CHALMERS

Functional development of Driver Information Module using Simulink

Master of Science Thesis in the Program Software Engineering & Technology

Saeedeh Jadid Tavaf
Marjan Mahmoudifar

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Göteborg, Sweden, 2010

Functional development of Driver Information Module using Simulink

Examiner: JOACHIM VON HACHT

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden April 2010

Abstract

Safety seat-belt reminder software provides appropriate alerts -including gong sound alarm, visual indication and text messages- with the intention of reminding the passengers of using seat buckles in a car. The current thesis report elaborates on the procedure of functional development of the mentioned software for Volvo Car Corporation (VCC) which is an automotive manufacturing industry in Sweden.

Generally speaking, developing the seat-belt reminder software is initiated from extracting the software specifications according to a number of fixed and definite functional requirements. The specification document is handed to the code supplier based on which the final software is programmed. However, those pre-defined requirements are likely to be altered during the software life cycle. The new changes, during the development, are evaluated and then the decision will be made based on the severity and the complexity of those requirements to be implemented. If the new requirements are simple to implement the supplier includes them in the current software under development otherwise the requirements will be delivered within the next release.

In order to prevent the negative effects of mentioned changes, an idea can be simulating the preliminary requirements using the Simulink, generating the automatic code using Embedded Coder tool and consequently testing the first iteration code. This innovative idea provides VCC with detecting the probable faults together with analyzing the feasibility of new functionalities.

The current thesis report presents the development stages of the Simulink model, automatic code generation for target systems as well as validation of the generated code. In addition here, we compare two software development process models, V-model and model-based design, considering the advantages and disadvantages with respect to the driver information system (DIS) development. As a result, we develop the safety seat-belt reminder software using model-based design.

Contents

Abbreviations

SRS: System Requirement Specification

DIM: Driver Information Module

MBD: Model-Based Design

EU Gen4: Europe Generation 4

VCC: Volvo Car Corporation

1. Introduction

1.1. Background

We are witnessing an increase in the usage of embedded systems within a wide range of applications such as “aviation equipments, automotive electronic systems, communication equipments, etc”. [7] Due to the fact that embedded systems are increasingly becoming more complex and safety-critical, engaging appropriate development approaches to implement them is a significant issue. Therefore, the traditional development models which mostly involve testing phase subsequent to the manual coding are not suitable any longer.[7]

“The role and responsibility of embedded software in safety-critical automotive applications is ever increasing.”[17] Safety seat-belt reminder software is a considerable automotive embedded system. Achieving such a system consists of collecting the input signals from the corresponding sensors over the Bus, applying various behavioral requirements and finally sending the output signal back to the Bus.

Presently, the requirement specification is written as a text by an effort of the design group. According to the text-based specification document, the supplier starts implementing the software using a programming language. Consequently, the test procedure comes late in the development cycle hence; resuming all stages to fix the problem is very expensive in terms of both time and money. It is worth noting that the requirements are always changing due to responding to the market needs therefore, the code should be modified accordingly. [3]

In order to resolve above-mentioned problems, the idea of using model-based design was brought up. This approach provides the great opportunity to capture requirement faults and conflicting specifications early in design stages. Besides, applying changes is much cheaper and easier before the coding and hardware implementation get started [2]. Modeling and simulation tools can be utilized to take advantage of the model-based design approach.

1.2. Purpose

This thesis work specifically aims at studying V-model development process model, which is currently used at VCC, against the modern model-based design method. Therefore, two

models are weighed up in terms of the advantages and disadvantages. To support the comparison safety seat-belt reminder system is being developed following the model-driven development.
In sum, the objectives of the thesis are summarized as bellow:

- Research on model-based design and V-model
- Develop a Simulink model for safety seat-belt reminder to support the proposed model

The remainder of this report is as following: after the introduction, section two presents the research methodology applied and data collection methods will be discussed. Section three is dedicated to set the stage by giving some technical backgrounds. In this section different development processes models and modeling tools are talked about. The report is continued by giving an overview of the development process in section four. Section five presents the result of the current work by making a comparison of two presented development processes. Finally, the section six sums up and conclude the thesis work.

1.3 Restrictions

In the current work, there are some limitations and restrictions. The most important limitation we deal with is the production time which makes us to bind the thesis scope to fit into our schedule.

Fulfilling the first objective of our thesis, we needed to implement the safety seat-belt reminder functionalities using both V-model and model-based design in order to have more accurate comparison. However, because of two reasons we avoided implementing the system using V-model. One is it was not in the scope of our thesis and second is the shortage of time. Therefore, instead of developing the prototype using V-model, we conduct researches on V-model itself as well as reading articles and reports from VCC in order to gain good insight into it. In addition to the literature reviews, some interviews were done with the experts at VCC which will be described in detail later in this report.

2. Technical background

2.1 Software Development Process Models

Software development processes are regarded as an appropriate technique which provides efficient communication among anyone who is engaged in a software project. [12] " They enhance management's understanding, provide a precise basis for process automation, and facilitate personnel mobility and Process-reuse that yields in reduction of Cost, increase in Reliability, Productivity and Quality." [12]

As Somerville pointed out in his book software engineering [1], the software process is "a structured set of activities required to develop a software system:

- Specification
- Design
- Validation
- Evolution

"A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective."[1]

Waterfall, evolutionary development and component-based are some examples of generic process models, in which each model has its own pros and cons. The process model which is used in different organizations might vary based on the type of the software being developed, the requirements, resources and the policy of the organization. Each organization adopts one or combination of several software development process models according to its criteria and needs.

2.1.1 V-model

Volvo Car has been applying traditional V-model process model which is indeed similar to the waterfall model due to the inclusion of all waterfall stages from requirement analysis to testing and maintenance. However, the remarkable difference is that it has a bend at the coding stage which makes the model to go upwards. Figure 2.1 demonstrates the typical V-model including different phases [5].

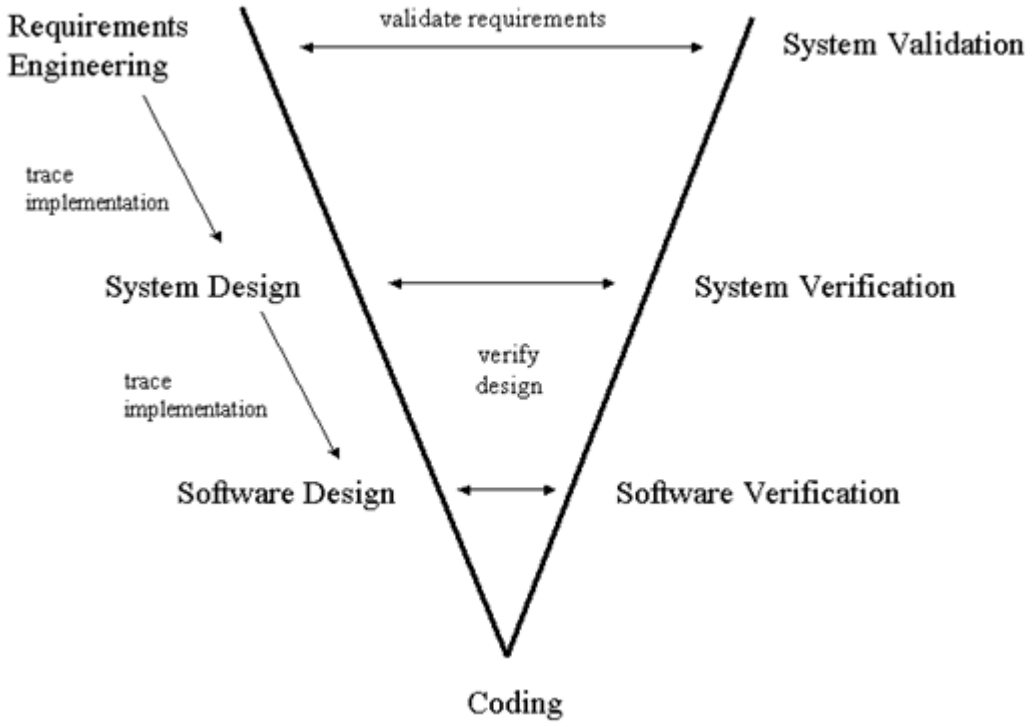


*Figure 2.1.*Typical V-model [5]

In this model, each stage of software development has an associated testing phase. Since the tests are directly extracted from the correlated development phase, each requirement and design phase can be verified independently.

Nevertheless, test and validation will be carried out late in the software development lifecycle which leads to project delays in addition to considerable cost and time for the product updates. [5].

