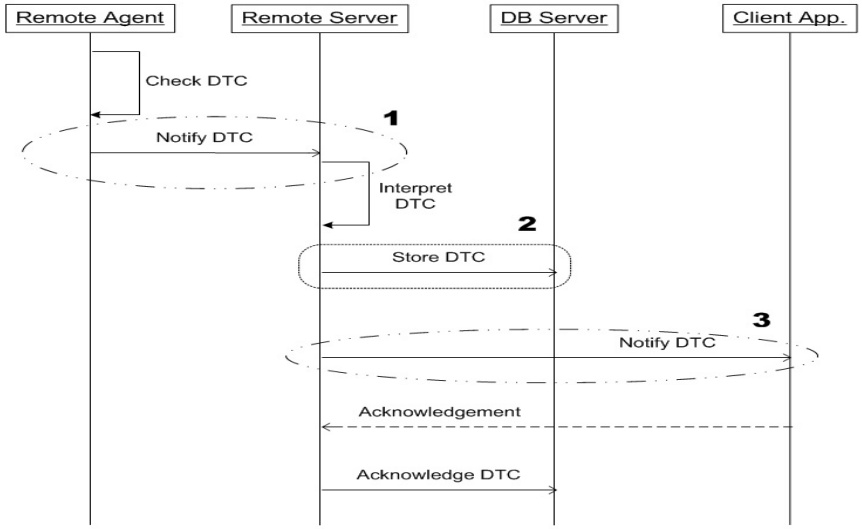


***Figure 8.8**: Simplified sequence diagram illustrating notification using the pushing technique (1 & 3), and data replication for permanent storage in the diagnostics database where information will always be available to authorized users (1 & 2).*

As indicated in the sequence diagram (figure 8.8), any occurrence of DTC will immediately be detected and a notification message (1), including the new occurred DTC data, will be sent to the Remote Server. The Remote Server will in turn take any necessary actions, such as interpretation of new arriving DTCs, before notifying (2) the Database Server to store the new vital vehicle information. The information will be permanently stored in the diagnostics database, thus always being available to authorized users (e.g. for statistical and historical overview of the vehicle health and performance). Another important feature with this solution is the notification generated from the Remote Server to the client Application (denoted as (3) in the sequence diagram). Any new or yet unmanaged DTC occurrences will be automatically forwarded (notification) to the client applications, of workshops responsible for the current vehicle, indicating the received and yet unmanaged DTC. Such notification (3) will be sent from the Remote

Server to the client application until an acknowledgement is sent from a client application indicating that the DTC is notified by for example a service technician. This solution relieves the responsibility of service technicians by not requiring guesses and manual checks from the technician looking for any new occurred DTCs specific to a vehicle or model of vehicles. The service technician will automatically be notified as soon as vehicles experience any state changes (such as emergences of new errors).

8.3 Implementation of Vehicle Client Data Caching

Unlike data replication for permanent storage of vital diagnostics data, vehicle client data caching by the Remote Agent is only temporary. As described earlier (see chapter 7.3) the caching property will solve many problems related to the vehicle itself and the influencing environment surrounding it, e.g. the unexpected shutdowns of the vehicle and intermittent or total loss of connection.

Due to these problems we decided to manage requests, prepared by the Remote Server, by forcing the vehicle to periodically query for eventual new requests that need to be processed by the current vehicle. For example, the vehicle's mobile nature is one of the contributory causes to the high possibility of changes in network and IP properties. Therefore we need to make sure that the vehicle, thus also the Remote Agent, provides the Remote Server with the latest information about available networks and IP properties. The Remote Agent will notify the Remote Server as soon as any network change occurs and at the same time asking for existing tasks to be processed by the specific vehicle (see Appendix A: scenario 1). For realization of such behavior the pulling technique is used to make sure that each vehicle receives belonging tasks that needs to be processed. Figure 8.9 illustrates a simple sequence diagram of the brief scenario given here.

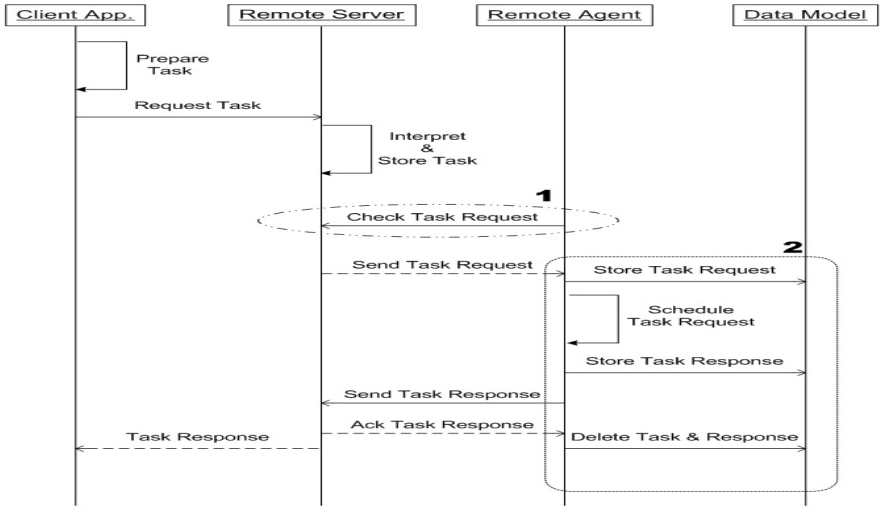


***Figure 8.9**: Simplified sequence diagram illustrating caching and the use of pulling technique. (1) The Remote Agent pulls for eventual task requests that need to be possessed. (2) Illustrating temporary caching, and deletion at the end, of task properties and corresponding response.*

As illustrated in the sequence diagram, the Remote Server will interpret, store and prepare the new request for the Remote Agent. The task will not be revealed to the Remote Agent until the Remote Agent itself pulls (1) the Remote Server to check for new belonging task requests. At this point any tasks that need to be processed will be stored locally by the Remote Agent at the vehicle side. The task properties and corresponding responses (which will be achieved by querying the vehicle for desired information given by the scheduled task) will be "temporary" cached at the vehicle side until the task is done, corresponding diagnostic data is sent back to the Remote Server and an acknowledgement is received from the Remote Server (2). At this point the Remote Agent will trash any temporary data that has been stored for the specific task. This solution, i.e. the vehicle client caching, makes sure that no information will be lost based on for example connection loss or sudden vehicle shutdown.

8.4 Implementation of Carrier Independency

To achieve carrier independency and overcome some of the problems introduced by the mobile nature of a vehicle, we started by separating the actual application component from any kind of communication protocols and related properties. Such utilization provides us communication transparency which, among other benefits, facilitates modification to both application components as well as changes to communication layer without introducing any difficulties based on classic problems such as strong coupling and/or incompatibilities.

At the communication level we use a communication platform called Slipstream. Slipstream provides desirable properties that need to be supported for a RVD system, e.g. the ability to alternate between different networks and carrier technologies, and avoiding unnecessary and costly redundant communications. The platform is provided by Newmad Technologies.

