Evaluation of Model-Based Testing for Embedded Systems based on the Example of the Safety-Critical Vehicle Functions

*Master of Science Thesis in the Software Engineering and Management*

**SHASHA LIU**

University of Gothenburg  
Chalmers University of Technology  
Department of Computer Science and Engineering  
Göteborg, Sweden, October 2012
The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Evaluation of Model-Based Testing (MBT) for Embedded Systems based on the Example of the Safety-Critical Vehicle Functions

SHASHA LIU

© SHASHA LIU, October 2012.

Supervisor: CHRISTIAN BERGER
Examiner: MIROSLAW STARON

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

[Cover:
Embedded systems in vehicle
Provided by Inxee Technologies Company]
Acknowledgements

There are many people I would like to thank for helping me during my thesis work.

First, I would like to express my deepest appreciation to all of my teachers. Especial for my supervisor Assistant Professor Dr. Christian Berger, for his excellent guidance, approachable understanding and pertinent advices he has provided over the entire thesis work. Thanks for my examiner Associate Professor Dr. Miroslow Staron’s useful feedback. I would also like to show my gratitude to my programme coordinator Assistant Professor Dr. Agneta Nilsson, for her guidance of thesis work process. I wish to thank Mr. Joakim Jahlmar, for his writing language support.

Second, I would like to give a special thanks to Inxee Technologies Company, for their support to provide an image that used in my thesis cover.

Finally, I want to show my heartiest gratitude to my friends and families for their love and support. They are always there when I needed them.
Abstract

Along with the announcement of vehicle safety standards, e.g. ISO 26262, ESP and AUTOSAR, embedded systems are used widely to realize the safety function in the automotive domain. Due to the increased number of sensors involved in the system, one important problem to be solved is to obtain enough appropriate test cases to ensure that the implemented system functions are satisfying the software requirements specification.

This thesis describes the systematic literature review performed on Model-Based Testing (MBT) approaches that are available in the automotive domain, mainly focusing on finding the MBT approaches that create models directly from software requirements specification. Furthermore, by applying selected MBT approaches in two conducted running examples of safety-critical functions in the automotive domain, the study shows the advantages and disadvantages of using such approaches. The first running example is the Seat-Belt Reminder System (SBRS) that represents discrete signal processing embedded systems, and the second one is a type of continuous signal processing embedded system called Collision Detection System (CDS).
Table of Contents
1. Introduction & Motivation ................................................................. 7
2. Related Work................................................................................. 9
3. Research Methodology ................................................................. 10
  3.1 Research Goal ......................................................................... 10
  3.2 Research Questions .................................................................. 10
  3.3 Systematic Literature Review .................................................... 10
    3.3.1 Search Strategy ................................................................ 10
    3.3.2 Search Criteria .................................................................. 11
    3.3.3 Search and Selection Process ............................................. 11
    3.3.4 Search Results ................................................................. 12
    3.3.5 Data Extraction Strategy ...................................................... 13
  3.4 Case Study ................................................................................ 17
4. Available Model-Based Testing Techniques ...................................... 18
  4.1 Sequence Based Specification (SBS) ........................................... 18
  4.2 Event Sequence Graph (ESG) .................................................... 19
  4.3 Classification Tree .................................................................. 20
  4.4 Conformance and Fault Injection (CoFI) ...................................... 21
  4.5 UML/OCL ............................................................................... 22
  4.6 Stateflow Automata .................................................................. 22
  4.7 Fault Tree Analyses (FTA) ......................................................... 23
  4.8 Markov Chain ........................................................................ 24
  4.9 Coloured Petri Net (CPN) .......................................................... 24
  4.10 Finite State Machine (FSM) ....................................................... 25
5. Case Study .................................................................................... 26
  5.1 Seat-Belt Reminder System ....................................................... 26
  5.2 Collision Detection System ......................................................... 26
    5.2.1 Assumptions .................................................................... 27
    5.2.2 Operating Principle and Algorithm ..................................... 27
    5.2.3 Formulas ........................................................................ 28
  5.3 MBT Methods Applying Descriptions .......................................... 29
6. Results .......................................................................................... 31
7. Conclusion and Outlook .................................................................. 35
References ......................................................................................... 36
Appendices ......................................................................................... 40
Appendix A --- Seat-Belt Reminder System (SBRS) Functional Requirements .... 40

Page 5 of 46
Appendix B --- Collision Detection System (CDS) Functional Requirements ............................................ 42
Appendix C --- Glossary ............................................................................................................................ 46

List of Figures and Tables

Figure 1 Thesis Scope Overview ................................................................................................................. 7
Figure 2 Search and Selection Process ......................................................................................................... 12
Figure 3 Different Search Queries Results in Different Year ..................................................................... 13
Figure 4 Available MBT Approaches Trend ............................................................................................... 17
Figure 5 Sequence Based Specification Approach Overview ................................................................... 18
Figure 6 Event Sequence Graph Approach Overview .............................................................................. 19
Figure 7 ESG Diagrams ............................................................................................................................... 19
Figure 8 Classification Tree Approach Overview ..................................................................................... 21
Figure 9 CoFI Approach Overview ........................................................................................................... 21
Figure 10 UML/OCL Approach Overview ................................................................................................. 22
Figure 11 Stateflow Automata Approach Overview ................................................................................. 23
Figure 12 Fault Tree Approach Overview .................................................................................................. 23
Figure 13 Markov Chain Approach Overview ........................................................................................... 24
Figure 14 CPN Approach Overview .......................................................................................................... 25
Figure 15 Finite State Machine Approach Overview ................................................................................ 25
Figure 16 Collision Detection System Overview ....................................................................................... 26
Figure 17 CDS definition terms .................................................................................................................. 27
Figure 18 CDS Algorithm Diagram ........................................................................................................... 28

Table 1 Initial Search Results ..................................................................................................................... 12
Table 2 Extracted Data of Selected Papers ................................................................................................. 14
Table 3 Available MBT Approaches .......................................................................................................... 16
Table 4 SBRS System Input Stimuli ........................................................................................................... 29
Table 5 Missing Requirements ..................................................................................................................... 32
Table 6 SBRS Functional Requirements .................................................................................................... 40
Table 7 CDS Functional Requirements ..................................................................................................... 44
1. Introduction & Motivation

Embedded software systems are used widely nowadays to realize comfort and safety functions in vehicles, planes, and trains domain. Along with the increased number of sensors involved in the systems, the development is getting increasingly complex. Hence, in order to ensure a reliable and safe operation, a thorough test is required for validating the expected behavior by the implementation. Furthermore, the important prerequisite of a thorough test depends on the relevant test cases [GG93]. It has been claimed that more than 50% cost for embedded systems development are caused by testing and error correction in the late development stage, and arguable selection of test cases is one of the main reasons [PFH+06]. Therefore, a test engineer is faced with the question of how to find enough appropriate test cases to ensure an effective and efficient thorough testing. As a common and popular solution, Model-Based Testing (MBT) plays an important role in testing automotive embedded systems [CHG12].

In this thesis, the model describes the formal representation of valid and allowed input stimuli sequences combined with expected output values, which can be used to derive test cases. Model-based Testing is an approach to design possible test cases in a platform-independent manner from which platform-specific test cases are derived automatically [UL06]. It is used as a cost-effective approach for embedded systems, especially for the systems in the automotive area. Model Based Testing can detect system under test fault in the very early stage. It also provides requirements traceability [NE08]. In model-based testing, the expected behavior is created as model from the System Under Test (SUT), and the test cases are derived automatically from the model.

The purpose of this study is using a Systematic Literature Review (SLR) to find recent available MBT approaches which are used for validating embedded systems in the automotive domain and to evaluate those approaches with two running examples. The outcome of research is to provide a suggestion for test professionals to choose the proper MBT approaches by considering merit and demerit for generating enough appropriate test cases. This study mainly focuses on the MBT approaches that create models manually directly based on two example specifications to derive test cases, see figure 1.
The paper remainder is organized as follows. Section 2 presents the related work for systematic review on model-based testing. Section 3 describes the research method in detail. Section 4 provides the available MBT approaches information. After that, two running examples are demonstrated in section 5. Section 6 provides the results after evaluating the MBT approaches with running examples. Finally, the conclusion and outlook is shown in section 7.
2. Related Work

Since the 1990s, many model based testing methods has been presented [ZZZ+10], which attracts some researchers to do study on it.

A survey on modeling language shows that behavioral model can be taken from many forms, like diagrams, grammars, tables and control flow graphs etc. Those models have two main functions; one is used to describe the set of stimuli applied to the SUT, the other one is to describe the possible responding system responses to those stimuli. That study provides some guidelines to help in the decision between different types of testing modeling language [HKO06].