Basically, two approaches implementing V-model are specification-based and model-based design. “Specification-based development is traditionally a more manual, documentation led, method leading to manual coding”.[17] As an alternative, the model-based approach proposes the potentiality to decrease the development time and cost together with the

opportunity of taking advantages of the automated code generators. The next sub-section elaborates more on the model-based development. [17]

2.1.2 Model-based design

Considering the drawbacks of traditional software development process models, using a model which proposes modeling and analysis of the system in the early stages of design would be a promising solution. Moreover, the project tasks are accomplished in an iterative way so that each project step is fulfilled and fully verified before transition to the next step. Furthermore, since timing is considered as an important factor in the real-time embedded systems, we require a proper process model which includes timing constraints, the time of getting inputs and producing results. [8]

The model-based design is mostly focusing on design, simulation, prototyping and analysis of the requirements. The modeled specifications can be executed and tested offline providing the designers with detection of the requirement defects and conflicts prior to the implementation phase. [8]

On the other hand, creation of a rapid prototype supplies the customer with a general overview of the final product. Furthermore, modeling tools are paired with the real-time target hardware to provide the opportunity of the simulated model testing on the real-world hardware.

Considering the stated benefits of model-based design approach, development process model of the seat-belt reminder software is replaced by a modified version of the old V-model (figure 2.2). [9]

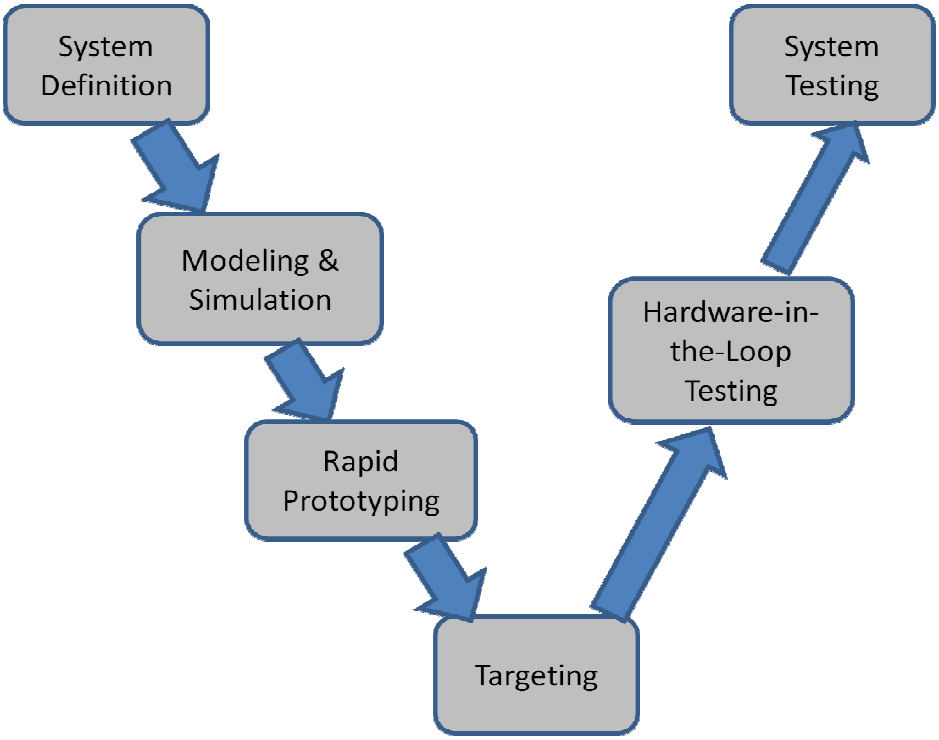


Figure 2.2. The modified V-model development cycle [8]

The above figure shows the development cycle of embedded control applications common to automotive, aerospace, and defense applications [8]

The cycle starts with the analysis and documentation of the defined requirements and specifying the overall architecture of the software system. In addition, the specifications of the target hardware can be determined. [9] Afterwards, the virtual software prototype is constructed directly from system requirements by the means of simulation and modeling tools. Consequently, the executable model is validated to assure that the control system is functioning thoroughly according to the specification. Using automatic code generators, the source code is generated from the model. Although generated code can be deployed on a wide variety of the target hardware systems, the actual hardware prototype might be required in order to test the control system in the real-time environment. To achieve this, the various Commercial-Off-The-Shelf components can be applied to connect the physical signals to the control model. [9, 10] The development cycle continues by deploying the

generated code on the target hardware and doing numerous tests to assure that the final system has satisfied requirements. If all requirements are met, the model-based design process will be concluded. [9, 11]

2.2 An Overview of the Modeling Tool

2.2.1 MATLAB/Simulink

To apply model-based design for embedded systems, designers take advantage of numerous modeling and simulation environments including VisSim (Visual Solutions Inc.), RIDE (Hyperception Inc.), MATLAB/Simulink and MATRIXx (National Instruments). All of these tools are interactive, graphical environment using block-based diagrams, focusing on modeling, simulation and evaluation of control systems. However, this thesis work utilizes Simulink in order to produce the executable requirements together with Workshop Embedded Coder to automatically generate code from model blocks.

Simulink, concatenation of SIMulation and LINK, is an extension to MATLAB® which is a mathematical tool. It aims at visually programming by modeling, simulation and analysis of a system using a graphical user-interface environment. Simulink provides the possibility of using the available standard blocks included in various libraries for implementing any dynamic systems which causes the designers to avoid coding.

Generally speaking, there are four steps involved in the model-based design by Simulink:

- Building executable specification with models
- Performing design with Simulation
- Achieving implementation with automatic code generator using Real-time Workshop Embedded Coder
- Test & Verification [3]

In addition, the architecture of the model can be hierarchical so that the bottom-up and top-down approaches can be employed. That is to say, in order to build a well-organized and user-friendly graphical model, designers can classify the blocks into the number of subsystems. In fact, each subsystem represents one or a group of identical functionalities.

The initial step for modeling is to define the mathematical equations which illustrate each subsystem either using available blocks or building new ones. Together with subsystems or functions which implement the system logic, the source and sink blocks are required to get the input and output signals. Blocks which are transferring data to each other are connected by arrows. Data are processed throughout the model and generate the result which is either displayed or stored in a file.

Eventually, the built model is simulated and the result is evaluated. It should be cited that configuration parameters such as start and stop simulation time are set before running simulation. [4]

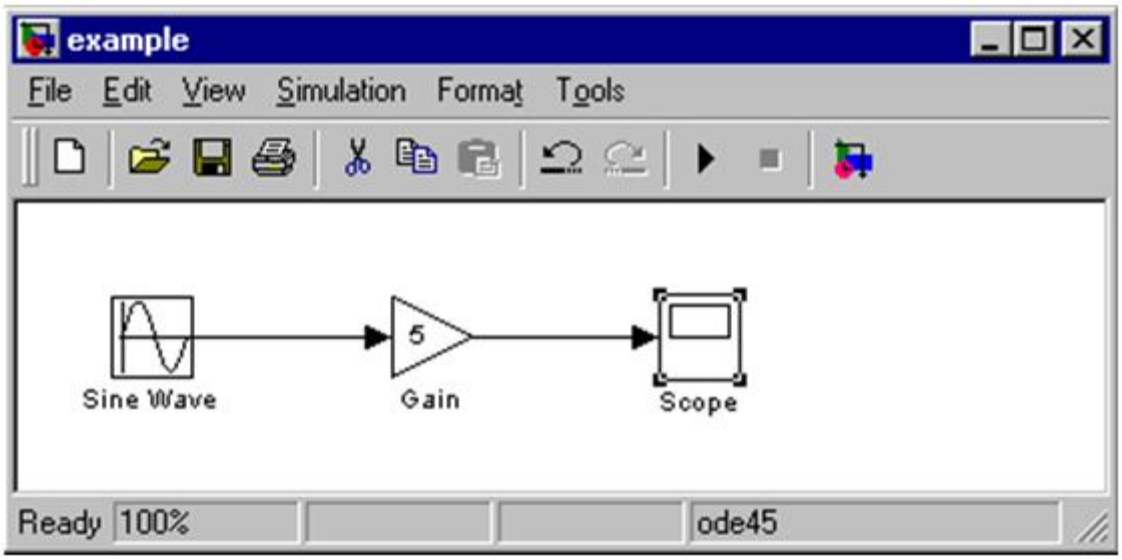


Figure 2.3. A sample Simulink model [13]

The above illustration (figure 2.3) "consists of three blocks as well as the lines to connect the blocks. Sine Wave block is a Source block which a sinusoidal input signal originates. This signal is transferred through a line in the direction indicated by the arrow to the Gain Math Block. The Gain block modifies its input signal (multiplies it by a constant value) and outputs a new signal through a line to the Scope block. The Scope is a Sink Block used to display a signal (much like an oscilloscope). "[13]

Following the model construction, the simulation can be commenced by clicking on the Start button. As it is obvious from figure 2.4, the output signal is resulted from the sine wave multiplied by 5.