8.4.1 Communication Transparency

As mentioned earlier the great dissimilarity between different kinds of carriers can easily be the foundation of complex and untenable application development if no efforts are put into separation of application from the communication properties. We have separated application and the corresponding logics to a higher structure, and by taking such action the communication property is forced to be totally transparent to the rest of the application, thus avoiding any consideration about the type of communication being used. This solution will not only eliminate the strong coupling that otherwise would have been presented between the application and the wireless communication carrier, but it will also facilitate the compensation of one carrier with another. Any modifications of a carrier protocol or its implementation, the possibility of using multiple carriers simultaneously, and also the risk of being disconnected at any time will all be transparent to the application, thus making it carrier independent. The only responsibility of the application will be to prepare information/data in a way recognized by the interface produced as a middleware between application and communication instances. Gamma et al. (1999)

present different kind of design patterns that can be used to achieve such behavior, e.g. by using mediator, abstract factory, factory pattern etc.

As illustrated in figure 8.4 the Remote Agent's internal interface as well as the vehicle's interface, which all together compose the vehicle client application components, are separated from the underlying communication platform which is used to manage all kind of external communications, e.g. to the Remote Server. The Remote Server interface is the only part responsible for providing information to and from the underlying communication platform. The information prepared by the Remote Server interface is based on XML and does not require any knowledge about the actual communication protocol being used for the moment.

8.4.2 Slipstream

The separation of application logic from the communication rules and implementation demands a suitable solution for managing the internal interaction between the communication layer and the application components. As described earlier the proposed architecture "demands" that the application is relieved from any kind of responsibilities or knowledge about the communication properties and its current state, i.e. the application behavior must be the same regardless of the communication carrier being used and its current state at the moment. The application should also always keep its integrity and act consistently whether working online or offline. This is in fact one of the important features towards the solution of communication reliability and carrier independency.

Slipstream is a communication platform developed by Newmad Technologies, offering intelligent and efficient data management for mobile devices. Necessary properties are provided to meet the requirements for the communication reliability and carrier independency described earlier for the proposed architecture. Slipstream is based on client/server architecture, i.e. it consists of two core parts; one acting server and the other one client. At each side applications are developed as plug-ins to the main slipstream components, e.g. in this case Remote Server as a plug-in to Slipstream Server and Remote Agent as a plug-in to Slipstream Client. The internal communication between each application and the core components is accomplished using sockets, and data is represented as XML. The use of socket as the internal communication protocol between applications and Slipstream components, and XML as standard data carrier, implies the independency of development languages. As long as socket is supported and data is represented in XML there will be no integration problems between Slipstream and whatever development language and/or platform is used to produce the application. Figure 8.10 illustrates an overview of the slipstream architecture.

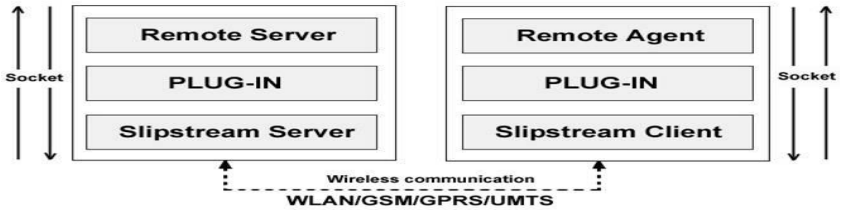


***Figure 8.10**: Slipstream architecture overview.*

Slipstream client is developed to be suitable for devices with limited memory, CPU and battery power. It includes functionality such as:

- *Intelligent network monitoring* – Connectivity to Slipstream Server and general Internet is continuously monitored. Moving from a network to another normally involve change of network properties, e.g. new assigned IP. Slipstream Client will notify the changes and immediately inform Slipstream Server about the changes. This is the roaming capability of Slipstream Client which is suitable for an environment and behavior model such as the one typical for vehicles, i.e. the extreme mobile property.
- *Multiple receivers* – It has the ability to communicate with several servers plug-in simultaneously.
- *Cache handling* – It has repository for data storage. This property is great for handling offline situations. Information will be buffered and sent to server as soon as a connection is established. Application does not need to be concerned about network status.

The Slipstream server is able to manage requests from one or many slipstream clients. Depending on the type of request, the server will either perform it immediately or schedule it for later processing. The server offers functionalities such as:

- *Dynamic network redirecting* – Because of the client mobility the current network address of mobile device may change unpredictably. Slipstream Server will immediately update references to connected devices as soon as it gets notified about the changes.
- *Data transfer with auto-resume support* – In contrary to standard file-transfer methods where data must be resent if the connection is broken - which can result in large overhead and unnecessary costs, e.g. when using communication bearer where billing is based on amount of data-packets transmitted over the network - Slipstream Server offers the ability to resume from the position where it where interrupted without needing to resent all information again. This property is especially useful if there is a need for software upgrade (at the vehicle side) and where normally large amount of data needs to be sent over the network. Because of the mobile nature of vehicles the risk for connection interruption is high, e.g. entering an area with no network coverage, thus it is extremely valuable having auto-resume support in such environments.

- *Connectivity problems* – because of the dynamic redirection, network failure handling and auto-resume support it is possible to transfer data even during weak connectivity.

8.5 Implementation of Filtering

Our filtering properties are principally based on two different approaches, i.e. redundancy elimination and predefined rules, which can also effectively be combined to produce the most efficient data reduction without loosing any vital information.

8.5.1 Redundancy Elimination

Redundancy elimination is about avoiding storage and interpretation of redundant data. Consider a periodic log demanding certain parameter data. The properties of the periodic log might comprise a total time of 30 minutes and a frequency of 500 ms for certain MID with its corresponding PID. A simple mathematic calculation elucidates that the "best case" scenario of maximum reads during the total time will be equal to 3600. Consequently it implies storage of 3600 values. Now, imagine that the value of the parameter has been constant during the duration of the periodic log, i.e. the value of all 3600 is exactly the same. It implicates obviously a waste of storage capacity and unnecessary additions to the data transmission expenses. The redundancy elimination property of the filtering algorithm will make sure that no subsequently equal values, i.e. redundant values, are stored. The subsequently equal property is of major importance for avoiding lost of any important variation of the data pattern (this is especially important when analyzing data using visualization tools such as graphs). Reflecting on the periodic example, by using the redundancy elimination algorithm instead of storing 3600 values, only the first value with corresponding timestamp, i.e. snapshot of the exact time when the reading took place, will be kept for later transmission to the Remote Server. The redundancy elimination will only act on heterogeneous data and consequently ignore subsequently homogeneous values. Figure 8.11 illustrates a simple diagram of the redundancy elimination filter.