Dias-Neto et al. did a systematic review on model-based testing approaches that were published between 1990 and 2006. This research shows that 66% MBT approaches are applied for system testing and they are suitable to support structural testing from software requirements. The investigation indicates that 60% models are derived from software requirements. 23.2% models are described using UML diagrams. UML statechart, class and sequence diagrams are most often used in particular, and 76.7% models are described using non-UML notations that include finite state machine and Z Specification [DSV+07].

A systematic review [DT08] provides supporting the MBT approaches selection for software projects. That study proposed an infrastructure with some activities to provide criteria for choosing MBT approaches. Those activities are software projects characterization, adequacy level and indicators for the selection of MBT approaches, MBT approaches combination charts, and MBT approaches measurement and evaluation.

This study mainly focuses on reviewing the MBT approaches that used for embedded systems, especially in the automotive domain, from 2007 until 2012.
3. Research Methodology

This section illustrates the research goal and the research questions of this study. In this thesis, systematic literature review and case study were used to address the research questions, which help to achieve the research goal.

3.1 Research Goal

This study intends to achieve the following goals:

- Find recent available MBT approaches for validating embedded systems.
- Identify the MBT approaches for validating embedded systems in the automotive domain.
- Evaluate the identified MBT approaches by applying such approaches with two automotive safety functions systems examples.
- Summarize the advantage and disadvantage of applied MBT approaches.

3.2 Research Questions

In order to achieve the goal (see 3.1), the following research questions are listed:

- Which MBT approaches are available?
- Which MBT are applicable for embedded systems?
- What are their particular strength and weaknesses?

3.3 Systematic Literature Review

Systematic Literature Review (SLR) is a method used to identify, evaluate and interpret all available publications relevant to a particular research topic [SSM07]. In this study, a SLR was conducted to identify all available Model-Based Testing (MBT) approaches to validate automotive embedded systems, and to evaluate each selected MBT approach and after that, to interpret the research results.

This study followed an applied search strategy that includes five parts. The first part illustrates the search queries, after that the search resources are listed, and the third part shows how the search queries are applied with search resources, and the fourth section demonstrates the selection process. Finally, the last part provides the results.

3.3.1 Search Strategy

This strategy is used to guide the search for the study. It contains search queries and search resources.

3.3.1.1 Search Queries

The search queries have been produced by breaking down the research questions and topic according to the population and intervention criteria where population means the application area, intervention is the software methodology used to address a specific problem [SSM07]. In this study, the keywords for searching are listed as follows:

- Population: embedded systems, automotive embedded systems, active safety systems and safety critical systems
- Intervention: model based testing approaches
Each search term contains two phases by constructing Boolean ‘AND’, hence, five search queries are conducted as follows:

1) "model based testing" AND approaches
2) "model based testing" AND "embedded systems"
3) "model based testing" AND "automotive embedded systems"
4) "model based testing" AND "active safety systems"
5) "model based testing" AND "safety critical systems"

### 3.3.1.2 Search Resources

This study has used eight digital libraries that are related to software engineering [Tur10] by applying the defined search queries. The digital libraries are listed below:

1) ACM
2) IEEE Xplore
3) SpringerLink
4) ScienceDirect
5) Citeseer
6) Google Scholar
7) Web of science
8) SCOPUS

### 3.3.2 Search Criteria

The exclusive criteria are used to exclude the results that unrelated to the study, whereas inclusive criteria are used to include the relevant results.

#### 3.3.2.1 Exclusive Criteria

- Repeated articles in different libraries
- Duplicated topic from the same author
- Not describe the model based testing itself
- Not related to testing for automotive related systems, e.g. GUI testing, web testing, medical systems, printer and calculator
- Repeated in different search queries results
- Model derived from source code

#### 3.3.2.2 Inclusive Criteria

- Covered systems are related to automotive embedded systems, track-bounded embedded systems and flight related systems
- Model derived from requirement specification

### 3.3.3 Search and Selection Process

In order to obtain the most relevant articles with the research goal (see 3.1) from tens of thousands of results, the entire process followed four steps. First, search queries were applied into digital library by combining two search factors i.e. published between 2007 and 2012 and each string in the paper’s abstract completely, to obtain the initial search results. Second, apply the exclusive criteria to exclude the unrelated results and inclusive criteria to include the related results. Third, extract data from selected final search results. Finally, classify the
papers into classes based on different extracted approaches. The entire procedure is shown in figure 2.

![Figure 2 Search and Selection Process](image)

### 3.3.4 Search Results

902 papers were obtained in total after the first round search by applying the search queries. Table 1 shows that the number of model based testing publications has been increasing in the past five years. Among this, 569 out of 902 (63%) papers were published between 2009 and 2011. 85% publications were found from ACM, IEEE Xplore, Web of Science and SCOPUS digital libraries. Figure 3 demonstrates that there is a dramatic increasing trend for publications on search query “model-based testing” AND “embedded systems” in 2010 and 2011 years.

<table>
<thead>
<tr>
<th>Source</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>No. of papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>20</td>
<td>28</td>
<td>39</td>
<td>47</td>
<td>51</td>
<td>20</td>
<td>205</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>10</td>
<td>17</td>
<td>21</td>
<td>30</td>
<td>36</td>
<td>14</td>
<td>128</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>7</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>14</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>Citeseer</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Google Scholar</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>12</td>
<td>6</td>
<td>39</td>
</tr>
<tr>
<td>Web of science</td>
<td>26</td>
<td>26</td>
<td>52</td>
<td>18</td>
<td>21</td>
<td>8</td>
<td>151</td>
</tr>
<tr>
<td>SCOPUS</td>
<td>34</td>
<td>41</td>
<td>51</td>
<td>66</td>
<td>64</td>
<td>31</td>
<td>287</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>111</td>
<td>129</td>
<td>185</td>
<td>186</td>
<td>198</td>
<td>93</td>
<td>902</td>
</tr>
</tbody>
</table>

Table 1 Initial Search Results
3.3.5 Data Extraction Strategy
This section provides the data extraction strategy in detail. It contains two sub sections. The first part demonstrates the data that extracted from each selected paper according to 8 criteria. Available Model-Based Testing (MBT) approaches extracted from the first sub section (3.3.5.1) is illustrated in the second part (3.3.5.2).

3.3.5.1 Extracted Data
After applying with exclusive and inclusive criteria (see 3.3.2), 27 selected papers have been analyzed. The data has been extracted from each selected paper by using 8 criteria. The reference column indicates the citation of each paper. The detailed information is illustrated in table 2. Due to the space limitation, table 2 has been divided into two sub-tables.