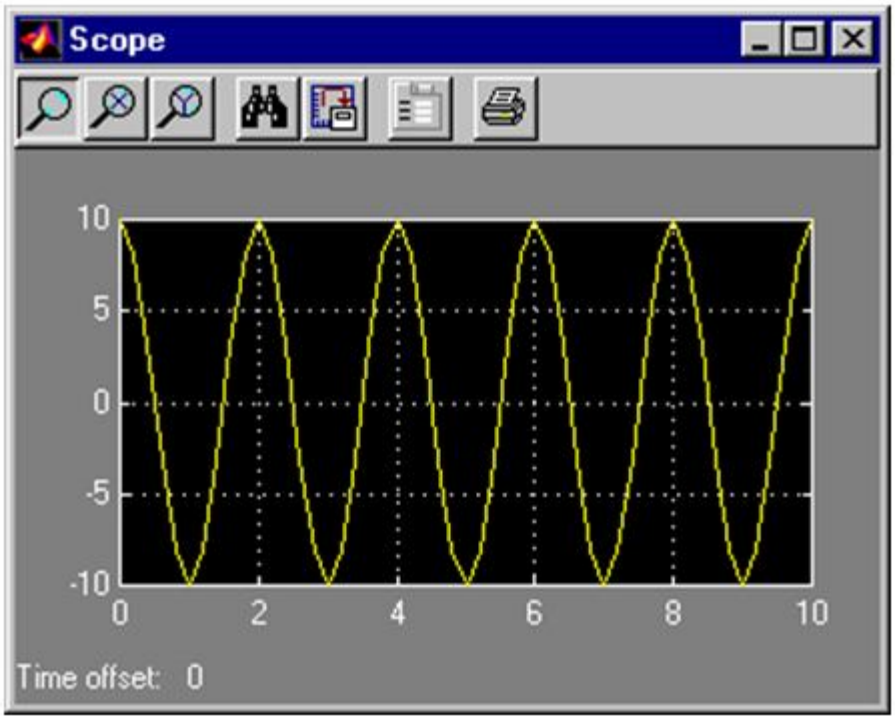


Figure 2.4. The output scope of the Simulink model [13]

2.2.2 Stateflow chart

A Stateflow chart is a finite state machine which works based on different states, transitions, conditions and actions. When one condition is fulfilled, model is transited from the current state to another and the defined action is also done. This way, the workflow between different system states is obviously demonstrated in the figure 2.5. Each Stateflow block represents a corresponding Stateflow chart. Incorporating Stateflow blocks into the Simulink model provides the designer with the possibility of controlling complex systems and supervising the precise workflow.

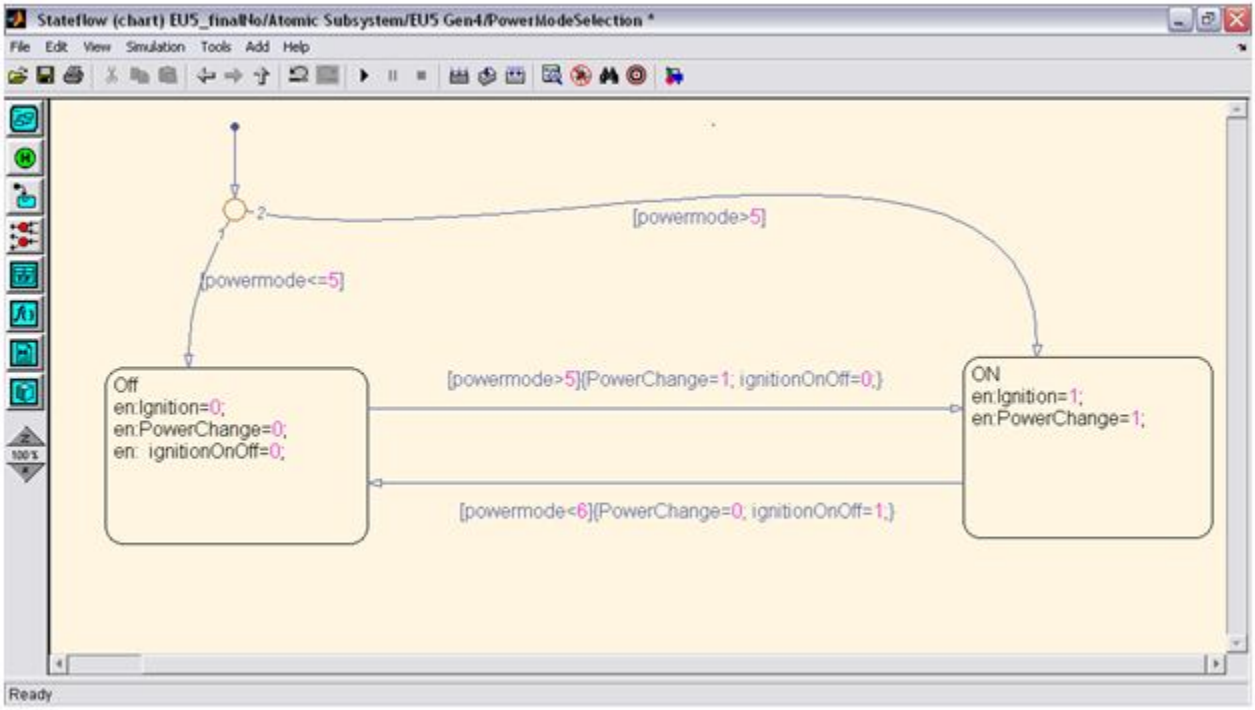


Figure 2.5. A sample Stateflow chart

Above Stateflow chart contains two states which are connected by the transition lines. The upper arrow is the default transition which is directed as the simulation starts and is divided into two transitions with two different conditions. When either of the conditions on "powermode" is met, the corresponding transition is directed to the appropriate state. In each state, several actions are performed in order to set the outputs.

Model is fed by many different signals generated by the corresponding sensors over the Bus and directed to the various blocks and Stateflow charts. Finally, the outputs are displayed to the user. Due to the fact that all computer systems are discrete, the data process and transfer is only conducted at discrete time, hence, the simulation time step should be specified.

3. Exploring V-model

3.1 Research methodology

The current thesis is based on an empirical study on functional development of safety seat-belt reminder system at VCC and it follows a qualitative research approaches. Therefore, we utilize a number of interviews and literature reviews to obtain desired information regarding V-model concept and positive and negative aspects.

3.1.1 Literature reviews

Literature reviews were conducted to address the first objective of our thesis. In fact, the literature reviews were done first to gain good insight into the V-model approach. Besides, the benefits and drawbacks of using V-model as a software development process model are collected in order for us to accomplish argumentations and comparisons.

The research papers used in our thesis, in addition to the documents provided by our supervisors to review, were mostly obtained through searches in IEEE, ACM, Springer and SAE. Next step of the literature review was done by reviewing the abstracts and conclusions of the papers to refine and finalize our selection.

In addition, the Volvo documents were deeply utilized to attain clear vision on how Volvo defines the seat-belt reminder requirements and how they communicate these requirements with their supplier for further development.

3.1.2 Semi-structured Interviews

To acquire comprehensive knowledge about the software development process model presently employed at VCC, few interviews were done. Through these interviews, we became acquainted with the different employee’s analysis and experiences of the V-model.

Before each interview some questions were prepared and new questions based on the interviewee's answers were asked during each interview. The questions varied from an interview to another one based on our understanding of the topic and our progress in development. During each interview we took notes of the interviewee's answers for further analysis. To give an example, Appendix A presents a sample interview including a number

of inquiries from an engineer in SEM (body domain) group. Finally, after each interview we reviewed our interview notes to validate our conclusions. Our findings through interview are summarized in sections technical background, development process and prototype development.

3.2 Findings

The responses to the inquiries in the interview carried out in SEM group revealed that V-model is being used throughout the development cycle of the embedded software systems at VCC. The most significant cause is the huge possibility of designing and performing tests at the same time as the development phases. As a result, designers take the opportunity to evaluate the fully-developed system regarding the requirements synchronized with the design stages. Simply stated, the verification can be directly carried out on the design document prior to the final software product. In the case of any verification failure, the process returns to the design phases and keeps the iteration until completion. Nevertheless, the main concern is to shrink the V-model so that the gap between the development and testing phases gets diminished.

According to the interviewee, the best process model is to integrate the model-based design (next section elaborates more on this) stages into the V-model to benefit from the positive points of both mentioned models.

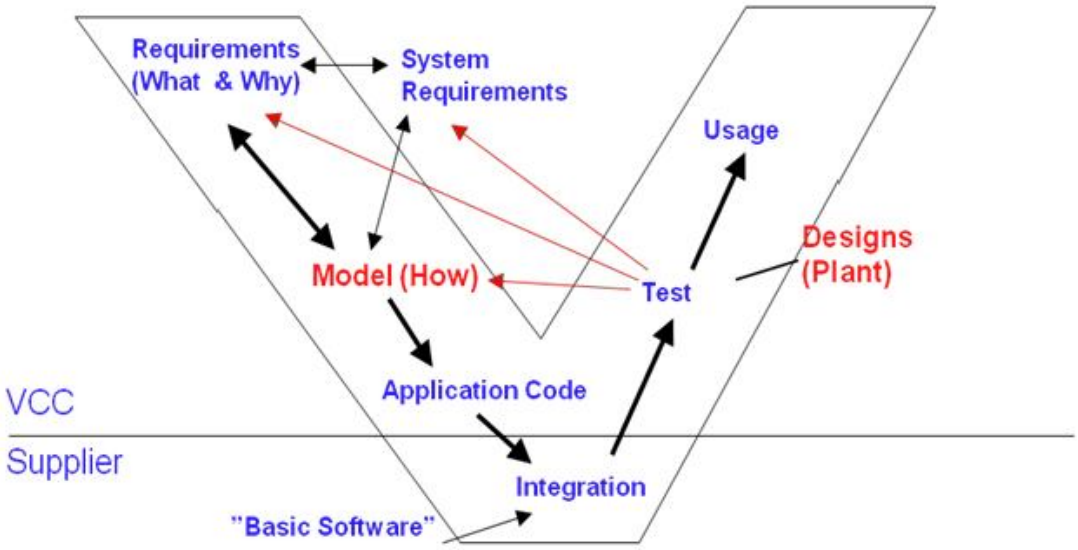


Figure 3.1. Combination of V-Model and Model-Based Design employed in SEM (body domain) group

As it is totally obvious from above diagram (Figure 3.1.), the development cycle is initiated through digging into the text-based system requirements document and creating the counterpart model which visibly demonstrates the entire procedure to satisfy the requirement specifications. The final effort done at VCC is to automatically generate the application code in C programming language. At the supplier side, the software source code is integrated and tested on the hardware model plant. However, validation is constantly carried out throughout the cycle on various artifacts including SRS, model and generated code.