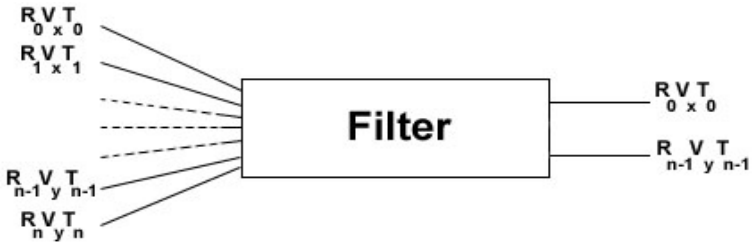


***Figure 8.11**: Filter out consequent homogeneous data.*

As illustrated in the diagram, only $R_0V_xT_0$ and $R_{n-1}V_yT_{n-1}$ will be the values passing through the filter and will be stored for later transmission. The first read, R_0, holds the value V_x at the time T_0. The subsequent reads have exactly the same value, i.e. V_x, and therefore will be filtered out by the algorithm. The next value acted upon is the first value differing from the current V_x value, in this case reading R_{n-1} with value V_y at the time

T_{n-1}. This means that between the times T_0 to T_{n-1} the value V_x has been exactly the same. Conclusively the outcome of N reads will be "compressed" from N nr of storage to an "insignificant" volume, in this case solely 2 values.

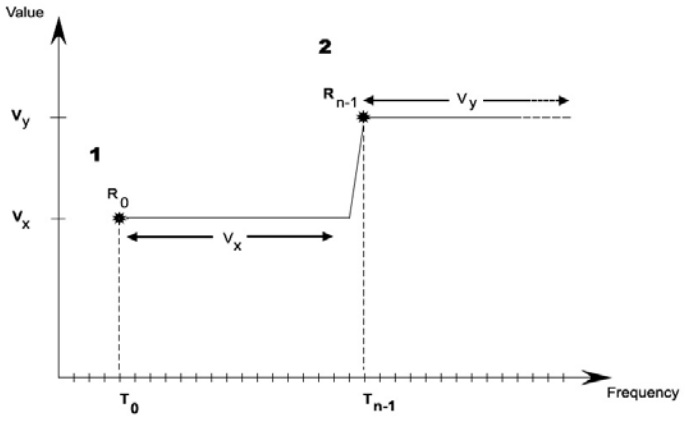


***Figure 8.12**: Filtered values represented in a graph.*

Values that do travel through the filter, in this case $R_0V_xT_0$ & $R_{n-1}V_yT_{n-1}$, do posses sufficient information for complete analysis of the course of event. Figure 8.12 illustrates how it is possible to actually do present a complete graph by only having access to two different values, i.e. the $R_0V_xT_0$ & $R_{n-1}V_yT_{n-1}$. The first read R_0 took place at the time T_0. The next divergent value obtained is the R_{n-1} at the time T_{n-1}. The application responsible for preparing the graph do know, for certain, that all values read at a specific frequency between T_0 and T_{n-2} are exactly the same, thus no explicit information except the first read has been provided. This implies that a horizontal line can be drawn between T_0 and T_{n-2}. As illustrated here the redundancy delimitation is an important filtering feature can provide considerable smaller data without loosing any information.

The accuracy of the provided information is not affected by the filter, but it is rather dependent on the frequency. The less frequency value, e.g. 500 ms instead of 2 s, the more accurate data collection can be provided. The redundancy elimination filtering will make sure that only vital data that are needed to present exact match of the actual course of event are stored.

8.5.2 Predefined Rules

Predefined rules are based on boundaries defined explicitly by for instance a service technician. Service technicians have the ability to generate boundaries and limitations as addition rule to the filtering algorithm. Figure 8.13 illustrates a simple diagram of the idea behind predefined rules, i.e. being able to "sort out" desired data.

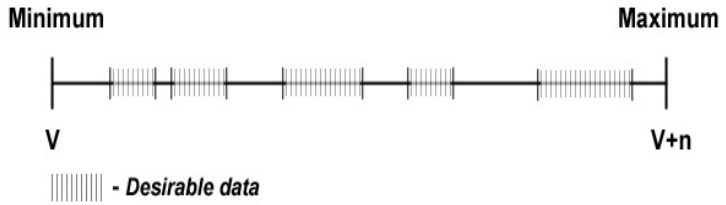


***Figure 8.13**: Predefined rules forcing action to be taken only on data located inside or deviation from pre-specified boundaries, dependent on the type of predefined rules specified.*

Predefined rules provide two important properties, i.e. enabling the provision of semi-critical notifications and the ability to explicitly specify the type of data that is of interest. The former is based on prevention of malfunctioning, i.e. taking action before a potential DTC is generated. As previously described service technicians can specify proprietary marginal values (see chapter 6 & Appendix A). This property of the system enables service technicians to utilize their knowledge and many years of domain experience to define proprietary interval for notification of semi-critical DTCs. The prospect of such utilization can be seen as a basis for prognostics behavior. Service technicians will be able to detect any kind of possible hazards by receiving an alert specifying a possible future DTC occurrence, thus also a future possibility for vehicle performance deviation and malfunction.

The latter property is basically the same as the former describe above. The difference is that action will not be taken on values deviating from the specified boundaries, as it is the case for rules specified for semi-critical DTCs. The latter will only act on values inside specified boundaries. This property entails cost-effectiveness by filtering out unwanted data. This implicates minimization of the amount of data that needs to be sent from the Remote Agent to the Remote Server. The less data that needs to be sent over the wireless billing system the less it will cost.

8.6 Implementation of Resource-effectiveness

Unlike regular servers, such as mainframe computers, small devices (e.g. PDA) have less power and storage capabilities. As we have mentioned earlier these devices suffer from limitations such as insufficient bandwidth over a wireless network as well as memory and CPU constraints of the device itself. These issues are normally the contributory cause to the limited access and/or restricted usage of available resources.

8.6.1 CPU Constraints

The CPU constraints are principally related to speed issues but also related to the bandwidth on the system bus. Therefore it is not desirable to take action on any kind of heavy executions on devices with CPU constraints. In our case this problem is related to the Remote Agent. We did not want to make any CPU or time consuming operations on the Remote Agent. Our ambition has been to relieve the Remote Agent from any kind of operations that would cause the system to slow down or neglect other scheduled

operations. The outcome of such feature in our case has debouched into complex programming on the Remote Agent but at the same time relieving the Remote Agent from any kind of heavy resource consuming operations. Reflecting on the figure 8.4 presented earlier, the diagram illustrates that there is an absence of any kind of internal interface that encapsulates objects responsible for heavy executions (such as extraction and interpretation of "raw" vehicle information). The internal architecture and implementation of components adherent to the Remote Agent is pretty complex, but any kind of heavy execution and/or interpretation of data that are CPU and time consuming is delegated to the Remote Server. This solution also implicated the reduction of latency by "delegating" all kind of heavy execution to the mainframe computer, i.e. the Remote Server, which has more capacity than the Remote Agent.