1) Author/Year
2) Testing level : The technique applicable testing level.
3) Applicable Domain : The domain applied for the approaches.
4) Approaches/Techniques : The approaches have been used.
5) Behavior Model : The behavior model used for the approaches.
6) Tool Support : The supported tool mentioned for the approaches.
7) Case Study/Example : Case study or examples provided in the paper.
8) Model Origin: It shows the original source of model that described in the paper, from source code or from requirement specification.
<table>
<thead>
<tr>
<th>Reference</th>
<th>Author/Year</th>
<th>Testing Level</th>
<th>Applicable Domain</th>
<th>Approaches/Techniques</th>
<th>Behavior Models</th>
<th>Tool Support</th>
<th>Case Study/Example</th>
<th>Model Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>[PVA+12]</td>
<td>Pontes et al., 2012</td>
<td>Integration Testing</td>
<td>Space Embedded Software</td>
<td>Conformance and Fault Injection (CoFI)</td>
<td>Finite State Machine (FSM)</td>
<td>Yes (Condata)</td>
<td>On-board Data Handling Satellite Software</td>
<td>Req</td>
</tr>
<tr>
<td>[LPG11]</td>
<td>Lasalle et al., 2011</td>
<td>Not Defined (ND)</td>
<td>Automotive Mechatronic Systems</td>
<td>End to End Test MBE Framework</td>
<td>UML &amp; OCL</td>
<td>Yes (Test Designer™)</td>
<td>Vehicle Front Axle Unit</td>
<td>Req</td>
</tr>
<tr>
<td>[CSV10]</td>
<td>Cristia et al., 2010</td>
<td>Integration Testing</td>
<td>Embedded Systems</td>
<td>Statechart based testing &amp; Z based testing</td>
<td>Statecharts Model, FSM, Z-bases Testing Tree</td>
<td>Yes</td>
<td>Software Embedded into the Payload Data Handling Computer</td>
<td>Req</td>
</tr>
<tr>
<td>[LG10]</td>
<td>Lochau et al., 2010</td>
<td>Integration Testing</td>
<td>Embedded Control Systems</td>
<td>Statechart-like Formalisms</td>
<td>Stateflow Automata</td>
<td>Yes (Matlab/ Simulink)</td>
<td>Car Door Controlling System</td>
<td>Req</td>
</tr>
<tr>
<td>[ABJ+10]</td>
<td>Aichernig et al., 2010</td>
<td>ND</td>
<td>Hybrid Systems</td>
<td>Abstraction</td>
<td>Qualitative Action Systems</td>
<td>Yes</td>
<td>Two – tank System</td>
<td>Req</td>
</tr>
<tr>
<td>[BHK09]</td>
<td>Belli et al., 2009</td>
<td>System Testing</td>
<td>Automotive Electronic Control Units</td>
<td>Event Sequence Graph (ESG)</td>
<td>ESG Graphs, Classification Tree</td>
<td>Yes (CANoe)</td>
<td>Adaptive Cruise Control Unit</td>
<td>Req</td>
</tr>
<tr>
<td>[NE08]</td>
<td>Nakao et al., 2008</td>
<td>System Testing</td>
<td>Embedded systems with safety critical functions</td>
<td>Sequence-Based Specification (SBS)</td>
<td>SBS Model</td>
<td>ND</td>
<td>Car Door Control Unit</td>
<td>Req</td>
</tr>
<tr>
<td>[BS12]</td>
<td>Berger et al., 2012</td>
<td>ND</td>
<td>ND</td>
<td>SBS</td>
<td>SBS Model</td>
<td>ND</td>
<td>Auxiliary Heating System</td>
<td>Req</td>
</tr>
<tr>
<td>[CP07]</td>
<td>Carter et al., 2007</td>
<td>ND</td>
<td>Embedded Control Systems</td>
<td>SBS</td>
<td>SBS Model</td>
<td>Yes (Simulink)</td>
<td>Automotive Power Window</td>
<td>Req</td>
</tr>
<tr>
<td>[L.LB11]</td>
<td>Lasalle et al., 2011</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>SysML4MBT Notation</td>
<td>SysML4MBT model</td>
<td>Yes</td>
<td>Car lighting system</td>
<td>Req</td>
</tr>
<tr>
<td>Author/Year</td>
<td>Reference</td>
<td>Model Origin</td>
<td>Case Study/Example</td>
<td>Tool Support</td>
<td>Applicable Domain</td>
<td>Testing Level</td>
<td>Approaches/Techniques</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td>--------------</td>
<td>--------------------</td>
<td>--------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>Iyenghar et al., 2011</td>
<td>[LPW11]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Junior et al., 2011</td>
<td>[JB11]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Schick et al., 2011</td>
<td>[SH11]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Davies et al., 2011</td>
<td>[DL11]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Flight Critical</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Wu-Hen-Chang et al., 2011</td>
<td>[WAE+11]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Critical</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Mittal et al., 2011</td>
<td>[MWF+11]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Lindlar et al., 2010</td>
<td>[LWW10]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Malik et al., 2010</td>
<td>[MLT10]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Complex Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Ferraro et al., 2010</td>
<td>[FGM+10]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Safety Critical</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Yu et al., 2010</td>
<td>[YX09]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Safety Critical</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Zhao et al., 2010</td>
<td>[ZZZ10]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Safety Critical</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Cartaxo et al., 2008</td>
<td>[CAO+08]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Peleska et al., 2008</td>
<td>[P08]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
<tr>
<td>Killmann et al., 2007</td>
<td>[KH07]</td>
<td>Req</td>
<td>ND</td>
<td>ND</td>
<td>Embedded Systems</td>
<td>ND</td>
<td>UML and UTP</td>
<td></td>
</tr>
</tbody>
</table>
3.3.5.2 Extracted Available MBT Approaches

This study focuses on the approaches that create model from software requirements specification only according to the thesis scope (see figure 1), hence, the papers that describe creating models from source code were excluded. Ten different approaches were obtained from table 2 in total. The detailed information is shown in table 3.

<table>
<thead>
<tr>
<th>Approach/ Technique</th>
<th>Amount of Papers</th>
<th>Paper References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Based Specification</td>
<td>3</td>
<td>[NE08],[BS12],[CP07]</td>
</tr>
<tr>
<td>Event Sequence Graph</td>
<td>1</td>
<td>[BHK09]</td>
</tr>
<tr>
<td>Classification Tree</td>
<td>1</td>
<td>[BHK09]</td>
</tr>
<tr>
<td>Conformance and Fault Injection (CoFI)</td>
<td>1</td>
<td>[PVA+12]</td>
</tr>
<tr>
<td>Unifies Modeling Language(UML)</td>
<td>6</td>
<td>[LPG11],[LPW11],[SHJ11],[MTL10], [CAO+08],[KH07]</td>
</tr>
<tr>
<td>Stateflow Automata</td>
<td>1</td>
<td>[LG10]</td>
</tr>
<tr>
<td>Fault Tree Analysis</td>
<td>1</td>
<td>[KHE11]</td>
</tr>
<tr>
<td>Makov Chain</td>
<td>1</td>
<td>[YXD09]</td>
</tr>
<tr>
<td>Coloured Petri Net</td>
<td>1</td>
<td>[ZZZ+10]</td>
</tr>
<tr>
<td>Finite State Machine</td>
<td>3</td>
<td>[CSV10],[PVA+12],[ WAE+11]</td>
</tr>
</tbody>
</table>

Table 3 Available MBT Approaches

Figure 4 below demonstrates the usage status of MBT approaches used for embedded systems in the past 5 years, i.e. from 2007 until 2012.

- Compared to the survey [DSV+07] from 1999 to 2006, there are many new MBT approaches conducted from 2007 to 2012, but UML and finite state machine are still most often used.
- 2010 and 2011 are the most active years, the reasons might be the following:
  a. ISO 26262 standard “Road vehicle – Functional safety” was published in 2011. It is mandatory during the development of safety functional requirements [ISO12].
  b. Increase of active safety systems in vehicles
     ▪ From 1 Nov 2011, ESP (Electronic Stability Programme) must be equipped to all new car and light commercial vehicle models mandatorily. As the news point out “ESP equipped with all new vehicle models as standard paves the way for increased use of driver assistance systems and sensors that monitor vehicle surrounding” [Rob11].
     ▪ Model-Based Testing is more suitable for validating safety function of braking guards.
  c. Trend of increased usage of modeling techniques [ZZZ+10].
  d. AUTOSAR (AUTomotive Open System ARchitecture) is open and standardized automotive software architecture [AUT12a]. It paves the way for innovative automotive electronic systems that further improve safety [AUT12b].
3.4 Case Study

The case study was used to validate the MBT approaches that obtained from the systematic literature review, with two simplified systems specifications. One represents discrete signal processing embedded system that provides discrete input stimuli, the other one is type of continuous signal processing embedded system that produces continuous input stimuli. By following the procedures of the methodology that described in the paper, the MBT approaches were applied with two running examples. The detailed information is shown in section 5.
4. **Available Model-Based Testing Techniques**

This section provides the brief description of Model-Based Testing (MBT) approaches that extracted from section 3.3.5.2. For easy understanding, each approach is described with the corresponding diagram.

4.1 **Sequence Based Specification (SBS)**

Sequence Based Specification (SBS) is a systematic approach used to ensure the completeness and correctness of the specified requirements in the very early stage of development. This method treats the system as a black-box by only considering the inputs and outputs rather than knowing the internal structure of the system [BS12, BR10].

Figure 5 is used to demonstrate how SBS approach works. First define the input stimuli from the system requirements specification and then organize the stimuli sequences in order by length. Each sequence is given a required response that specified in the requirement specification. Sign λ means empty input, ω represents response for the illegal input stimuli sequences and 0 represents response for the input stimuli that don’t produce any external observable behavior. If a further stimuli sequence, e.g. AB, leaves the system in the same condition with the responses of a previously sequence, e.g. A, then sequence AB is equivalent to A. As shown in figure 5, A is stated in Equiv column of sequence AB. The corresponding requirement for each sequence and its response is noted in Trace column. The input stimuli sequences that are legal or don’t equivalent to any previous sequence are extended by each stimulus. The input stimuli sequences that are illegal or has equivalent relation to another input stimuli are not extended. The model steps stop when there is no more stimuli sequences can be extended [BR10]. In figure 5, the model process stops at sequence length 3, because there is no sequence stimuli can be extended.

![Sequence Based Specification Approach Overview](image-url)
4.2 Event Sequence Graph (ESG)

Event Sequence Graph (ESG) is a technique used to model the interactive systems behavior by using a collection of event sequence graphs. This approach uses a finite set of ESGs to model the desirable behavior of SUT, and then invert each ESG to represent the undesirable behavior algorithmically. Finally, the ESGs and their inventions, called CESG, are used for generating test cases [BNB+05], see figure 6.