Moreover, the interviewee emphasized that modeled system requirements is not a good potential to substitute the traditional text-based document. The reason is that, the simulated requirement specification may seem to be complicated and ambiguous for the users who do not have sufficient knowledge of simulation environments. However, by incorporation of text into the model, it is probable to completely replace the specification document with the model.

4. Exploring MBD

4.1 Research methodology

To collect the valuable data about the Model-based design approach, we used three qualitative research methods: literature reviews, interviews and observation. The same procedure as V-model was employed for literature reviews and interviews. To avoid the redundancy, we suffice to elaborate only on observation in this section.

In fact, the more focus was placed on studying the requirements specification of seat-belt reminder system to understand its functionalities for further development. At this step the collected data from the literature review was combined with the data from our observation. In our observation we had the chance to ride a real Volvo car to test different conditions and different functions. The data gained from the observation was studied against the specification and it served to give us clear understanding of the whole system. Section three, technical background and development process, summarizes all our findings through literature reviews.

The next sub-section gives explanation on the development stages using MBD approach. It is worth mentioning that this section tries to demonstrate how MBD contributes specifically to the functional development of the safety seat-belt reminder software.

4.2 Developing system using MBD

We are going to implement the modeling and simulation of the safety seat-belt reminder software suing Simulink environment. As a matter of fact, the text-based software requirement specification (SRS) of the mentioned system is converted to an executable model which is extensively more obvious to the development suppliers. The simulated requirement can be straightforwardly tested and modified with no need to resume the entire development cycle. Eventually, the verified model together with the automatic generated code will be delivered to the software supplier to provide them with the truly comprehensive system requirement.

Simulink along with its different integrated libraries is used to implement this project. In order to minimizing the complexity, we organized the system into 3 separate subsystems

each of which is implementation of one variant. There is a parameter called VehicleConfParameter which is set initially on the hardware and determines the variant. Our model consists of both the standard Simulink and Stateflow blocks in order to implement all system functionalities.

Our development approach, as it was previously mentioned, is model-based design with some iterations, each containing implementation of chosen functionalities along with testing. After accomplishing the first iteration, we encountered the following weaknesses:

- The high complexity due to the usage of numerous blocks to implement a simple logic
- The difficulty of tracing the specific functionality against the SRS
- The decreased readability resulted from above items
- The increased total development time

To overcome these drawbacks, the Stateflow chart seemed to be the best alternative to be applied from the next iteration on. Using the Stateflow chart along with the Simulink standard blocks, the complexity and readability are somewhat reduced by the means of the direct interpretation from the state machines in the SRS. Accordingly, the traceability against the SRS is raised since each Stateflow chart in the model can be directly compared to the corresponding state machine in the SRS. The overall development time is declined due to the fact that there is no need to re-design the state machines implying functionalities. Considering all benefits of using Stateflow charts, we came to the conclusion of substitution of some Simulink blocks for Stateflow blocks.
The following example elaborates on the reason of using Stateflow charts.
The figure 4.1 illustrates the implementation of an “if statement” using Simulink blocks.

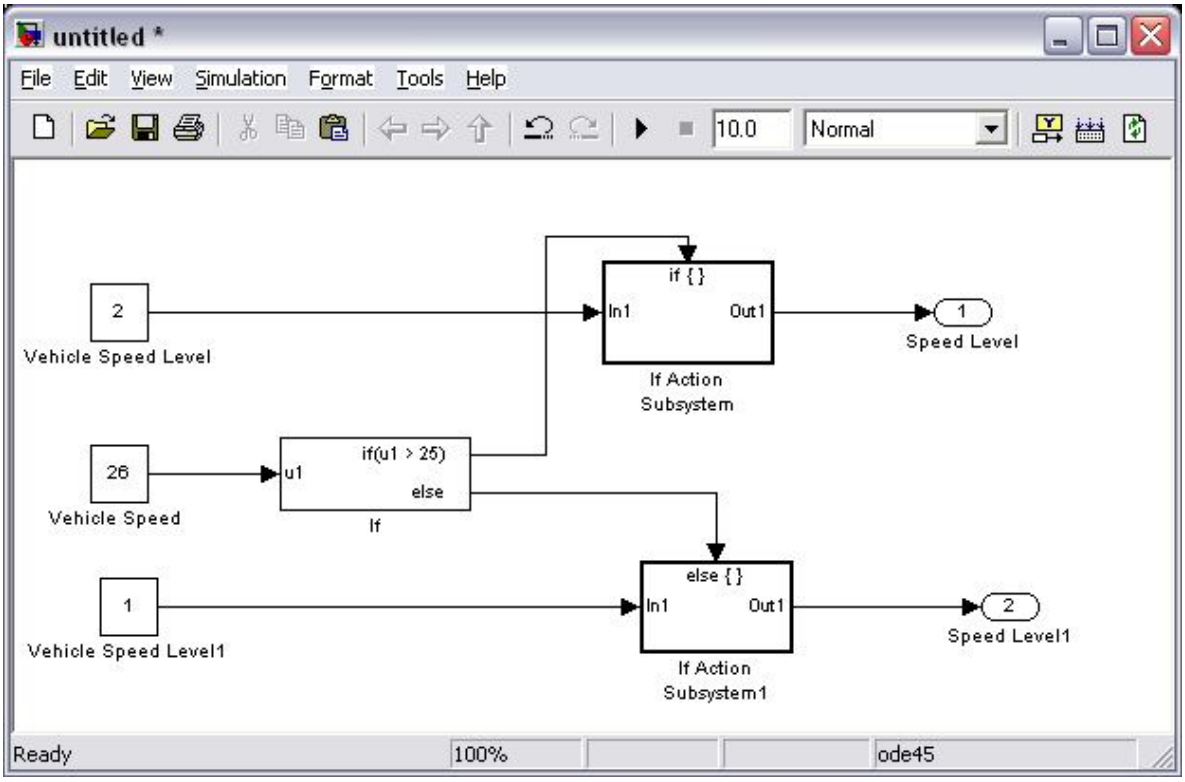


Figure 4.1. Implementation of "if statement" by Simulink blocks

In this model, the vehicle speed is evaluated against the defined condition of greater than 25. If the condition is fulfilled the upper action is performed and the corresponding input (2) is transferred to the output. Otherwise, the lower action subsystem is executed. Obviously, this model can be significantly complicated as the evaluation condition expands.

Alternatively, the same logic can be implemented by means of Stateflow chart as figure 4.2 shows.

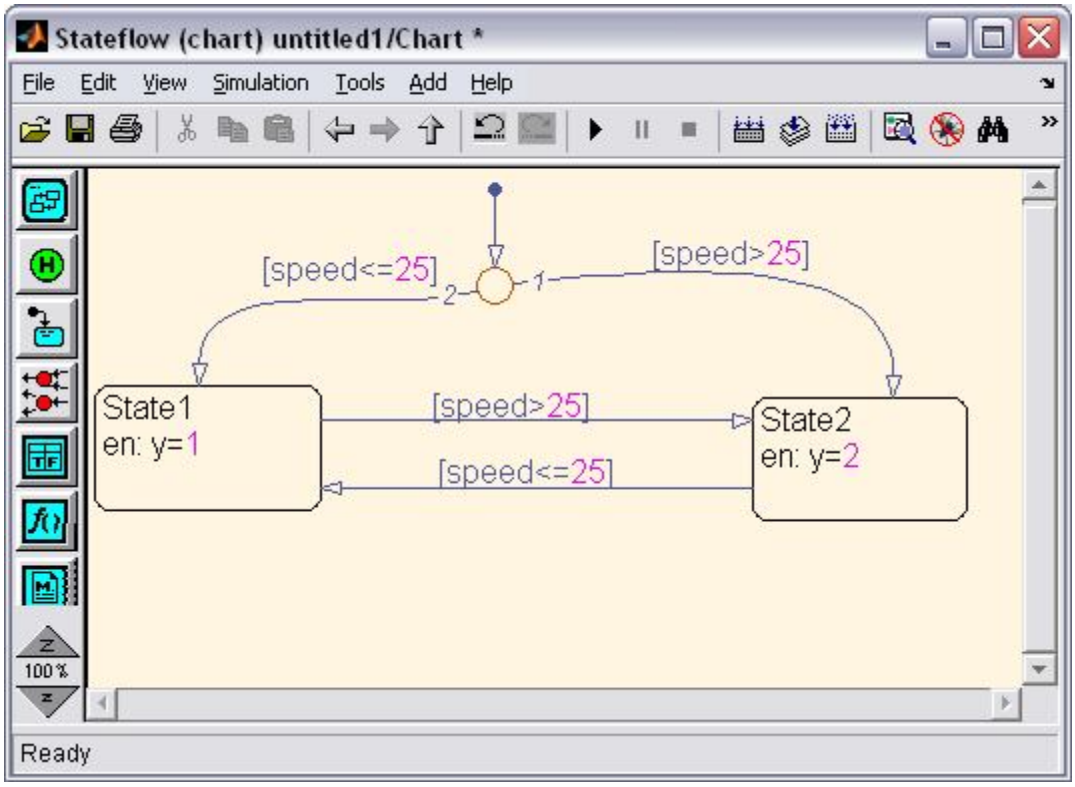


Figure 4.2. The Stateflow chart for implementing the "if statement"