8.6.2 Memory Constraints

The description given above about the minor capabilities of smaller devices, such as a PDA, is also tenable for memory constraints. Although recent improvements in storage technologies have eased the memory constraints, the available memory - RAM as well as disc/flash EPROM - for such devices are still limited for this kind of business solution. Imagine a scenario where periodic log (see chapter 6.1.2) is inevitable. Depending on the request itself, e.g. the total duration time of the periodic log and the amount of MIDs and corresponding PIDs that need to be queried following a specific frequency, a heavy commercial vehicle is able to produce gigabyte of data during a shorter period of time! Therefore the memory constraints are of big issue and need to be solved, especially in a situation equal to our case where cost-effectiveness has major importance for the overall solution. As mentioned in the beginning of this chapter, because of the "many to many" characteristics of aftermarket, the expenses for delivering such infrastructure should be kept to a minimum. Therefore the problems arisen by memory constraints should not be "solved" by supplementing each vehicle with increased storage capabilities.

We have solved the memory constraints by only providing temporary data storage at the vehicle side, i.e. the Remote Agent. Data, which is normally response to a request generated from the Remote Server but also the actual request properties itself that is stored until the task is done and can be deleted from the internal scheduling system, is only "cached" temporary until acknowledgement is received from the Remote Server confirming retrieval of information. Reflecting on the example given earlier based on periodic log, any kind of hazard related to shortcoming of storage is managed by providing "Interval trashing", i.e. at each interval all temporary data stored for a specific periodic log will be sent to the Remote Server and, consequently, be "trashed" from the local storage management. This solution will prevent the problem with data overflow, i.e. a situation where lack of memory would prevent data storage. Any kind of information that needs to be stored permanently (e.g. DTCs) are stored, via the Remote Server, in the diagnostics database located in the Database Server layer.

Yet another approach used during this project that indeed also contributes to the solution of the memory constraints problem, is filtering unnecessary or redundant information (see chapter 8.5) and thereby preventing storage of undesired information.

9 Discussion

The requirements of business solutions intended for aftermarket are, among others, based on economical prerequisites and customer relationship, i.e. the effort put into trying to satisfy and keep customers as well as improving customers' loyalty by making services more efficient. As Michael Dornan states, changes to EU-legislations will break car manufactures control over services and part supplies (Dornan, 2003). The author also predicts that RVD will become an important tool as a solution for finding new ways to bypass disadvantages generated by EU-legislations. He also suggests that RVD should not be offered as it is, but rather be offered as an important part of the safety and security package.

We share the thoughts and suggestions presented by Michael Dornan, especially about how to introduce RVD on the market, i.e. bundling diagnostics services as part of the safety and security package. However, we also believe that the safety and security package should be extended with cost-effectiveness, especially when dealing with heavy commercial vehicles. In business domains where "just in time" delivery is of major importance it is most significant that customers are well familiar with the economical benefits gained from such services. Prognostics and diagnostics capabilities will prevent major problems that otherwise would have been the contributed factor to severe vehicle damages and downtime, thus not only breaking the "just in time" delivery scenarios and damaging the companies business reputation, but also bring forth unnecessary compensation and reparation costs for the company to handle.

Cost-effectiveness is not only an important fact for customers but also for manufactures. Manufactures providing such services should not need to spend more resources than necessary when producing a RVD infrastructure, i.e. the expenses should be kept as low as possible. Our proposed software architecture, based on presented design principles for aftermarket RVD, do consider these important aspects. The proposed architecture relieves manufacturers from large investments in hardware as well as communication infrastructure. In principle, a minimum of additional hardware and computing power is required at the vehicle-side to provide RVD capabilities and services. Further on, principles and mechanisms are incorporated to effectively gain the most out of diagnostic data as well as restricting data transmission to a minimum without excluding vital information.

The automotive industry recognizes a long term potential of RVD, as a way of improving the relationship between manufacturer, customer and the vehicle, by maintaining a continuous contact with customers. This might also imply positive side-effects such as increased customer loyalty to the manufacturer's brand. RVD opens up an opportunity for vehicle manufacturers to expand existing business models, from merely producing and selling vehicles and spare parts, into actually consider the vehicle and customers as value-adding assets, which favors both sides. Vehicle manufacturers are therefore increasingly motivated, by the strategic possibilities that RVD have to offer in the commercial aftermarket. However, despite the increased opportunities and potential benefits provided by RVD, it must be clarified that such system is not "bullet-proof".

Certain types of vehicle deviations might in fact escape RVD detection, such as abnormal noise coming from the vehicle, but on the contrary gets detected by the driver. Therefore it should be an important aspect of RVD systems to be aware of the importance of "human interaction". We believe that such scenario has to be taken under consideration when deploying software architecture for RVD systems. The driver has to be given the opportunity, if necessary, to act on and notify for example a service technician in a workshop responsible for service of the corresponding vehicle. Such "human interaction" must be a natural part of any RVD architecture. We believe that it should be possible to invoke failure notifications by either the system or a human actor.

Finally, considering the research question of this study, the objective has been to propose suitable software architecture based on the needs and characteristics of the commercial vehicle aftermarket. The conclusion of our study debouched into several design principles that together can be generalized as two major findings that need to be considered in order to accomplish such suitable architecture. The first finding emphasizes the importance of a thoroughly examined and flexible architectural design, e.g. it should facilitate system modifications, integration of additional systems/subsystems, support system interoperability, and being adaptable to different kinds of work practice. The second finding highlights the importance of cost-effectiveness, e.g. keeping hardware and wireless communication costs to a minimum. Each design principle has its own contribution for the overall ability to provide flexible, robust and cost-effective software architecture. Considering these principles we manage to avoid common problems related to existing architectural designs, which normally are denominated as "weak" architectures. The common problems are normally based on the architecture design not being thoroughly reviewed, but are only put together in a rush to manage some critical requirements, e.g. only supporting one kind of communication technology and/or having difficulties to coexist/cooperate with additional system components.

By introducing several appropriate guiding principles for deployment of suitable software architecture, intended for the aftermarket of heavy commercial vehicles, we have managed to answer our research question. We believe that our contribution will help decrease the risk of deploying weak software architectures and instead increase the probability of introducing a more valuable commercial RVD system which will meet current requirements as well as future demands.

10 Future Work

In a future development of the proposed software architecture we suggest some central aspects that would be valuable features and/or ideas to consider in future work. These concern issues of data compression, ability to perform remote software updates of ECUs and evaluation of accumulated diagnostics data.

Due to the characteristics of the requirements of our software architecture we have adopted several techniques to minimize transmitted data (for instance, the use of filtering, see chapter 8.5). To further decrease the amount of data sent over the wireless network, data could be compressed before transmission, e.g. using "zip", "rar" or any similar compression routine. Implementation of this functionality would contribute to a more optimal usage of bandwidth, thus even further improve cost-efficiency of the software architecture. Compression would be especially applicable in our case, where XML is used as data transfer format between the Remote Agent and the Remote Server. The verbosity and the amount of overhead information encapsulated in a XML message are well suited for data compression, which in many cases might reduce the original size of the message with more than 50%. Data compression on the fly will most definitely increase the user-perceived latency (especially when dealing with large amount of data) but the cost-effectiveness gained by using compression mechanisms are much more valuable to meet the economical prerequisites required for the RVD system developed for the aftermarket then trying to avoid the additional latency. In such sense, data compression would most definitely be an important complement to the current system.