An event sequence graph is a directed graph that contains a set of events and their relations, where the events can be divided into two sub-categories: input stimuli and system response. And the incoming arrow with no source and outgoing arrow without target are considered as entry and exit node respectively [BNB+05]. Figure 7 (a) shows the ESG diagram with three events and their interactions. Event A, B and C are connected by arrows, an arrow from A to C means that event C can follow event A. Figure 7 (b) demonstrates the inversion of ESG (figure 7(a)). The Complete Event Sequence Graph (CESG) is made of ESG and its inversion, see figure 7 (c).
ESG approach includes some terminologies [BNB+05] that needed to be known before using. In order to be easily understandable, the following terms will be explained with the help of figure 7 (c).

- **Event Pair (EP):** each edge of the ESG, e.g. AB, CB.
- **Event Sequence (ES):** the sequence of n number of consecutive edges of ESG.
- **Complete Event Sequence (CES):** the ES starts at the entry of the ESG and ends at the exit. The set of CESs specify the system functions, which can be treated as test cases. E.g. AC.
- **Faulty Event Pair (FEP):** Event pair of ESG inversion’s edges, e.g. AA, BC.
- **Faulty Event Sequence (FES):** the sequence of n number of consecutive edges of FESG.
- **Faulty Complete Event Sequence (FCES):** is conducted by set of FEPs, each FEP starts at entry node can be treated as FCES. Furthermore, the FEP doesn’t start at entry node can be extended as FCES by the EP that starts at entry node and its last symbol is the first symbol the FEP. E.g. FEP: BC can be extended as FCES by adding AB, and then ABC is FCES.

In ESG approaches, CES based test cases are proposed to succeed the test whereas FCES based test cases are supposed to fail the test [BNB+05].

ESG approach uses exception handler to execute defense actions for responding the undesirable input event sequences. The system will be brought by appropriate defense action from current state to less risky state when the threats detected. Defense actions are enforced sequences of events, which specified based on the defense matrix. The set of exception handlers and defense matrix are specialized by domain expert according to the risk of the given unexpected behavior. The states risky level is conducted by using risk ordering relation. The risk ordering relation defines the comparison of states’ risky level [BNB+05].

### 4.3 Classification Tree

Classification tree method comes from partition testing, which is used to support the test cases determination in a systematic way [GG93]. According to figure 8, classification tree partitions the input domain of SUT into different classifications according to different aspects, and each classification is continued to be divided until cannot be divided further. All the impartible classes are combined as a table, called combination table, which used to form test cases. The test cases are obtained by selecting combination of different classes [BHK09]. The choosing of combination of classes decides the test cases number. The minimum number requires each class to be used at least once, and the maximum number requires each logical compatible combination of classes as a test case. As a rule of thumb, the minimum should always be satisfied [GG93].
4.4 Conformance and Fault Injection (CoFI)

Conformance and Fault Injection (CoFI) is a systematic way of model-based testing approach used to create test cases for critical software [AMV+06]. It has been applied to space embedded systems traditionally. According to figure 9, there are 3 steps to follow the CoFI method. First of all, identify all the services of System Under Test (SUT) specification. Secondly, create a set of Finite State Machine Models (FSMs) for each service. Each finite state model should represent system services and behavior types under four different input classes. These four different input classes are: normal, specified exceptions, inopportune inputs and invalid inputs caused by hardware faults. Finally, derive test cases from the created models by applying switch cover algorithm that all the reachable paths from the initial state of the model are covered [PVA+12].
4.5 UML/OCL

This tooled approach is proposed to validate automotive mechatronic systems. This method takes UML (Unified Modeling Language) /OCL (Object Constraint Language) model that describe the stimuli of SUT environment as input [LPG11]. In this method (see figure 10), the UML model contains class diagram and object diagram. The class diagram is used to define the static view of environment, which contains entities, the relationships between entities and actions. The object diagram defines the initial value of the entities that represent the environment. OCL formula is used to annotate the class diagram operations, which formalizes the expected behavior [LPG11].

![Figure 10 UML/OCL Approach Overview](image)

4.6 Stateflow Automata

In order to overcome the unexpected safety problem occurred during feature interactions at the system integration level, a MBT method is described for efficiently generating test cases that particularly aim at feature interaction analysis [LG10]. According to figure 11 feature interactions of SUT specification are characterized in a formal way as a functional architecture model that contains set of three types of components. System components part includes input value read by the component, output value changed by the component and internal behavior that used to implement the components functionality. Sensor components contain only output value that used to deliver. Actuator components only have input values that will affect them. And then, the internal behavior of system components will be modeled by using stateflow automata technique. Stateflow Automata technique, as a part of Matlab/Simulink tool set, is a Statechart-like [LG10]. It contains two sub-states, basic states and composite states that include XOR states and AND states. XOR states are used to lead the hierarchical scopes of the states and the AND states are introducing the concurrent sub machines. Each sub state includes the source state, the destination state and their transition relation. From source state to destination state, ECA rules must be followed. E represents the events occur when system triggers the transition. C stands for conditions that needed to be satisfied when transition wants to fire. A represents the action that performed when the transition is taken. The test cases are generated from behavior model stateflow automata model with the help of Matlab/Simulink tool [LG10].
4.7 Fault Tree Analyses (FTA)

FTA is a deductive top-down method that considers information derived from the safety analyses [KHE11]. According to figure 12, fault tree model contains a failure mode as top event. The failure mode contains a set of event set that used to describe the potential safety-critical situation and those situations must be handled by the system. Each event set includes a set of basic events, and these events can either cross the interface or occur inside the system. The basic events can be divided into four types: external, controllable, observable and internal. External events occur out of the system boundary and don’t imply input stimuli and system responses. Controllable events correspond to the sequence of stimuli to the system. Observable events represent condition on the system response. Internal events describe the events that happen inside the system completely, which is the opposite of external events. In order to avoid extremely large fault tree, this method prioritize test scenarios based on their likelihood and impact. The higher critical one will be selected for testing. The test cases are derived from the combination of a behavior model (FSM) and fault tree model [KHE11]. The FSM modeling process please refers to section 4.10.
4.8 Markov Chain

This Model-Based Testing (MBT) approach is proposed to test safety-critical software systems based on safety requirements [YXD09]. According to figure 13, the models are derived from the SUT requirements. The FSM model is derived from the system functional requirements and markov chain model is extracted from the system safety requirements. The detailed information of FSM modeling method, please refer to 4.10. In Markov Chain modeling method, the state space can be divided into three state subsets: Normal State Subset (NSS), Fail-Safe Subset (FSS) and Risky State Subset (RSS). NSS state subsets cover all the predefined safety control functions and all the controlled objects. FSS state subsets include all the definitely abnormal inputs and the caused failures results. RSS state subsets cover all the indefinitely abnormal inputs and the caused failures results. The FSS and RSS are from the field experts and practice [YXD09].

4.9 Coloured Petri Net (CPN)

CPN is an extended Petri Nets which is a graphical and mathematical modeling method proposed by Kurt Jensen. It can be used to model systems with complex procedures for many systems, e.g. communication protocols, distribution systems and automated production [ZZZ+10].

In figure 14, the CPN model contains three main parts: input ports, conditions and output ports. Input ports include the finite set of input data, and output ports is made up of finite set of output data. The conditions have two sub parts: start condition and end condition. Start condition contains set of fusion places ($GF_{SC}, IF_{SC}$ in figure 14) and set of internal input ports (IP). End condition contains set of fusion places ($GF_{EC}, IF_{EC}$ in figure 14) and set of internal input ports (OP). The test cases can be derived from the CPN model by following two rules. The first one is $GF_{SC}$ and $GF_{EC}, IF_{SC}$ and $IF_{EC}$ cannot be empty at the same time. The second one is that the situation ($GF_{SC} = GF_{EC}$) $\cup$ ($IF_{SC} = IF_{EC}$) cannot exist in one test case [ZZZ+10].
4.10 Finite State Machine (FSM)

Finite state machine is used to model the SUT behavior. According to figure 15, finite state model contains three parts: finite set of inputs, state transitions and finite set of responses [WAE+11]. Inputs represent the input stimuli. Transitions are the conditions that cause from one state to another state. The responses indicate the system responses for corresponding input stimuli.
5. Case Study

This section provides the description of two running examples in the automotive domain, one represents discrete signal processing embedded system and the other one represents continuous signal processing embedded system. Their functional requirements are enclosed in Appendix A and B respectively. In this section, the previously described MBT approaches were applied to these two examples.

The applying procedures of those MBT methods with two simplified system cases are shown in section 5.3.