Above Stateflow chart consists of two states each of which is used for setting the output to the appropriate value. The default initial transition is divided into two transitions in accordance with the matching condition.

As one can claims, the Stateflow chart has diminished the complexity as well as the ability to trace the decision procedure among states.

4.2.1 Code generators

The main goal of using code generators is to increase the productivity and quality of the software by directly deriving production code from formal model [18]. Automated code generation reduces the development time as well as human made mistakes. As figure 4.3 demonstrates the procedure for generation of code, the code generator takes the implemented model as an input and produces the C code for further stages.

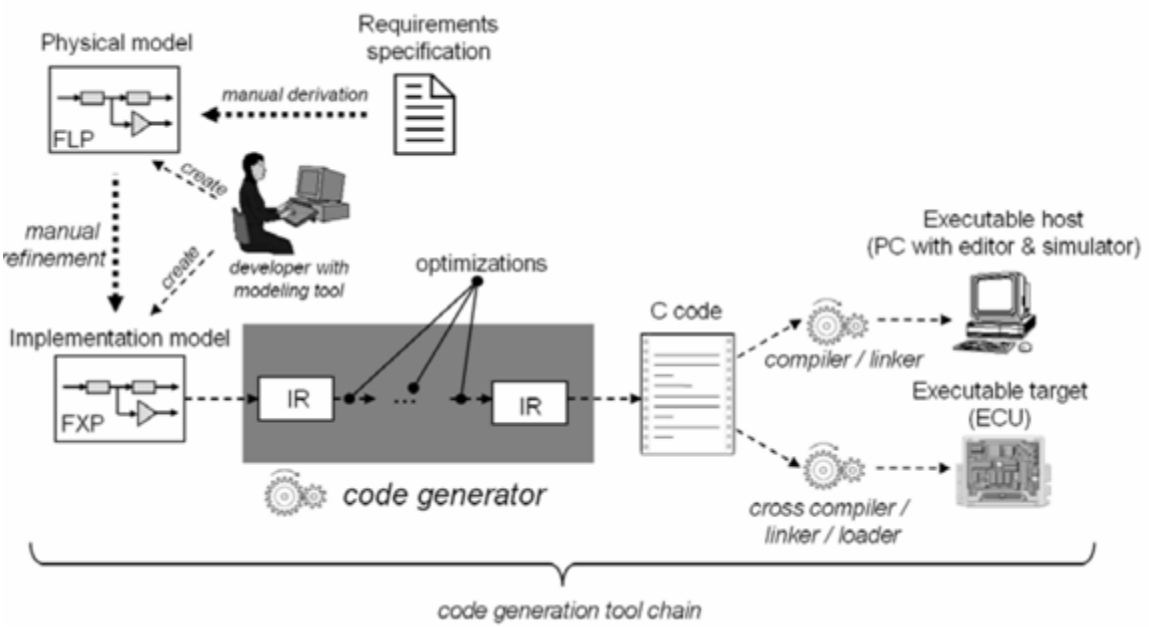


Figure 4.3. Principle of model-based code generation [16]

There are many different tools for generating automatic code in industry. These tools ease the process of producing code by letting the developers automatically generate code from models instead of writing it by hand. Considering a Simulink model, different blocks represent different operations like mathematical or logical operations and the input signals can be counted as variables which might have different values during simulation. Although, using code generation tools doesn't guarantee to produce error free software, however, it decreases number of syntax errors as well as keeping the code consistent with the simulated model. Another advantage of using automatic code generation tool is to have auto generating documentation in each code generation [6].

Among different code generation tools this section introduces the Target Link and the Real-time Workshop Embedded coder and will discuss their advantages and disadvantages. The comparison is based on literature reviews on both tools as well as our own experience trying both tools. Different factors are considered while comparing the tools. Following factors are the criteria to choose the appropriate one: [6]

Available Resources and Manual

Both tools have good manuals and help functions. There are lots of offline and online documentation for the Simulink and MATLAB which give the users good technical support. The MathWorks homepage also provides helpful information about Simulink and Real-Time Workshop Embedded Coder environment with explanation of setting options in different panes.

Target Link, on the other hand, has also help function which provides different settings for the tool but sometimes it gets more difficult to find particular topic using its help function. Overall, considering all the available documentation and helps for both tools it seems that working with Real-Time Workshop Embedded Coder is easier for novice users. The manuals are comprehensive and helpful to cover all needed information.

Supported Blocks

Most of the Simulink blocks are supported by both TargetLink and Real-Time Workshop Embedded Coder with some limitations; however, Real-time Workshop Embedded Coder has fewer limitations than TargetLink. The most important limitation of TargetLink which could affect our model is the lack of support for the user-defined and complex functions like embedded MATLAB function. Because of using these two functions in our model, Real-Time Workshop Embedded Coder is tailored to our model.

Optimizing and Customizing the Generated Code

Real-Time Workshop Embedded Coder is more flexible than TargetLink to let user select different options for optimization of the generated code. When it comes to choosing of full optimization the differences are small, however, in our thesis Real-Time Workshop Embedded Coder is better choice by letting us to try different options and see differences in the generated code.

Customizing code is a way to allow developers to change the code to fulfill their goals. There are different settings in each tool to customize the code. One of the good possibilities which both tools support is adding custom code to the model. This feature permits

developers to add custom code to the model and then generate the custom code along with the rest of the model.

5. Comparison of V-Model versus MBD

Considering all the characteristics previously mentioned for both models, there are some benefits and drawbacks associated to them. This section provides more detailed comparison of the two models based on our literature reviews, interviews and our experience in developing seat-belt reminder using model-based development. The comparison is being achieved according to different aspects within various phases involved in development cycle as well as benefits they can bring to the Volvo Cars. These aspects are listed below.

- Gathering the requirements of seat-belt reminder software,
- Coding,
- Testing,
- Production time and cost,
- Quality

Gathering the requirements of seat-belt reminder

The software requirement specification document is mostly textual based. V-model follows the specification-based development which is the current circumstance at Volvo Car. The problem of using specification-based development is related to the ambiguity of the requirements described in textual format. In other words, there might be misinterpretation of the requirements among the development team, resulting in different understanding of the same requirements.

Model-based requirement specification, in contrast, has less ambiguity in a sense that it provides a model describing the requirements. By simulating the model, the same output is obtained for a set of inputs. Although model-based development brings much more certainty, it has its own drawbacks. According to one of our interviewees in the group SEM using MBD, it is harder to trace the model for an individual requirement and they usually attach the textual specification to the model before delivering to the supplier.

Coding

Depending on the approach which is applied for the development process, the code is generated manually or automatically. Following specification-based development leads to the manual process in coding according to some strict procedures and standards. The written code should be reviewed in respect to the design integrity and compliance [17]. Conversely, model-based development brings the advantage of using automatic code generators which was previously explained.

Testing

As it was previously described, V-model process model drastically strives to correlate the testing and validation stages with the corresponding development phases. Hence, the artifacts resulted from any initial development stage can be separately validated through the counterpart testing phase. Nevertheless, as demonstrated in figure x, the final system is validated against the requirement specification document to pass the customer acceptance test in the late stage of the development cycle. When it comes to the changes, dependant on the essence of the change, it might be required to resume the entire cycle from the requirement analysis to the coding.

Alternatively, the modern embedded systems are concentrating on the testing and verification of the executable requirement model early at design phases when there is no coding. The code-level approaches are substituted by model-level which is achieved by the modeling tools. [7]

Reduction in Time and Cost

The V-model process model focusing on the text-based specification document, takes substantially much more time compared to the circumstance when the model-based approach is employed. Firstly, the time which is spent on writing the specification and communication with the supplier is outstandingly reduced through engaging model-driven method. Because of the vagueness mentioned before, it might be needed to put some time

to resolve the ambiguities with the supplier. Additionally, misinterpretation of the requirement specification might end up with wrong implementation and consequently in increased time of development.