In addition to the data compression mechanism, another desirable function to be offered by future implementations of the software architecture is the possibility to perform remote updates of ECU software. This idea, together with a number of other possible use-cases for RVD, was introduced in one of our scenarios (see Appendix A). Several of the thoughts presented here were implemented in our final prototype and successfully validated in "proof-of-concepts" tests. But due to lack of time, the software update scenario was not implemented. To validate the capability of our software architecture, the implementation of this feature would further confirm the strengths as well as revealing potential weaknesses of the architecture.

Lastly, we think that many possibilities are provided by the software architecture's ability to store large amount of permanent diagnostics data in its database system. The diagnostics data can constitute the basis for future development of new diagnostics and/or prognostic applications which consequently could enable a number of different services. These services could for instance provide functionality that offers statistical evaluation and analysis of collected diagnostics data (e.g. DTCs). Examination of the accumulated data might help to find correlations between defective components and certain vehicles and/or models.

References

Agrawal D. and El Abbadi A. (1990). *The Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data*, Proceedings of the Sixteenth International Conference on Very Large Databases, p.243-254, September 1990, Brisbane, Australia.

Bass L., Clements P. and Kazman R. (2003). *Software Architecture in Practice 2nd Edition Reading*, MA: Addison-Wesley.

Bisdikian C., Boamah I., Castro P., Misra A., Rubas J., Villoutreix N., Yeh D., Rasin V., Huang H. and Simonds C. (2002). *Intelligent Pervasive Middleware for Context-Based and Localized Telematics Services*, In Proceedings of the 2nd international workshop on Mobile commerce, 15-24, 2002.

Campos F. T., Mills N. W. and Graves M. L. (2002). *A Reference Architecture for Remote Diagnostics and Prognostics Applications*, In IEEE Autotestcon Proceedings, 842-853, 2002.

Conan D., Chabridon S., Villin O., Kotchanov A. and Saridakis T. (2002). *Handling Network Roaming and Long Disconnections at Middleware Level*, In the Proceedings of the Software Infrastructures for Component-Based Applications on Consumer Devices, September 16, Lausanne, Switzerland.

Dennis M. and Kambil A. (2003). *Service Management: Building Profits after the Sale*, White Paper – Service Management Supply Chain Perspectives, Accenture, March 2003.

Dornan M. (2003). *Remote Diagnostics Could Cure Car Servicing Headaches*, GartnerG2 Report, February 2003.

Ejvegård R. (1993). *Vetenskaplig metod*, Lund: Studentlitteratur.

Fagrell H. and Kuschel J. (2003). *Personal communication*, 5 November 2003.

Ford S. (2002). *Telematics: A 21st Century Service Opportunity?*, Motor Magazine, April 2002.

Fitzpatrick R. (2003). *The Software Quality Star: A conceptual model for the software quality curriculum*, Workshop paper for Closing the Gaps: Software Engineering and Human-Computer Interaction at INTERACT 2003: Ninth IFIP TC 13 International Conference on Human-Computer Interaction, September 2003, Zurich, Switzerland.

Gamma E., Helm R., Johnson R. and Vlissides, J. (1999). *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley Co., Inc, Reading, MA.

Hamilton. K. (2002). *Practical Self-diagnosing AUVs*, SeeByte Ltd.

Hansen P. and Wolfe B. (2004). *Remote Diagnostics – the Next OEM Frontier*, The Hansen Report on Automotive Electronics, Vol. 16, NO. 10, p.1-3, DEC 2003/JAN 2004.

Henfridsson O., Holmström H., Lindgren R., Olsson C-M. and Svahn F. (2003). *Framtidens* fordon – mötet mellan två mobila världar, Vinnova-rapport VR 2003:3.

Hsiao H-I. and DeWitt D. J. (1991). *A Performance Study of Three High Availability Data Replication Strategies*, In Proceedings of the First International Conference on Parallel and Distributed Information Systems (1st PDIS´91), Los Alamitos, CA, December 1991, pp. 18-28, IEEE Computer Society Press.

Holme I. M. and Solvang B. K. (1997). *Forskningsmetodik : om kvalitativa och kvantitativa metoder*, Lund: Studentlitteratur.

Howells-Tierny J. (2002). *The Prognosis for remote diagnostics*, Transport Topics, Academic Research Library, Dec 2002, p12.

Ince D. C. and Hekmatpour S. (1987). *Software prototyping – progress and prospects. Information and Software Technology*, 29(1), 8-14. (Ch. 8).

Jameel A., Stuempfle M., Jiang D. and Fuchs A. (1998). Web *on Wheels: Toward Internet-Enabled Cars*, Computer, v.31 n.1, p.69-76, January 1998.

Kristoffersen S. and Ljungberg F. (2000). *Mobility: From stationary to mobile work*, Planet Internet (pp.137-156). Lund: Studentlitteratur.

Kuschel J. and Ljungberg F. (2004). *Decentralized Remote Diagnostics: A Study of Diagnostics in the Marine Industry*, In Proceedings of the 18th British HCI Group Annual Conference, 6-10 September 2004, Leeds, UK.

Leen G. and Hefferman D. (2002). *Expanding Automotive Electronic Systems*, IEEE Computer, Volume: 35 Issue: 1, January 2002, Pages: 88-93.

Ljungberg F. (1999). *Practical Informatics. Informatics in the Next Millennium – Essays in Honor of Bo Dahlbom*, 83-94. (Ch. 6).

Loesgen B. (2000). *XML Interoperability*, WROX Conferences, Amsterdam, 2000.

Luo J., Namburu M., Pattipati K., Qiao L., Kawamoto M. and Chigusa S. (2003). *Model-based prognostic techniques*, IEEE AUTOTESTCON 2003 Conference, Anaheim, California, September 22-25.

Maintenance Council´s (TMC). (1998). *Tomorrow's Vehicle Electronics Architecture – Service Diagnostic Expectations*, TMC Tomorrow's Truck Position Papers Series, Issued: October 1998.

Mathur A., Cavanaugh K. F., Pattipati K. R., Willett P. K. and Galie T. R. (2001). *Reasoning and Modeling Systems in Diagnosis and Prognosis*, Proceedings of the 2001 SPIE Aerosense Conference on Component and Systems Diagnostics, Prognostics, and Health Management, Orlando FL, April.

Miller J. M., Goel D., Kaminski D., Shöner H-P. and Jahns T. M. (1998). *Making the Case for a Next Generation Automotive Electrical System*, International Congress on Transportation Electronics Convergence '98, pp. 41-51.