5.1 Seat-Belt Reminder System

The Seat-Belt Reminder System (SBRS) is used to remind the passengers when they are not fastened. The reminder generates a gong alert according to the driver seat-belt buckle status under different conditions. The main function of this system is to process the input data and to present the result as a gong sound. In this case, the SBRS collects input data from the engine, driver seat-belt buckle sensor and wheels: Vehicle front left wheel (Vwfl), Vehicle front right wheel (Vwfr), Vehicle rear left wheel (Vwrl), and Vehicle rear right wheel (Vwrr), speed sensors. The sign(x) indicates the car wheel’s moving direction, i.e. +1 means forward and -1 means backward. The count (sign(x)) displays the car’s moving direction. In the specified specification, four situations are considered:

- Count (sign(x)) >= (+3) means that the car is moving forward
- Count (sign(x)) <= (-3) means the car is moving backward.
- Count(Sign(Vwfl)+sign(Vwfr)+sign(Vwrl)+sign(Vwrr))=(-2) means the car is moving backward towed by other vehicle
- Count(sign(Vwfl)+sign(Vwfr)+sign(Vwrl)+sign(Vwrr))=(+2) means the car is moving forward towed by other vehicle

5.2 Collision Detection System

The Collision Detection System (CDS) consists of a sensor that is placed in front of the vehicle and a reminder which is shown in figure 16. The sensor detects the distance with the front car, and the reminder is used to warn the driver to avoid the crash with front car. The CDS collects input data from the front sensor and wheels speed sensors.

The main function of this system is to process the input data to provide a predicted trend of the car’s driving safety situation in next time points and to present the result as a warning sound to caution the driver to take actions from an unsafe situation to a safe situation.
5.2.1 Assumptions

The CDS example that defined in the specification is used for the academic research purpose only instead of practical implementation. Therefore, the following assumptions are made as follows:

- The two vehicles are moving forward toward the same direction and one vehicle drives after another one straightly, see figure 16.
- The front sensor can detect the relative distance with the front vehicle perfectly without deviation
- The vehicle wheel speed sensors can detect the velocity perfectly without deviation
- Reaction time for the driver to take actions during emergency situation is 1 second
- The deceleration for the vehicle is $7\,\text{m/s}^2$
- The coefficients are chosen arbitrarily for the sake of simplicity
- All the values involved are ideal and no uncertainties are considered

5.2.2 Operating Principle and Algorithm

The CDS uses the 3 latest time points’ relative distance $d$ between the front and the rear vehicles, and their velocities, i.e. $V_f$ and $V_r$, information, to predict the next 3 time points’ trend of car’s driving safety situation. In order to be more understandable, $\{t_{n-2}, t_{n-1}, t_n, t_{n+1}, t_{n+2}, t_{n+3}\}$ is used for representing the time points in the following sections, $t_{n-2}, t_{n-1}$ are latest time points, $t_n$ is the current time point and $t_{n+1}, t_{n+2}, t_{n+3}$ are next future time points. The entire simplified algorithm works in the following steps:

1. Obtain the relative distance $d$ and $V_r$ of the $t_{n-2}, t_{n-1}, t_n$
2. Obtain the $TTC, \Delta TTC, tp_{\text{crash}}, t_{\text{brake}}, tp_{\text{braking}}$ and $tp_{\text{warning}}$ (see figure 17) by applying related formulas, shown in below formulas section, respectively.
3. Calculate $\Delta TTC_{pre}$ of $t_n$
4. Predict $t_{n+1}$’s $TTC_{pre}$, $t_{brake_{pre}}$, $tp_{braking_{pre}}$ and $tp_{warning_{pre}}$
5. Repeat step 3 and 4 until get all the required information of $t_{n+2}$ and $t_{n+3}$
6. Check $TTC_{pre}$ at different time points, e.g. $TTC_{pre_{t_{n+1}}}$ means $TTC_{pre}$ at time point $t_{n+1}$. And then compare the corresponding $t_{crash_{pre}}$ of with the relevant time point.

The detailed procedures are shown in figure 18

![Figure 18 CDS Algorithm Diagram](image)

### 5.2.3 Formulas

- $TTC = \frac{d}{V_r-V_f}$ (\(<0\) : safe situation ; \(>0\): unsafe situation)
- $\Delta TTC = TTC_n - TTC_{n-1}$
- $TTC_{pre} = (0.5 \ast (\Delta TTC_t - \Delta TTC_{t_{n-1}}) + 0.375 \ast (\Delta TTC_{t_{n-1}} - \Delta TTC_{t_{n-2}}) + 0.125 \ast (\Delta TTC_{t_{n-2}} - \Delta TTC_{t_{n-3}})) + TTC$
- $\Delta TTC_{pre} = (0.5 \ast (\Delta TTC_{t_n} - \Delta TTC_{t_{n-1}}) + 0.375 \ast (\Delta TTC_{t_{n-1}} - \Delta TTC_{t_{n-2}}) + 0.125 \ast \Delta TTC_{t_{n-2}} - \Delta TTC_{t_{n-3}} + \Delta TTC)$

(Note: 0.5, 0.375 and 0.125 are probability distribution, and the sum of them is 1. The probability distribution here is used for the example only. The latest time point gets the highest weight, i.e. 0.5, and the older time point gets the weight followed by decreasing 0.125)
\[ V_{pre} = (0.5*(V_n - V_{n-1}) + 0.375*(V_{n-1} - V_{n-2}) + 0.125*(V_{n-2} - V_{n-3})) + V_n \]

\[
t_{brake} = \frac{v}{a} \quad (a: \text{acceleration})
\]

\[
tp_{crash} = TTC + t_n
\]

\[
tp_{crash_{pre}} = TTC_{pre} + t_n
\]

\[
tp_{braking} = tp_{crash} - t_{brake}
\]

\[
tp_{warning} = tp_{braking} - t_{reaction}
\]

\[
tp_{warning_{pre}} = tp_{braking_{pre}} - t_{reaction}
\]

5.3 MBT Methods Applying Descriptions

By following the descriptions in section 4, these MBT methods were applied to these two simplified systems examples. In order to show how these MBT methods actually work, this section provides the four methods applying description for the seat-belt reminder system as an example. The remainder methods were not applied with these two cases, the reasons are stated in section 6.

Sequence Based Specification

1. Define input stimuli from the Seat-Belt Reminder System (SBRS) functional requirements specification. The defined input stimuli were shown in table 4.

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Car is on</td>
</tr>
<tr>
<td>CF</td>
<td>Car is off</td>
</tr>
<tr>
<td>F</td>
<td>Car is moving forward</td>
</tr>
<tr>
<td>B</td>
<td>Car is moving backward</td>
</tr>
<tr>
<td>FT</td>
<td>Car is moving forward by towed</td>
</tr>
<tr>
<td>BT</td>
<td>Car is moving backward by towed</td>
</tr>
<tr>
<td>DO</td>
<td>Driver seat-belt is on</td>
</tr>
<tr>
<td>DF</td>
<td>Driver seat-belt is off</td>
</tr>
<tr>
<td>SF</td>
<td>Detected speed &gt;= 4m/s</td>
</tr>
<tr>
<td>SS</td>
<td>Detected speed &lt;4m/s</td>
</tr>
<tr>
<td>DL</td>
<td>Duration &gt;=5s</td>
</tr>
<tr>
<td>DS</td>
<td>Duration &lt;5s</td>
</tr>
</tbody>
</table>

Table 4 SBRS System Input Stimuli

2. Combine the above stimuli in sequences by length. According to the SBRS system functional requirements specification, the system responses rely on the combination of several input stimuli, hence, the combination of input stimulus was be treated as input stimuli sequences in length 1, e.g. (CO,F,DO,SF,DL).

3. Capture the corresponding system responses for each input stimuli sequence and respective linked requirements. The input stimuli sequences without linked requirements marked as missing requirements.

4. Form the modeling table based on the length of input stimuli sequences, there were sixty-four input stimuli sequences defined when the length equals to 1 and only 2 input stimuli
sequences can be extended. Those 2 input stimuli sequences were extended with sixty-four input stimuli sequences respectively.

5. The modeling process stopped at length equals to 2 since there were no input stimuli sequences can be extended anymore.

6. Derive test cases by using the table when input stimuli sequences in length 2.

**Classification Tree**
1. Define expected input aspect and output aspect. The expected input aspect contains five classifications: car status, car moving direction, driver seat-belt, detected speed and duration. Car status partitions into on and off. Car moving direction divides into forward, backward, forward by towed and backward by towed. Driver seat-belt includes on and off. Detected speed contains two classes: greater than or equal to 4 meters per minute and less than 4 meters per minute. Duration partitions into two classes: greater or equal than 5 seconds and less than 5 seconds. Expected output contains gong on and gong off.