Secondly, there are always new coming requirements that should be considered and put into practice. If the new requirements are introduced in the time period between each release, there is no chance to incorporate them into the current release and Volvo should wait until the next one.

By using model-based development, Volvo has a vast opportunity to test and model the requirements before delivering them to the supplier. Via validation of executable simulated requirements before delivering, Volvo may ensure the correctness of them as well as enclosing the model which is used to generate the automatic code, resulting in cutting down the cost and time of development. Moreover, system models generated by simulation tools can be reused for the next simulation and modeling projects, leading to minimizing the production time.[9]

Quality

"Quality in model-based software development covers the quality of models, modeling languages, modeling tools, modeling processes and even transformations performed on models" [19]. MBD remarkably attempts to enhance the understanding and communication between development team members and the capability of tracing the models, ending up with the improvement of quality. [20]

Although we could not find much quantitative data to support the statement of the direct impact of MBD on the software defects, there are some evidence that shows the reduction of number of software defects through using MBD. [21] In addition, the model is constantly simulated and checked within various iterations with respect to the correctness and completeness. Since model-driven approach presents the automatic code generation, the human fault in coding phase is considerably decreased which results in higher quality. [21]

6. Conclusion

As a matter of fact, the model-based design is a promising approach to complement the present V-model through substituting the modeled requirement for the specification document. In other words, using a modeling environment such as Simulink provides the developers with the simulating and testing the modeled requirement at early stages of development cycle even before any line of code has been produced.

Accordingly, any defects or conflicts in the requirement specification can be detected at design stage before delivering the specification document to the supplier. The result would be offering a specification with less defects and higher level of quality. Furthermore, the market continuously demands new requirements which should be considered in each release. Following model-base design approach serves the integrating and validating those new requirements into the model prior to the supplier starts developing. Moreover, Model-based design offers the opportunity of using automatic code generators like Real-Time Workshop Embedded Coder which in turn facilitates the coding procedure by diminishing the human-made mistakes.

To sum up, engaging model-based design introduces the benefits of cutting down the total cost and time of production by eliminating the unnecessary needs to resume the whole development cycle. Besides, automatic code generation leads to the higher-quality product through the reduction of code defects.

7. Future works

Due to providing the clear overview of the Simulink-based modeled requirement for the stakeholders who are not involved in development, the current thesis work can be followed by designing an appropriate GUI. Hence, the future work can be investigation of usage of existing methods and tools for implementing a user-friendly interface which eliminates the need of Simulink knowledge.

8. References

1. Ian Sommerville. (2008). Software Engineering. 7th edition. Chapter 4
2. Paul F. Smith, Jie Chen, Hongxing Hu. (2007). Model-Based Design Study and Evaluation of New HMI Concepts for Vehicle Multimedia, Climate Control and Navigation Systems, Accessible at : http://www.mathworks.com/mason/tag/proxy.html?dataid=9331&fileid=40511 Model-based Design Study and Evaluation of New HMI Concepts for Vehicle Multimedia, Climate Control and Navigation Systems
3. Friedman, J. (2006). MATLAB/Simulink for Automotive Systems Design, MathWorks, Inc., Natick, MA, This paper appears in: Design, Automation and Test in Europe
4. The manuals available at: http://www.mathworks.com
5. http://www.the-software-experts.de/e_dta-sw-process.htm
6. Michael Beine, Rainer Otterbach and Michael Jungmann.(2004). Development of Safety-Critical Software Using Automatic Code Generation. SAE 2004 World Congress & Exhibition
7. Guoqing Yang, Minde Zhao, Lei Wang; Zhaohui Wu. (2005). Model-based design and verification of automotive electronics compliant with OSEK/VDX, Embedded Software and Systems, 2005. Second International Conference on, vol., no., pp. 7 pp.
8. J. Jensen, Elements of Model-Based Design,(2010). University of California, Berkeley, Technical Memorandum. UCB/EECS-2010-19, February, 2010. Available at: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-19.html
9. http://zone.ni.com/devzone/cda/tut/p/id/4074#toc7
10. Hercog, D., Curkovic, M., Jezernik, K.; (2006). DSP based rapid control prototyping systems for engineering education and research.
11. Boderc: Model-based design of high-tech systems. (2006). Embedded Systems Institute TU/e Campus
12. Dr. Adnan Shaout, Tejas Chhaya. A new process model for embedded systems control for automotive industry, The University Of Michigan

13. http://www.engin.umich.edu/group/ctm/working/mac/simulink_basics/index.htm
14. www.mathworks.com/access/helpdesk/help/pdf.../**simulink**/sl_gs.pdf
15. Bhatt, D.; Hall, B.; Dajani-Brown, S.; Hickman, S.; Paulitsch, M.; (2005), "Model-based development and the implications to design assurance and certification," Digital Avionics Systems Conference, 2005. DASC 2005. The 24th, vol.2, no., pp. 13 pp. Vol. 2, 30 Oct.-3 Nov. 2005,

16. Stuermer, Ingo; Conrad, Mirko; Doerr, Heiko; Pepper, Peter; (2007). "Systematic Testing of Model-Based Code Generators," Software Engineering, IEEE Transactions

17. Forst G., Automatic code generation for safety critical systems. Available at: http://www.ricardo.com/Documents/IA/ControlsandElectronics/Downloads/Autocodegen_paper.pdf , Last viewd: 2009-09-17

18. Whalen, M.W.; Heimdahl, M.P.E.; (1999), "An approach to automatic code generation for safety-critical systems," Automated Software Engineering, 1999. 14th IEEE International Conference on.

19. Mohagheghi, P., Dehlen, V., and Neple, T. (2009), Definitions and approaches to model quality in model-based software development - A review of literature. Accessible at: http://dx.doi.org/10.1016/j.infsof.2009.04.004

20. Werner Heijstek, Michel R. V. Chaudron, The Impact of Model Driven Development on the Software Architecture Process, Leiden Institute of Advanced Computer Science,Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

21. Mohagheghi, P. and Dehlen, V.(2008), Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. Available at: http://dx.doi.org/10.1007/978-3-540-69100-6_31Bookmark

Appendix

Appendix A: Questionnaire

The following questions were asked to address the reasons for employing model-based design:

1. Which development process has ever been followed at VCC?

If any, then

2. How do you evaluate it regarding strengths and weaknesses?
3. Have you ever employed V-Model?

If yes,

4. How do you evaluate it regarding strengths and weaknesses?
5. Which process model (V-model or Model-based design) do you think is the match with Volvo Car system?
6. What are pros and cons of using model-driven approach at VCC?
7. In which groups the mentioned approach is currently used?

Appendix B: SRS review of the seat belt reminder software

According to the market, the seat-belt reminder software document, the requirement specification is categorized into 3 different variants which are Euro Gen4, US and Beltgong.

Each variant includes a number of Stateflow charts to realize different functionalities. In the beginning of simulation, Stateflow chart receives the inputs from the Simulink model, process them and sends the resultant outputs back to the model. The outputs are displayed to the user at the end of simulation time.

Euro Gen 4:

The Euro variant is used for EU market and embodies belt reminder for the driver, front passenger and the rear passenger seats. It consists of the alarm at low, medium and high volume sound. In addition, visual indication signs and text messages are other functionalities which are expected result of the current prototype.

US:

This variant is intended for US market and it works in the same way as Euro version except for the sound that is restricted only to the low volume.

BeltGong:

On the contrary, this variant which is for US market and used in taxi represents the functionalities of the driver seat. More particularly, the output is confined to the low gong sound and the visual indication.

Appendix C: Compressed SRS

Preliminary Knowledge:

In order to describe the requirement specification of safety seat-belt reminder software, a quantity of concepts should be pointed out.

- All input signals are received from the CAN-BUS and the output signals are sent back to the CAN-BUS.
- There is an input named Powermode which determines the status of vehicle engine and gets 9 different numbers each of which is related to a specific state. The number greater than 5 shows the engine is on.

  KeyRecentlyOut, KeyOut , KeyApproved_0 , PostAccessory_0, Accessory_1 are less than 5.

  PostIgnition_1, IgnitionOn_2, Running_2, Crank_3 are greater than 5, so the engine is ON.

- **Definition of the different speed states**

This function is to determine different speed levels which are low, medium, high or very high. These levels are specified as follows while speed is increasing:

· If 0 < vehicle speed <= 10 km/h ==> low level of speed

· If vehicle speed > 10 km/h ==> medium level of speed

· If vehicle speed > 25 km/h ==> high level of speed

· If vehicle speed > 40 km/h ==> very high level of speed

While speed is decreasing:

· If vehicle speed < = 5 km/h ==> low level of speed

· If vehicle speed < = 20 km/h ==> medium level of speed

. If vehicle speed < = 38 km/h ==> High level of speed

1) Beltgong Variant:

This variant is mostly used in taxi cars and is the belt reminder for only the driver's seat. It functions once when the car starts.