Patel R. and Davidsson B. (1991). *Forskningsmetodikens grunder: att planera, genomföra och rapportera en undersökning*, Lund: Studentlitteratur.

Porter M. E. (1985). *Competitive Advantage: Creating and Sustaining Superior Performance*, New York: Free Press.

Ringland G. (1998). *Scenario Planning – Managing for the Future*, John Wiley & Sons.

SAE. (1998). *J1587 – Joint SAE/TMC electronic data interchange between microcomputer systems in heavy-duty vehicle applications*, SAE 1939 Standards Collection, Rev. FEB2002.

Shaw M. and Garlan D. (1995). *Formulations and Formalisms in Software Architecture*, Computer Science Today: Recent Trends and Developments, Springer-Verlag LNCS Vol 1000, 1995.

Sommerville I. (2001). *Software Engineering*, Addison-Wesley.

Vachtsevanos G. and Wang P. (1999). *An Intelligent approach to Fault Diagnosis and Prognosis*, The 53rd Meeting of the Society for Machinery Failure Prevention Technology, MFPT Forum, Virginia Beach, April 19-22.

Valsan A. (2002). *European Remote Vehicle Diagnostics Market*, Frost & Sullivan - Interactive Analyst Briefing, 7th November 2002.

Waern M. (2003). *Real-Time Communication - Evaluation of protocols for automotive systems*, Master of Science Thesis, KTH, Stockholm, Sweden 2003.

Walker G. H., Stanton N. A. and Young, M. S. (2001). *Where Is Computing Driving Cars*, International Journal of Human-Computer Interaction, 13(2), 203–229, 2001.

Wallén G. (1996). *Vetenskapsteori och forskningsmetodik*, Lund: Studentlitteratur.

Watson T. (1994). *Application Design for Wireless Computing*, Proc. Workshop Mobile Computing Systems and Applications, pp. 91-94, Santa Cruz, CA, Dec. 1994.

Wright J. and Pugh T. (2003). *The Changing Landscape for After-Sale Service*, Frontline Solutions, July 2003.

AD&P, Automotive Design and Production (2004). Available on Internet:
http://www.autofieldguide.com/articles/090205.html
[2004-03-29]

Bosch, CAN Homepage (2004). Available on Internet:
http://www.can.bosch.com
[2004-03-29]

Bosch, LIN Hompage (2004). Available on Internet:
http://www.can.bosch.com/LIN/content/What_is_LIN.html
[2004-03-29]

Carr B. J. (2004). *History and Future of On-board Diagnostics*. Available on Internet:
http://www.asashop.org/autoinc/may2004/mech.cfm
[2004-05-31]

Intel Corporation (2004). Available on Internet:
http://support.intel.com/design/auto/autolxbk.htm
[2004-04-20]

Hodgson T. (2003). *Strategic Thinking With Scenarios*. Available on Internet:
http://www.metabridge.com/assoc/stratscen1b.html
[2004-04-15]

Most Cooperation (2004). Available on Internet:
http://www.mostcooperation.com/
[2004-03-29]

Nationalencyklopedin (2004). Available on Internet
http://www.nationalencyklopedin.se
[2004-09-23]

OSGi Allience (2004). Available on Internet:
http://www.osgi.org/
[2004-07-08]

SAE, Society of Automotive Engineers (2004). Available on Internet:
http://www.sae.org
[2004-04-03]

SEI, Software Engineering Institute. Available on the Internet:
http://www.sei.cmu.edu
[2004-04-30]

University of Waterloo - Faculty of Engineering (2004). Available on Internet:
http://www.mechatronics.uwaterloo.ca/home.shtml
[2004-04-29]

Appendix A - Scenarios

Scenario 1: Truck X is driving towards the specified destination. X is part of a bigger fleet with many vehicles owned by company Y. Y have signed a service contract with a Volvo service workshop, who is responsible for maintenance and supervision of the vehicles. While driving on the road, X approaches an area which has coverage of the GSM network. The monitoring system located in truck X, sense the presence of the available communication technology and immediately notifies the server about its online status. The server registers current status of X and communicates when necessary, utilizing the stored information. Further up the road another communication technology, e.g. UTMS, becomes available in parallel with the earlier notified GSM network. Once again the monitoring system notifies the server, which in turn updates the stored information about X's status. Truck X only notifies about the presence of communication technologies in range.

Category: ***Notification – Status***

Scenario 2: The haulage firm Y contracted a Volvo service workshop for maintenance and supervision of its fleet of trucks. The workshop is responsible for the operational function of the vehicles and that possible problems are detected in an early stage. The extent of maintenance and supervision is determined by the specific package deal ordered by Y. One example of this kind of package deal is *semi-critical notification*, which Y has agreed upon with the workshop. Experts within the workshop are responsible for specification of logic, rules and the definition of predetermined limits/intervals for each supervised vehicle parameter. The flexibility of the application give the expert the possibility to specify own limits based on experience and discretion. Truck X is driving towards the specified destination. While driving, the monitoring system located in X, supervises a number of predefined parameter values. Suddenly a value deviates from the specified limits. The deviation is instantly detected by the monitoring system, which notifies the workshop about the semi-critical condition.

Category: ***Notification – Semi-critical***

Scenario 3: A driver and her truck are on their way to a destination with freight delivery. The truck is part of a great fleet at a major haulage firm. The haulage firm has an agreement with a workshop-cooperation. As part of this agreement, RVD will be performed on all the trucks in the fleet, minimizing maintenance costs. The driver of the specific truck is well acquainted with her vehicle, after many hours on the road, and recognizes that something appears to be wrong. Since she is aware of the hauling firm's deal with the workshop, he/she contacts a service technician and provides a description of the experienced problem. The service technician sits down in front of the RVD application in the workshop and connects to the truck. From the received problem description the service technician already has a pretty good idea of which parameters to examine. Thus, the service technician decides to perform a real-time examination of some parameters. If any of these parameters shows deviating values an identification of the actual problem might be made. If the problem cause cannot be established, a greater

number of parameters can further be examined, either in real-time or with a periodic log during a specified amount of time. A periodic log can provide data for a more thorough analysis. Based on the collected data from RVD the service technician can come to a conclusion of how to solve the problem. This can concern fine-tuning of some parameters, software updates of the control units or if the problem cannot be solved remotely, schedule time for service at a workshop. The service technician might also come to the conclusion that the problem doesn't require immediate attention and notifies the driver about scheduled maintenance in the nearby future. How the service technician acts in a situation like this depends on the significance of the discovered problem.

Category: ***Notification, Periodic reading, Real-time***

Scenario 4: A truck is part of major haulage firm. The haulage firm has an agreement with a workshop-cooperation. As part of this agreement, RVD will be performed on the trucks in order to prevent vehicle malfunction. During operation of the truck, some parameters produce deviating data which in turn generates a DTC. This DTC is sent to the workshop by the monitoring system in the vehicle. The service technician who receives the DTC cannot, based on his own experience, figure out what the actual cause of the generated DTC is. He decides to extinguish the DTC in order to see if it occurs again. If it reappears the service technician can initiate a periodic logging of relevant parameters. With the knowledge gained from the collected vehicle data, the source problem of the occurring DTC can hopefully be identified and corrected.