2. Combine the impartible classes of expected input and output as combination table.

3. Derive test cases by following test cases determination criteria.

**Finite State Machine**
1. Define input states. $State_{input} = \{\text{CS,MD,DSBS,DS,DU}\}$, where CS means car status, MD stands for moving direction, DSBS means driver seat-belt status, DS represents detected speed and DU means duration. $state_{response} = \{\text{GO,GF}\}$, where GO means gong on and GF means gong off.

2. Capture the transitions from the functional requirements specification.

3. Link the input states and response states by corresponding transitions.

4. Derive test cases from the completed state diagram by finding complete paths that start from initial state to the destination state.

**Event Sequence Graph**
1. Capture expected behavior of SBRS system from the requirements specification, i.e. SBRS system presents a gong sound based on different conditions.

2. Define input stimuli events and system response events.

   $Event_{input} = \{\text{CO,CF,F,B,FT,BT,DO,DF,SF,SS,DL,DS}\}$, description please see table 4.

   $Event_{response} = \{\text{GO,GF}\}$, where GO stands for gong on and GF stands for gong off.

3. Conduct ESG graph by linking the input stimuli events to system response events using arrows based on the SBRS functional requirements specification.

4. Obtain the CES based test cases from the ESG graph.

5. Form ESG inversion graph by inverting the ESG graph.

6. Obtain the FCES from the ESG inversion graph. As the FCESs here only represent the unexpected input stimuli, the system responses are not known in SBRS functional requirements specification. The FCES based test cases could not be derived. Hence, the modeling process stopped.
6. Results

This section provides the results that came out by applying extracted Model Based Testing (MBT) approaches with two running examples. It includes two sub parts, one illustrates the results from the applied MBT approaches, and the other part states the reasons that why the remainder MBT approaches was not applied.

Applied MBT Approaches

There are five criteria that used to demonstrate results for approach.

a) No. of test cases: This is used to indicate that how many test cases are generated after applying the approach.

b) Find missing requirements: This shows the missing requirements in the defined software specification. The missing requirements are defined if there is no system response for the given input stimuli.

c) Requirements coverage: This shows that the requirement coverage status by applying the specific approach.

d) Advantage: This shows the benefit obtained after performing the specific approach.

e) Disadvantage: This illustrates the drawback obtained after applying the specific approach.

- Sequence Based Specification

<table>
<thead>
<tr>
<th>Applied example</th>
<th>SBRS</th>
<th>CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of test cases</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>Finding missing requirements</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Requirements coverage</td>
<td>All</td>
<td>All</td>
</tr>
</tbody>
</table>

**Advantage:**
- Covered all the requirements
- Discovered the missing requirements
- Easy to manipulate, even for the beginners, if the input stimuli are defined well.
- Provide requirements traceability, every stimuli sequence should have a respective requirement’s support, which helps to find missing requirements.
- Provide multiple uses, for requirement validating, or derive test cases.

**Disadvantage:**
- The tester should understand the system requirements very well to define input stimuli, the input stimulus is the foundation of the following steps, the wrong defining will cause the later mistake.
- Not easy to manage the big table for large systems that have many requirements if there is no help from tool.
- A little time-consuming when extend the extendable stimuli.
- It is a challenge to combine many inputs as stimulus, especially for the system that has many inputs, e.g. CDS.
Table 5 Missing Requirements

<table>
<thead>
<tr>
<th>SBRS missing requirements</th>
<th>CDS missing requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Car is on, and moving forward, the driver seat-belt is off, and the detected speed is $\geq 4 \text{m/s}$, and the duration $&lt; 5 \text{s}$, the gong should off</td>
<td>1) When $t_n$’s TTC $&lt; 0$ and $t_{n+1}$’s TTC $&lt; 0$ and $t_{n+2}$’s TTC $&lt; 0$ and $t_{n+3}$’s TTC $\geq 0$ and $t_{n+3}$’s $t_{\text{crash}}$ $&gt; t_{n+3}$, the gong should off</td>
</tr>
<tr>
<td>2) Car is on, and moving forward, the driver seat-belt is off, and the detected speed is $&lt; 4 \text{m/s}$, and the duration $\geq 5 \text{s}$, the gong should off</td>
<td>2) When $t_n$’s TTC $&lt; 0$ and $t_{n+1}$’s TTC $&lt; 0$ and $t_{n+2}$’s TTC $\geq 0$ and $t_{n+2}$’s $t_{\text{crash}}$ $&gt; t_{n+2}$ and $t_{n+3}$’s TTC $&lt; 0$, the gong should be off</td>
</tr>
<tr>
<td>3) Car is on, and moving backward, the driver seat-belt is off, and the detected speed is $\geq 4 \text{m/s}$, and the duration $&lt; 5 \text{s}$, the gong should off</td>
<td>3) When $t_n$’s TTC $&lt; 0$ and $t_{n+1}$’s TTC $&lt; 0$ and $t_{n+2}$’s TTC $\geq 0$ and $t_{n+2}$’s $t_{\text{crash}}$ $&gt; t_{n+2}$ and $t_{n+3}$’s TTC $&lt; 0$ and $t_{n+3}$’s $t_{\text{crash}}$ $&gt; t_{n+3}$, the gong should be off</td>
</tr>
<tr>
<td>4) Car is on, and moving backward, the driver seat-belt is off, and the detected speed is $&lt; 4 \text{m/s}$, and the duration $\geq 5 \text{s}$, the gong should off</td>
<td>4) When $t_n$’s TTC $&lt; 0$ and $t_{n+1}$’s TTC $\geq 0$ and $t_{n+1}$’s $t_{\text{crash}}$ $&gt; t_{n+1}$ and $t_{n+2}$’s TTC $\geq 0$ and $t_{n+3}$’s TTC $\geq 0$ and $t_{n+3}$’s $t_{\text{crash}}$ $&gt; t_{n+3}$, the gong should be off</td>
</tr>
<tr>
<td></td>
<td>5) When $t_n$’s TTC $&gt; 0$ and $t_{n+1}$’s TTC $&lt; 0$ and $t_{n+2}$’s TTC $\geq 0$ and $t_{n+2}$’s $t_{\text{crash}}$ $&gt; t_{n+2}$ and $t_{n+3}$’s TTC $&lt; 0$ and $t_{n+3}$’s $t_{\text{crash}}$ $&gt; t_{n+3}$, the gong should be off</td>
</tr>
<tr>
<td></td>
<td>6) When $t_n$’s TTC $&gt; 0$ and $t_{n+1}$’s TTC $\geq 0$ and $t_{n+1}$’s $t_{\text{crash}}$ $&gt; t_{n+1}$ and $t_{n+2}$’s TTC $\geq 0$ and $t_{n+3}$’s TTC $\geq 0$ and $t_{n+3}$’s $t_{\text{crash}}$ $&gt; t_{n+3}$, the gong should be off</td>
</tr>
<tr>
<td></td>
<td>7) When $t_n$’s TTC $&gt; 0$ and $t_{n+1}$’s TTC $\geq 0$ and $t_{n+1}$’s $t_{\text{crash}}$ $&gt; t_{n+1}$ and $t_{n+2}$’s TTC $\geq 0$ and $t_{n+2}$’s $t_{\text{crash}}$ $&gt; t_{n+2}$ and $t_{n+3}$’s TTC $&gt; 0$, the gong should be off</td>
</tr>
</tbody>
</table>

**Event Sequence Graph**

<table>
<thead>
<tr>
<th>Applied example</th>
<th>SBRS</th>
<th>CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of test cases</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Finding missing requirements</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Requirement coverage</td>
<td>All</td>
<td>All</td>
</tr>
</tbody>
</table>

**Advantage:**
- Helps to find comprehensive test cases, because it covers expected and unexpected behavior.
- It is easy to have the FCES when CES is defined.
- It is good at finding unexpected behavior of system and also providing solution to handle.

**Disadvantage:**
- So many similar terminologies, like EP, ES, CES, FCES etc, it is easy to be confused in the beginning.
- The completed graph looks so complex, especially the relation lines cross each other, which might cause vision problem for big system with complicated requirements.
• It is not easy for future modification.
• It is real confused to find FCES test cases manually, since all the inventions are really complex.
• It cannot guarantee detect all the functional faults. Because the succeed test should be checked that whether the expected results obtained.
• It is quite hard to obtain the test cases manually without the tool help.