- **Control of the belt reminder gong sound**

The gong sound is always played for 6 seconds whenever the car starts and driver is un-buckled.

- **Control of the belt reminder visual indication**

The visual indication sign is always ON for 6 seconds whenever the car starts regardless driver is un-buckled.

2) EU5 GEN4 Variant:

This variant is used for EU (generation 4) markets and consists of belt reminder for the driver, front passenger and the rear seat passengers. The reminder generates alert at 3 sound levels (low, medium and high) according to the speed. VehicleConfParameter is set to 0x10.

It should be cited that each Gong Sound level is equivalent with a number:

Gong OFF: 0

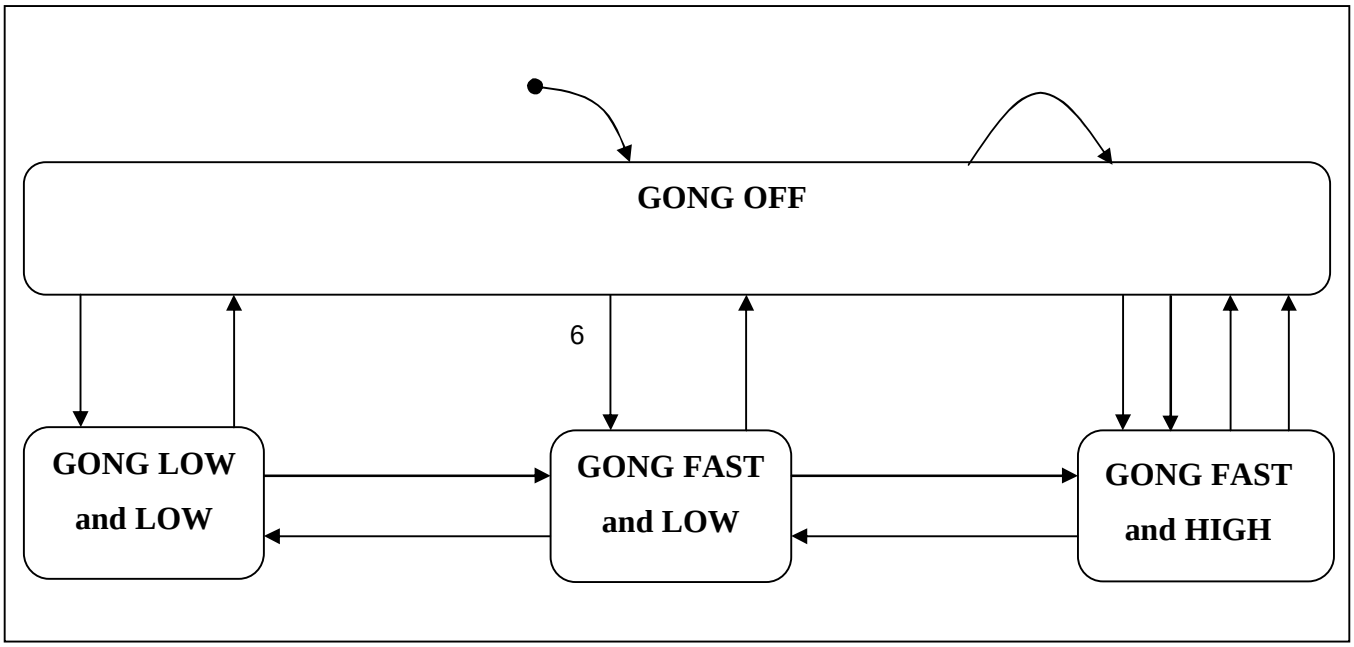
SLOW and LOW Sound level: 1

FAST and LOW Sound level: 2

FAST and HIGH Sound level: 3

The entire functionality is mentioned below:

- **Driver Belt Gong Sound :**



- When the car starts the Gong Sound is OFF

1- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is MEDIUM ➔ SLOW and LOW sound will be played

2- When the Driver Belt is ON **OR** car is OFF **OR** the reversed gear is engaged **OR** the speed level is LOW ➔ Gong Sound will be OFF

3- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is HIGH ➔ FAST and LOW sound will be played

4- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is MEDIUM ➔ SLOW and LOW sound will be played

5- When the Driver Belt is ON **OR** car is OFF **OR** the reversed gear is engaged **OR** the speed level is LOW ➔ Gong Sound will be OFF

6- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is HIGH ➔ FAST and LOW sound will be played

7- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is VERY HIGH ➔ FAST and HIGH sound will be played

8- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is (HIGH or MEDIUM or LOW) ➔ after 90 seconds the sound will change to FAST and LOW sound

9- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is VERY HIGH ➔ FAST and HIGH sound will be played

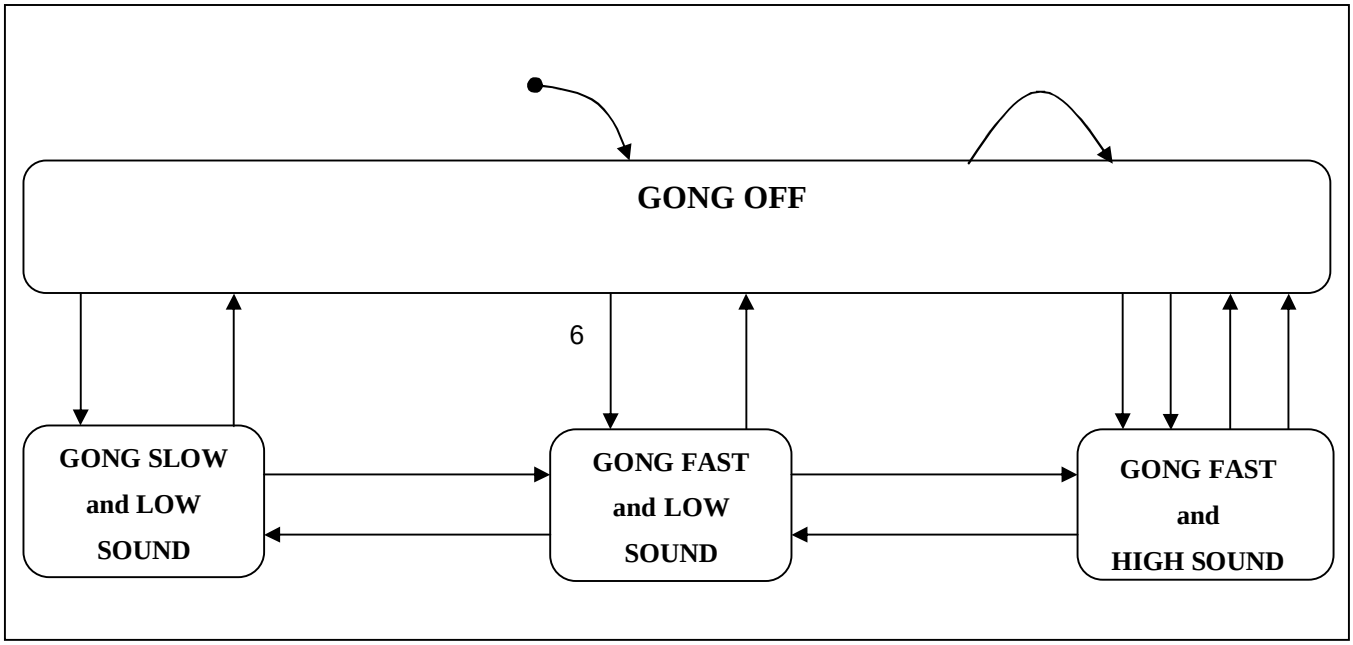
10- When the Driver Belt is ON **OR** car is OFF ➔ the Gong Sound will be OFF

11- When the reversed gear is engaged ➔ the Gong Sound will be OFF

12- When the Driver Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is (MEDIUM or HIGH or VERY HIGH) ➔ FAST and HIGH sound will be played

13- When (the Driver Belt is ON **AND** status of rear seats are NOT warning) **OR** the car is OFF ➔ the Gong Sound will be remained OFF

- **Passenger Belt Gong Sound :**



- When the car starts the Gong Sound is OFF

1- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is MEDIUM ➔ SLOW and LOW sound will be played

2- When the Passenger Belt is ON **OR** car is OFF **OR** the reversed gear is engaged **OR** the speed level is LOW ➔ Gong Sound will be OFF

3- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is HIGH ➔ FAST and LOW sound will be played

4- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is MEDIUM ➔ SLOW and LOW sound will be played

5- When the Passenger Belt is ON **OR** car is OFF **OR** the reversed gear is engaged **OR** the speed level is LOW ➔ Gong Sound will be OFF

6- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is HIGH ➔ FAST and LOW sound will be played

7- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is VERY HIGH ➔ FAST and HIGH sound will be played

8- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is (HIGH or MEDIUM or LOW) ➔ after 90 seconds the sound will change to FAST and LOW sound

9- When the Passenger Belt is OFF **AND** Passenger is seated **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is VERY HIGH ➔ FAST and HIGH sound will be played

10- When the Passenger Belt is ON **OR** car is OFF ➔ the Gong Sound will be OFF

11- When the reversed gear is engaged ➔ the Gong Sound will be OFF

12- When the Passenger Belt is OFF **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is (MEDIUM or HIGH or VERY HIGH) ➔ FAST and HIGH sound will be played

13- When the Passenger Belt is ON **OR** the car is OFF ➔ the Gong Sound will be remained OFF

Before this state machine can be executed, the WARNING input which determines if there is any alarm for rear seats should be resulted from other machines:

- **Rear Right Seat (RBR_SeatWarning)**

If the Rear Right Seat is buckled, the RBR_SeatWarning is set to False showing that there should not be any alert for this seat.
If this seat is un-buckled, the RBR_SeatWarning is set to True.
If the door is changed to closed from open state, the the RBR_SeatWarning is set to False. It means even if the seat belt is OFF, triggering door will remove the alarm.