Category: ***Notification – Critical, Real-time, Periodic***

Scenario 5: A new version of the truck's ECU-software is available. The workshop decides to perform a wireless update of this software. For security and safety reasons, this update must be carried out when the truck is standing still. Accordingly, the workshop informs the driver about the situation and the driver pulls over at nearest gas-station or roadside parking for a short pause. The driver takes a cup of coffee and after 5-10 minutes the truck has received the latest ECU software.

Category: ***Real-time***

Scenario 6: A truck has a delivery from Stockholm to Malmö. It is on a very tight schedule so it is important that everything runs smoothly, without any delays. Because of the truck's heavy cargo, some uphill parts of the road lower the speed considerably. When the truck approaches Hallandsåsen the road stretches a few kilometers uphill. Consequently, the truck only reaches a speed up to 50 km/h. This decrease of speed will delay the delivery with approximately 10 minutes. The driver calls up the workshop with his mobile phone and informs a service technician about the situation. The service technician estimates that an additional 200 hp would do the job. This extra power can be gained by the modification of a few parameters in the truck's ECU. When the truck is

over the hill, the modification of the parameters is reversed to original configuration. In this way the fuel consumption and emission levels are optimized.

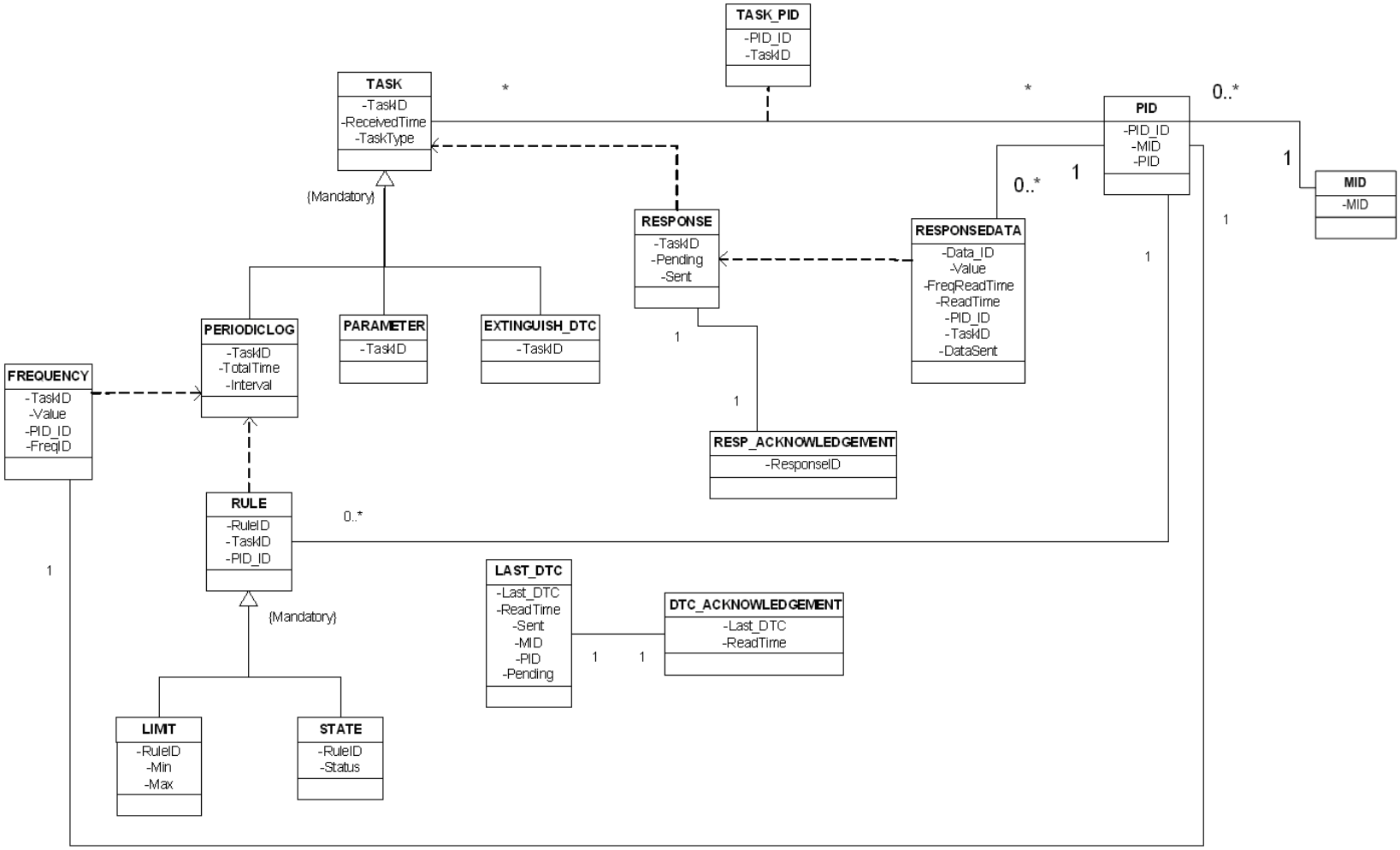
Category: ***Real-time***

Scenario 7: An occurring fault in the truck is detected by the monitoring system via a DTC generated from the ECU. The workshop receives a notification about the presence of the DTC. A service technician connects to the vehicle ECU in order to find other possible faults. The service technician successfully identifies the problem, but realizes that the specific problem cannot be repaired remotely. It seems to require replacement of a hardware component. Instead he sends necessary information about the flawing component to the workshop that will repair the vehicle. He also informs the driver about new scheduling time for service and maintenance. With the information sent from the service technician, the repairing workshop can have all parts ready when the truck arrives.