• Classification Tree

<table>
<thead>
<tr>
<th>Applied example</th>
<th>SBRS</th>
<th>CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of test cases</td>
<td>34</td>
<td>22</td>
</tr>
<tr>
<td>Finding missing requirements</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Requirement coverage</td>
<td>All</td>
<td>All</td>
</tr>
</tbody>
</table>

**Advantage:**
- Provide minimum and maximum criteria, which helps to obtain reachable number of test cases
- Easy to modify in the combination table
- Entire tree is easy to understand

**Disadvantage**
- Need rich experience and creativity to follow the maximum criteria

• Finite State Machine

<table>
<thead>
<tr>
<th>Applied example</th>
<th>SBRS</th>
<th>CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of test cases</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>Finding missing requirements</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Requirement coverage</td>
<td>All</td>
<td>All</td>
</tr>
</tbody>
</table>

**Advantage:**
- Easy to use for modeling system behavior
- Can be integrated with other methods easily

**Disadvantage**
- Can’t guarantee to detect all the functional faults

**Not Applied MBT Approaches**
The following approaches were not applied with running examples, the corresponding reasons are stated.

• UML/OCL: this approach is presented to model the behavior of the SUT environment rather than the behavior of the SUT. This thesis works on studying modeling approach for modeling SUT behavior.
• Stateflow Automata: this technique is proposed to test the feature interaction. In the defined running examples, each example has one feature only. It is not proper to apply this technique.
• Fault Tree Analyses: this approach focuses on failure mode analysis, which is not defined in the running example specification.
- Markov Chain: this approach needs system internal structure knowledge. This work focuses on black box testing approaches that do not consider system internal structure.
- Coloured Petri Net: this approach should be applied by knowing the internal structure of the SUT.
- CoFI: this approach specifies four classes of finite state machine: (1) normal behavior, (2) specified exceptions, (3) inopportune inputs and (4) hardware faults. This evaluation only considers the class (1), as the provided running example specification is defined under assumption without exception and unexpected inputs consideration. And there is no dependency on hardware in this study as well. Hence, the evaluation result is the same as finite state machine.

In Applied MBT Approaches subsection, four methods are applied with running examples. Sequence Based Specification is the only method that found the missing requirements of the software specification because it provides requirements traceability. There is no exact number of test cases derived for event sequence graph, because due to the specification limitation, i.e. only contains functional requirements, there is no response for unexpected input stimuli. Hence, the test cases cannot be derived successfully.

According to the analysis of the results, the following considerations are conducted:

- Sequence based specification can be considered as the basic behavior model that used to integrated with fault based analysis method, e.g. fault tree analyses. And sequence based specification method is strongly recommend as requirements specification validation method.
- Sequence based specification and event sequence based method are recommend for beginners, because they are intuitive to use without requiring much expert knowledge, and also they provide guidance during modeling process.
- Classification tree is pretty good for modeling small size system. However, this method needs experience and creativity to derive test cases according to its maximum criteria.
- Finite state machine approaches are used widely in model based testing by integrating with other approaches, e.g. CoFI, Markov chain and fault tree analyses.
- Event sequence graph is recommended to test fault mode of system under test.
7. Conclusion and Outlook

The intention of this thesis is to find the available Model-Based Testing (MBT) approaches for validating automotive embedded systems from publications and to provide the advantage and disadvantage of those approaches by evaluating such approaches with two running examples.

This work presents the findings from the systematic literature review. 5 search queries were applied in 8 digital libraries. 10 MBT approaches that used for automotive embedded systems were obtained. Those MBT approaches principles are displayed with corresponding graphical demonstration.

The study provides functional requirements of two conducted safety-critical functions in the automotive domain as running examples. One is called Seat-Belt Reminder System that represents discrete signal processing embedded systems. The other one is a representation of continuous signal processing embedded system called Collision Detection System. As these two running examples are defined for academic research instead of practical implementation, their specified requirements were defined under ideal assumptions. For instance, the involved parameters are simplified. However, their functional requirements can be considered as suggestive reference. This research also provides the evaluation results after applying such approaches with two running examples. The advantages and disadvantages are presented, which can be helpful in selecting proper approaches for validating automotive embedded systems.

Due to the scope limitation of the thesis, this work focused on the early stage of model based testing, i.e. manually creating models from system requirements specification and deriving abstract test cases. The future work could be to conduct research on transforming the abstract test cases into executable test scripts with the help of tools. As the present study applied and evaluated MBT approaches on fictive simplified system specifications, other future work could include doing research on evaluating those MBT approaches on real systems.
References


Page 36 of 46


Appendices

Appendix A --- Seat-Belt Reminder System (SBRS) Functional Requirements

This section provides the gong functionality of seat-belt reminder system. The gong functionality informs the driver if he/she is unbuckled. Table 6 demonstrates how gong sounds are performed based on specific conditions, as well as, detail description in below.

<table>
<thead>
<tr>
<th>Gong sound</th>
<th>Car</th>
<th>Sign(x)</th>
<th>Count(sign(x))</th>
<th>Driver seat-belt</th>
<th>Avg (Detected Speed)</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sign (Vwfl)</td>
<td>Sign (Vwfr)</td>
<td>Sign (Vwrl)</td>
<td>Sign (Vwrr)</td>
<td>+</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>+1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>+1</td>
<td>0</td>
<td>+1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>0</td>
<td>+1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1</td>
<td>0</td>
<td>+1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6 SBRS Functional Requirements
A gong sound shall be on to remind the driver based on the following different conditions:

- The car is on and the count(sign(x))\geq (+3) and the drive seat-belt is off and detected speed is equal or greater than 4m/s and the speed duration lasts for 5 seconds.
- The car is on and the count(sign(x))\leq (-3) and the drive seat-belt is off and detected speed is equal or greater than 4 m/s and speed duration lasts for 5 seconds.

A gong sound shall be off based on the following different conditions:

- The car is off
- The car is on and driver seat-belt is on
- The car is on and count (sign(x)) \leq (-3) and detected speed less than 4m/s and speed duration lasts for 5 seconds.
- The car is on and count (sign(x))\geq (+3) and detected speed less than 4m/s and speed duration lasts for 5 seconds.
- The car is on and car is moving backward towed by other vehicle, which the Count(Sign(Vwfl)+sign(Vwfr)+sign(Vwrl)+sign(Vwrr))=(-2)
- The car is on and car is moving forward towed by other vehicle, which Count(Sign(Vwfl)+sign(Vwfr)+sign(Vwrl)+sign(Vwrr))=(+2)
### Appendix B --- Collision Detection System (CDS) Functional Requirements

Table 7 provides the functional requirements of CDS, and it demonstrates how warning sound is performed based on specific conditions, as well as, detailed description in below.

<table>
<thead>
<tr>
<th>voice warning sound</th>
<th>( T TC )</th>
<th>( T TC_{pre} )</th>
<th>( t p_{crash_pre} )</th>
<th>( t n )</th>
<th>( t n+1 )</th>
<th>( t n+2 )</th>
<th>( t n+3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>On (( t n ))</td>
<td>( &gt;0 )</td>
<td>( &gt;0 )</td>
<td>( &lt;t n+1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T TC )</td>
<td>( T TC_{pre} )</td>
<td>( t p_{crash_pre} )</td>
<td>( &lt;0 )</td>
<td>( &gt;0 )</td>
<td>( &lt;t n+1 )</td>
<td>( &lt;t n+2 )</td>
</tr>
<tr>
<td>On(( t n+1 ))</td>
<td>( &gt;0 )</td>
<td>( &lt;0 )</td>
<td>( &gt;0 )</td>
<td>( &gt;0 )</td>
<td>( &lt;t n+1 )</td>
<td>( &lt;t n+2 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T TC )</td>
<td>( T TC_{pre} )</td>
<td>( t p_{crash_pre} )</td>
<td>( &lt;0 )</td>
<td>( &gt;0 )</td>
<td>( &lt;t n+1 )</td>
<td>( &lt;t n+2 )</td>
</tr>
<tr>
<td>On(( t n+2 ))</td>
<td>( &gt;0 )</td>
<td>( &lt;0 )</td>
<td>( &lt;0 )</td>
<td>( &gt;0 )</td>
<td>( &lt;t n+1 )</td>
<td>( &lt;t n+2 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$TTC$</td>
<td>$TTC_{pre}$</td>
<td>$tp_{crash_pre}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;tn+1$</td>
<td>$&gt;tn+2$</td>
<td>$&lt;tn+3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;tn+2$</td>
<td>$&lt;tn+3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;tn+1$</td>
<td>$&gt;tn+2$</td>
<td>$&gt;tn+3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;tn+2$</td>
<td>$&lt;tn+3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;tn+3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Off**

<table>
<thead>
<tr>
<th></th>
<th>$TTC$</th>
<th>$TTC_{pre}$</th>
<th>$tp_{crash_pre}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&gt;tn+1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&gt;tn+2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&lt;tn+2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
</tr>
<tr>
<td>$&lt;tn+3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&gt;tn+1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
<td>$&lt;0$</td>
</tr>
<tr>
<td>$&gt;tn+1$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A warning sound shall be provided to remind the driver based on the following different conditions:

The sound should be displayed on time point $t_n$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} < t_{n+1}$
- When $t_n$’s $TTC < 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} < t_{n+1}$

The sound should be displayed on time point $t_{n+1}$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} > t_{n+1}$ and $t_{n+2}$’s $TTC_{pre} > 0$ and $t_{n+2}$’s $tp_{crash_{pre}} < t_{n+2}$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} < 0$ and $t_{n+2}$’s $TTC_{pre} > 0$ and $t_{n+2}$’s $tp_{crash_{pre}} < t_{n+2}$
- When $t_n$’s $TTC < 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} > t_{n+1}$ and $t_{n+2}$’s $TTC_{pre} > 0$ and $t_{n+2}$’s $tp_{crash_{pre}} < t_{n+2}$
- When $t_n$’s $TTC < 0$ and $t_{n+1}$’s $TTC_{pre} < 0$ and $t_{n+2}$’s $TTC_{pre} < 0$ and $t_{n+3}$’s $TTC_{pre} > 0$ and $t_{n+3}$’s $tp_{crash_{pre}} < t_{n+3}$

The sound should be displayed on time point $t_{n+2}$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} > t_{n+1}$ and $t_{n+2}$’s $TTC_{pre} > 0$ and $t_{n+2}$’s $tp_{crash_{pre}} > t_{n+2}$ and $t_{n+3}$’s $TTC_{pre} > 0$ and $t_{n+3}$’s $tp_{crash_{pre}} < t_{n+3}$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} > t_{n+1}$ and $t_{n+2}$’s $TTC_{pre} < 0$ and $t_{n+3}$’s $TTC_{pre} > 0$ and $t_{n+3}$’s $tp_{crash_{pre}} < t_{n+3}$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} < 0$ and $t_{n+2}$’s $TTC_{pre} < 0$ and $t_{n+3}$’s $TTC_{pre} > 0$ and $t_{n+3}$’s $tp_{crash_{pre}} < t_{n+3}$
- When $t_n$’s $TTC > 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} > t_{n+1}$ and $t_{n+2}$’s $TTC_{pre} > 0$ and $t_{n+2}$’s $tp_{crash_{pre}} > t_{n+2}$ and $t_{n+3}$’s $TTC_{pre} > 0$ and $t_{n+3}$’s $tp_{crash_{pre}} < t_{n+3}$
- When $t_n$’s $TTC < 0$ and $t_{n+1}$’s $TTC_{pre} > 0$ and $t_{n+1}$’s $tp_{crash_{pre}} > t_{n+1}$ and $t_{n+2}$’s $TTC_{pre} < 0$ and $t_{n+3}$’s $TTC_{pre} > 0$ and $t_{n+3}$’s $tp_{crash_{pre}} < t_{n+3}$

<table>
<thead>
<tr>
<th>$TTC$</th>
<th>$&lt;0$</th>
<th>$&gt;0$</th>
<th>$&gt;0$</th>
<th>$&lt;0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TTC_{pre}$</td>
<td>$&gt;0$</td>
<td>$&gt;0$</td>
<td>$&lt;0$</td>
<td>$&gt;0$</td>
</tr>
<tr>
<td>$tp_{crash_{pre}}$</td>
<td>$&gt;t_{n+1}$</td>
<td>$&gt;t_{n+2}$</td>
<td>$&gt;t_{n+3}$</td>
<td>$&gt;t_{n+3}$</td>
</tr>
</tbody>
</table>

Table 7 CDS Functional Requirements
• When $t_n's \ T T C < 0$ and $t_{n+1}'s \ T T C_{pre} < 0$ and $t_{n+2}'s \ T T C_{pre} < 0$ and $t_{n+2}'s \ t_p^{crash}_{pre} > t_{n+2} \ and \ t_{n+3}'s \ T T C_{pre} > 0 \ and \ t_{n+3}'s \ t_p^{crash}_{pre} < t_{n+3}$

• When $t_n's \ T T C < 0$ and $t_{n+1}'s \ T T C_{pre} < 0$ and $t_{n+2}'s \ T T C_{pre} < 0$ and $t_{n+3}'s \ T T C_{pre} > 0 \ and \ t_{n+3}'s \ T T C_{pre} < 0 \ and \ t_{n+3}'s \ t_p^{crash}_{pre} < t_{n+3}$

**A warning sound shall not be provided based on the following different conditions:**

• When $t_n's \ T T C < 0$ and $t_{n+1}'s \ T T C_{pre} < 0$ and $t_{n+2}'s \ T T C_{pre} < 0$ and $t_{n+3}'s \ T T C_{pre} < 0$

• When $t_n's \ T T C > 0$ and $t_{n+1}'s \ T T C_{pre} > 0$ and $t_{n+1}'s \ t_p^{crash}_{pre} > t_{n+1} \ and \ t_{n+2}'s \ T T C_{pre} > 0 \ and \ t_{n+2}'s \ t_p^{crash}_{pre} > t_{n+2} \ and \ t_{n+3}'s \ T T C_{pre} > 0 \ and \ t_{n+3}'s \ t_p^{crash}_{pre} > t_{n+3}$

• When $t_n's \ T T C > 0$ and $t_{n+1}'s \ T T C_{pre} > 0$ and $t_{n+2}'s \ T T C_{pre} > 0 \ and \ t_{n+2}'s \ T T C_{pre} < 0 \ and \ t_{n+3}'s \ T T C_{pre} < 0$

• When $t_n's \ T T C > 0$ and $t_{n+1}'s \ T T C_{pre} > 0$ and $t_{n+1}'s \ T T C_{pre} > t_{n+1} \ and \ t_{n+2}'s \ T T C_{pre} > 0 \ and \ t_{n+2}'s \ T T C_{pre} > t_{n+2} \ and \ t_{n+3}'s \ T T C_{pre} > 0 \ and \ t_{n+3}'s \ T T C_{pre} > t_{n+3}$
### Appendix C --- Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBRS</td>
<td>Seat-Belt Reminder System</td>
</tr>
<tr>
<td>CDS</td>
<td>Collision Detection System</td>
</tr>
<tr>
<td>$V$</td>
<td>Vehicle velocity</td>
</tr>
<tr>
<td>$V_{pre}$</td>
<td>Predicted vehicle velocity</td>
</tr>
<tr>
<td>$V_f$</td>
<td>The front vehicle velocity</td>
</tr>
<tr>
<td>$V_r$</td>
<td>The rear vehicle velocity</td>
</tr>
<tr>
<td>$V_{fl}$</td>
<td>The vehicle front left wheel</td>
</tr>
<tr>
<td>$V_{fr}$</td>
<td>The vehicle front right wheel</td>
</tr>
<tr>
<td>$V_{rl}$</td>
<td>The vehicle rear left wheel</td>
</tr>
<tr>
<td>$V_{rr}$</td>
<td>The vehicle rear right wheel</td>
</tr>
<tr>
<td>$d$</td>
<td>The relative distance between the front vehicle and the rear vehicle</td>
</tr>
<tr>
<td>$TTC$</td>
<td>Time to Collision. It is used to distinguish the driving security status. The vehicle is in a safe situation if the value of $TTC$ is negative, otherwise the opposite.</td>
</tr>
<tr>
<td>$TTC_{pre}$</td>
<td>Predicted $TTC$</td>
</tr>
<tr>
<td>$\Delta TTC$</td>
<td>The difference between $TTC$s</td>
</tr>
<tr>
<td>$\Delta TTC_{pre}$</td>
<td>Predicted delta $TTC$</td>
</tr>
<tr>
<td>$t_{brake}$</td>
<td>The time period for car to stop</td>
</tr>
<tr>
<td>$t_{brake_{pre}}$</td>
<td>Predicted $t_{brake}$</td>
</tr>
<tr>
<td>$t_{braking}$</td>
<td>Time to start brake</td>
</tr>
<tr>
<td>$t_{braking_{pre}}$</td>
<td>Predicted $t_{braking}$</td>
</tr>
<tr>
<td>$t_{crash}$</td>
<td>Time point to crash</td>
</tr>
<tr>
<td>$t_{crash_{pre}}$</td>
<td>Predicted $t_{crash}$</td>
</tr>
<tr>
<td>$t_{warning}$</td>
<td>Time point to warn</td>
</tr>
<tr>
<td>$t_{warning_{pre}}$</td>
<td>Predicted $t_{warning}$</td>
</tr>
<tr>
<td>$t_{reaction}$</td>
<td>Time for driver to react during emergency situation</td>
</tr>
<tr>
<td>$x$</td>
<td>Detected signal of the vehicle wheel speed sensor</td>
</tr>
</tbody>
</table>