- **Rear Middle Seat (RBM_SeatWarning)**

If the Rear Middle Seat is buckled, the RBM_SeatWarning is set to False.
If this seat is un-buckled, the RBM_SeatWarning is set to True.
If the door is changed to closed from open state, the the RBM_SeatWarning is set to False.

- **Rear Left Seat (RBL_SeatWarning)**

If the Rear Left Seat is buckled, the RBL_SeatWarning is set to False.
If this seat is un-buckled, the RBL_SeatWarning is set to True.
If the door is changed to closed from open state, the the RBL_SeatWarning is set to False.

- **front Driver Seat (FBD_SeatWarning)**

If the Driver is buckled, the FBD_SeatWarning is set to False.
If this seat is un-buckled, the FBD_SeatWarning is set to True.

- **Front Passenger Seat (FBP_SeatWarning)**

If the Passenger is buckled, the FBP_SeatWarning is set to False.
If this seat is un-buckled, the FBP_SeatWarning is set to True.

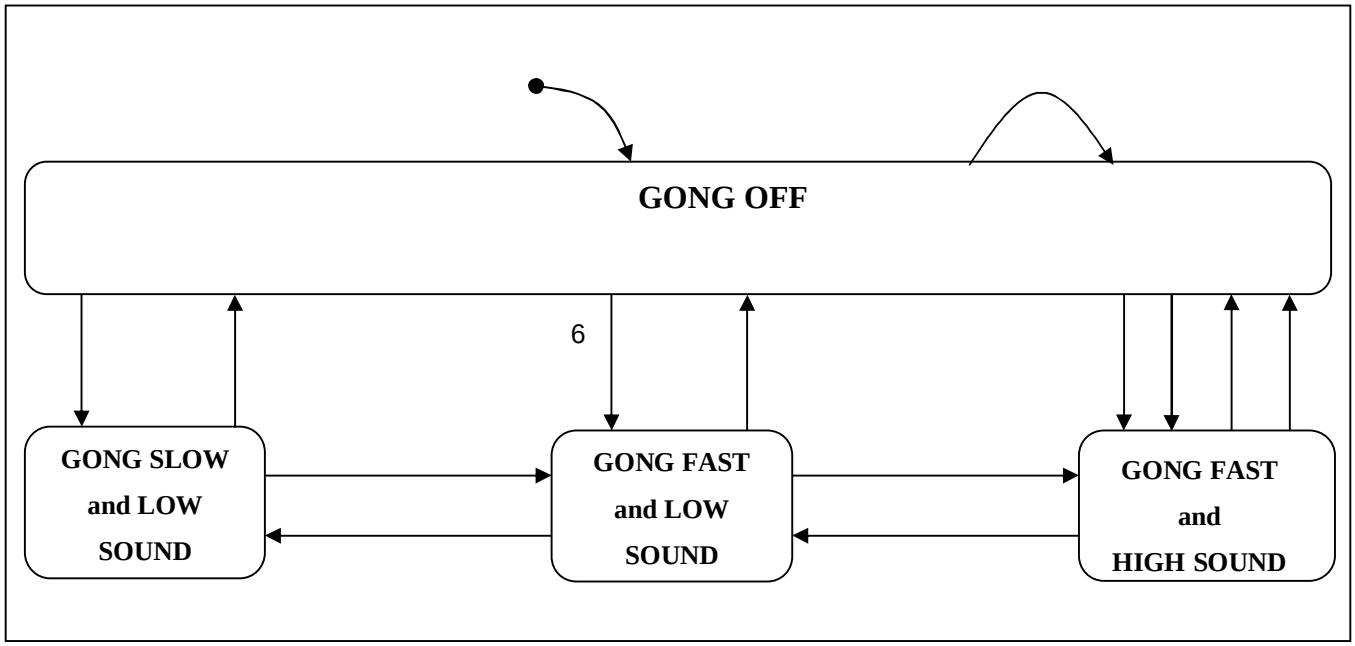
- **Logics of RearSeatStatus**

If the SeatWarning inputs for all rear seats are False, the state machine is remained in General condition.

If any of SeatWarning inputs changes from False to True, the state is Warning.
Pushing the Read Message button causes the state changes from Warning to General.

Consequently, the Warning derived from this machine is used as input to the next state machine.

- **Rear Belt Gong Sound :**



- When the car starts the Gong Sound is OFF

1- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is MEDIUM ➔ SLOW and LOW sound will be played

2- When the Rear seat status is NOT WARNING **OR** car is OFF **OR** the reversed gear is engaged **OR** the speed level is LOW ➔ Gong Sound will be OFF

3- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is HIGH ➔ FAST and LOW sound will be played

4- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is MEDIUM ➔ SLOW and LOW sound will be played

5- When the Rear seat status is NOT WARNING **OR** car is OFF **OR** the reversed gear is engaged **OR** the speed level is LOW ➔ Gong Sound will be OFF

6- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is HIGH ➔ FAST and LOW sound will be played
7- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is VERY HIGH ➔ FAST and HIGH sound will be played
8- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is (HIGH or MEDIUM or LOW) ➔ after 90 seconds the sound will change to FAST and LOW sound
9- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is VERY HIGH ➔ FAST and HIGH sound will be played
10- When the Rear seat status is NOT WARNING **OR** car is OFF ➔ the Gong Sound will be OFF
11- When the reversed gear is engaged ➔ the Gong Sound will be OFF
12- When the Rear seat status is WARNING **AND** car is ON **AND** the reversed gear is not engaged **AND** the speed level is (MEDIUM or HIGH or VERY HIGH) ➔ FAST and HIGH sound will be played
13- When the Rear seat status is NOT WARNING **OR** the car is OFF ➔ the Gong Sound will be remained OFF

- **Unbuckled Alert**

When the vehicle is ON and the belt alert status for any of seats is 1, the FAST and HIGH level of sound will be played for 5 seconds.
The belt alert status resulted from following state machines:

- **Belt-Alert-Status for Driver:**

When the FBD-SeatWarning is changed from False to True and the reversed gear is not engaged and car is ON, the belt alert status for driver is set to 1.

- **Belt-Alert-Status for Passenger:**

When the FBP-SeatWarning is changed from False to True and the reversed gear is not engaged and car is ON, the belt alert status for passenger is set to 1.

- **Belt-Alert-Status for Rear Left seat:**

When the RBL-SeatWarning is changed from False to True and the reversed gear is not engaged and car is ON, the belt alert status for rear left seat is set to 1.

- **Belt-Alert-Status for Rear Right seat:**

When the RBR-SeatWarning is changed from False to True and the reversed gear is not engaged and car is ON, the belt alert status for rear right seat is set to 1.

- **Belt-Alert-Status for Rear Middle seat:**

When the RBM-SeatWarning is changed from False to True and the reversed gear is not engaged and car is ON, the belt alert status for rear middle seat is set to 1.

The final Gong Sound is resultant of the Gong Sound level derived from:

- Driver Belt Gong Sound
- Passenger Belt Gong Sound
- Rear Belt Gong Sound
- Unbuckled Alert

The output of the first three machines will be compared together so that the highest sound level will be chosen. Unbuckled Alert sound dominates the resultant gong sound. That is to say, if there is unbuckled alert, the FAST and HIGH level of sound will be remained for 5 seconds, then the actual sound level will be determined according to the chosen sound level.

- **Control of the Belt Reminder Visual Indication**

When the Driver or the seated Passenger is unbuckled, the visual indication sign is ON.

3) US Variant:

This variant is intended for US markets and consists of belt reminder for the driver, front passenger and the rear seat passengers. The reminder produces merely the alarm of slow gong sound. VehicleConfParameter is set to 0x07.

- **Driver and Passenger Gong Sound:**
  - When the car starts, if the Driver is unbuckled, the SLOW Gong Sound will be played for 6 seconds.
  - While the car is running, if any of Driver, Passenger and Rear seats is unbuckled and the reversed gear is not engaged, , the SLOW Gong Sound will be played for 6 seconds.
  -

It should be stated that the Rear seat status is derived from above-mentioned Logics of RearSeatStatus.

- **Telltale (Visual Indication)**
  - There is always visual indication sign for the 6 seconds when the car starts.
  - While the car is running, if any of Drivers, Passenger and Rear seats is unbuckled and the reversed gear is not engaged, the sign will be turned ON.

- **Text Message Handling**