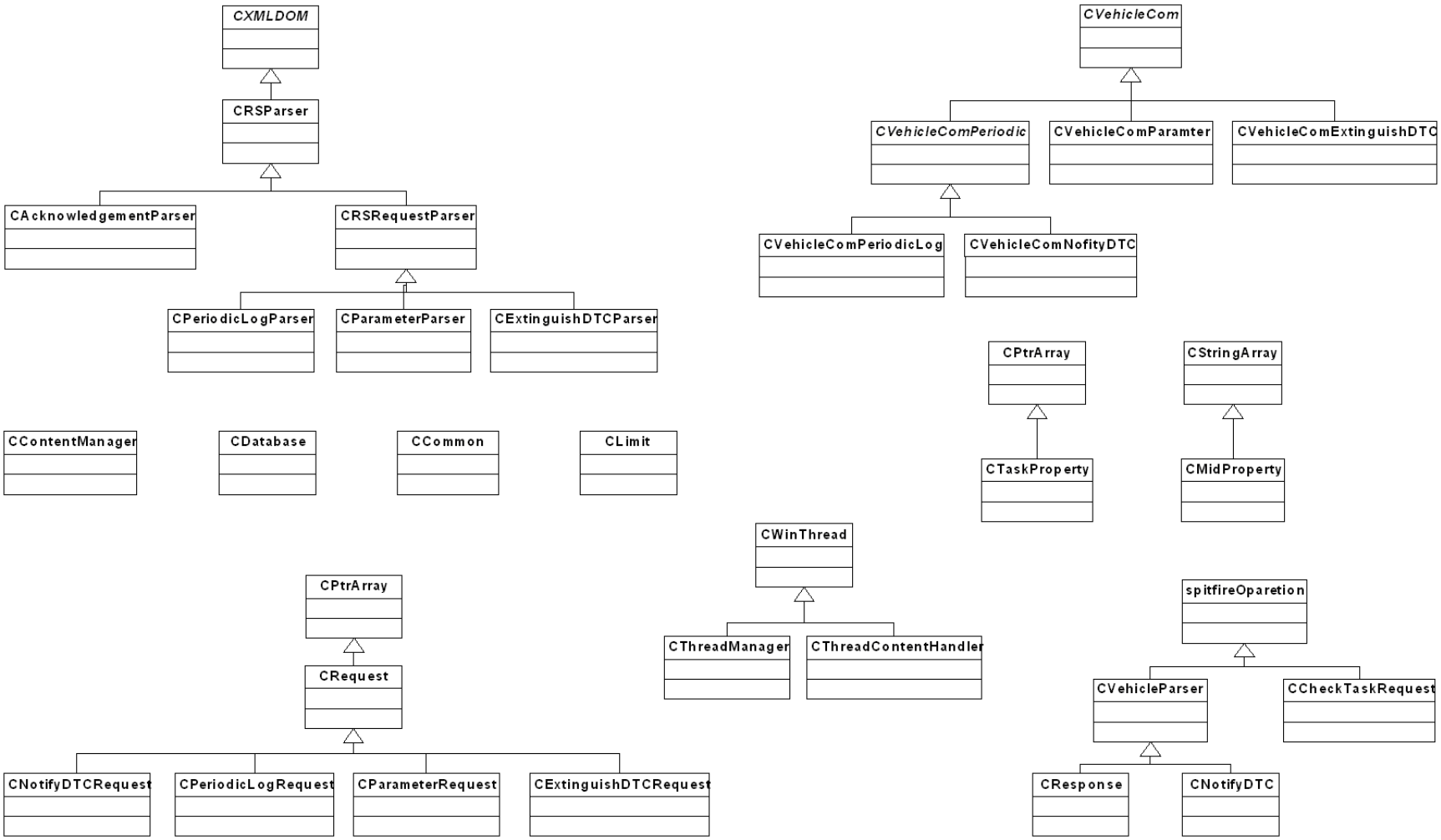
Category: Notification – Critical, Real-time

Appendix B – Diagrams

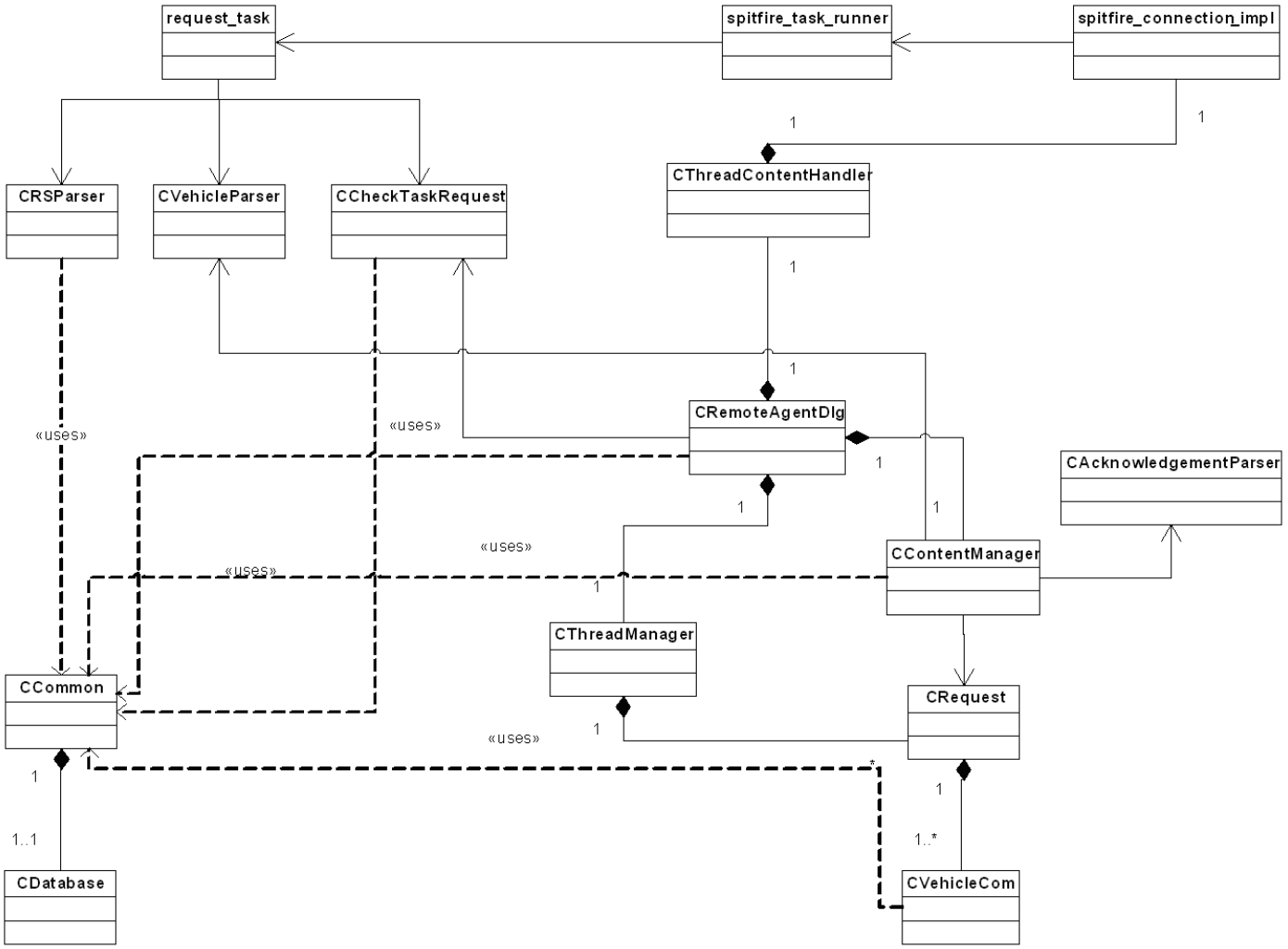
Remote Agent – DB Conceptual Model



Remote Agent – Inheritance Relationship



Remote Agent – Object Diagram



Diagnostic Database - Conceptual Model