A Method and Tool for Automated Analysis of Heavy Vehicle Requirements

Trevor Jagerfield
tjd12001@student.mdh.se

Examiner: Cristina Seceleanu
Mälardalen University, Västerås, Sweden

Supervisor: Predrag Filipovikj
Mälardalen University, Västerås, Sweden
Scania, Södertälje, Sweden

Supervisor: Guillermo Rodriguez-Navas
Mälardalen University, Västerås, Sweden

October 28, 2015
Abstract

The introduction of the functional safety standard ISO 26262 was motivated by an increasing demand to ensure reliability and correctness of safety-critical systems in the automotive industry. However, the adoption of this standard in the automotive industry is hindered by a number of obstacles. Scania is an industrial partner in the VeriSpec project which studies these obstacles and proposes relevant tools and methods compliant with academic and industrial needs. This thesis is within the scope of the VeriSpec project, and aims to address one of the project’s goals, which is to provide tool support for a pattern-based requirement formalization process. The Specification Property System (SPS) proposed by Konrad and Cheng is a patterning method that provides automatic translation of system properties into temporal logics. The SPS also helps in restricting the introduction of ambiguities and inconsistencies in system specification properties. However, the adoption of the SPS in the industry is hindered due to some issues. These issues are, a long learning curve, Constrained Natural Language (CNL) ambiguities, and the lack of tool-support for real-time SPS patterns. In this thesis, a qualitative research study with a literature survey has been performed to find and select state-of-the-art supportive methods to provide feedback on the formalized requirements’ semantics. The Scania Specifier tool has been extended and modified to support a requirement formalization process using the SPS qualitative and real-time patterns. In addition, three supportive methods that resulted from the research study have been integrated into the Specifier tool to provide different feedback options for the users. Finally, the performance of the Specifier tool and the feedback of the supportive methods have been evaluated. The outcome of the study shows that the feedback of the supportive methods helped in guaranteeing the intended behavior of the requirement developers. In addition supportive methods’ feedback enhanced user-friendliness, and aided the users in shortening the SPS learning curve. Finally, an additional outcome of the study is in the form of a number of suggestions and emerged patterns with regard to the SPS usage and supportive methods’ feedback.
Acknowledgments

I am deeply grateful to Predrag Filipovicj my supervisor at MDH and Scania for his valuable insights, and crucial feedback throughout the thesis process. This thesis would not have been possible without his guidance and support. I am also grateful to Dr. Guillermo Rodriguez-Navas for his support and for providing valuable feedback on my thesis.

I would like to thank Dr. Cristina Seceleanu the examiner at MDH for her correction of the thesis, and for approving this work. I would also like to thank my teachers at MDH IDT for the high quality education they provided.

I would like to express my gratitude to Dr. Mattias Nyberg at Scania for providing me with the opportunity to do this thesis, and for his support while I was at Scania. It has been a pleasure working with Engr. Anton Einarson and PhD Student Jonas Westman at Scania, and I am thankful for their guidance and help. I would also like to thank the Scania employees and thesis students who took part in the case study for providing valuable information that helped make this work a success.

Finally, I would like to thank my family and my good friend Sue Lee who supported and encouraged me all the time.
# Table of Contents

Abstract .......................................................................................................................... ii  
Acknowledgements ....................................................................................................... iii  
Table of Contents .......................................................................................................... v  
List of Figures ................................................................................................................ viii  
List of Tables .................................................................................................................. ix  
List of Abbreviations ..................................................................................................... x  

## Chapter 1  
Introduction ................................................................................................................... 1  
  1.1. Background and Thesis Motivation ................................................................. 1  
  1.2. Problem Statement and Research Questions ................................................. 2  
  1.3. Thesis Objectives ............................................................................................ 3  
  1.4. Overview of the Thesis Research Methodology and Implementation .......... 4  
  1.5. Limitations and Delimitations of the Study .................................................... 4  
  1.6. About Scania .................................................................................................... 4  
  1.7. Outline of the Thesis ...................................................................................... 5  

## Chapter 2  
Preliminaries .................................................................................................................. 6  
  2.1. Formal Verification (FV) ................................................................................ 6  
  2.2. Specification Pattern System (SPS) ............................................................... 8  

## Chapter 3  
Study Design and Methodology .................................................................................... 12  
  3.1. Selecting the Study Methodology and its Approach ...................................... 12  
  3.2. Data Collection Procedures .......................................................................... 13  
  3.3. Sample Selection Procedures ........................................................................ 13  
  3.4. Data Analysis Procedures ............................................................................ 14  
  3.5. Validation and Evaluation Procedures ......................................................... 15  
  Summary .................................................................................................................... 15  

## Chapter 4  
Supportive Methods ..................................................................................................... 16  
  4.1. Supportive Methods from the Literature Survey ......................................... 16  
    4.1.1. A Graphical Property Specification Language ........................................ 16  
    4.1.2. Real-Time Graphical Interval Logic ....................................................... 19
4.1.3. Runtime Monitoring Framework for Property Specification Patterns .......... 20
4.1.4. Use Case Maps Property Pattern System .............................................. 21
4.1.5. Template Based Approach for Temporal Logic Visualizing .................... 22
4.1.6. Finite State Automata Property Pattern Templates ................................ 25
4.2. Supportive Methods from the Preliminary Interviews Feedback .................. 26
  4.2.1. Modified Timing Diagram ................................................................. 26
4.3. Supportive Methods Features Comparison Table ....................................... 27
Summary ........................................................................................................... 27

Chapter 5
Case Study Implementation and Findings .............................................................. 29
5.1. Case Study Implementation Procedures ....................................................... 29
  5.1.1. Literature Survey .................................................................................. 29
  5.1.2. Case Study Material Preparation .......................................................... 30
  5.1.3. Preliminary Interviews .......................................................................... 32
  5.1.4. Case Study Interviews .......................................................................... 33
5.2. Case Study Findings .................................................................................... 34
  5.2.1. Preliminary Interviews Results .............................................................. 34
      5.2.1.1. Adding the Modified Timing Diagrams (MTD) Method .................. 35
      5.2.1.2. User-guidance Supportive Method .............................................. 36
  5.2.2. Case Study Interviews Results .............................................................. 36
  5.2.3. Case Study Discussions, Analysis and Conclusions ............................... 39
      5.2.3.1. Pattern I. Role of SPS experience in Supportive Methods Selections .... 39
      5.2.3.2. Pattern II. Difficulties with some SPS Patterns and Temporal Operators... 40
      5.2.3.3. Pattern III. Interviewees Biased Selection Tendencies ...................... 41
      5.2.3.4. Interviewees Suggestions and Study Observations .......................... 41
  5.2.4. Case Study Account ............................................................................. 42
Summary ............................................................................................................. 42

Chapter 6
The Scania Specifier Tool .................................................................................. 43
6.1. Specifier Modifications and Extensions ..................................................... 43
  6.1.1. Pattern Information and Views Window .............................................. 43
  6.1.2. The Edit/Add Requirement Window .................................................... 44
6.2. Evaluation of the Specifier Tool SPS Support ........................................... 47
  6.2.1. The Evaluation Events ........................................................................... 47
  6.2.2. Final Evaluation Results and Discussions .......................................... 48
Summary ............................................................................................................. 49
Chapter 7
Related Work.................................................................................. 50
7.1. Related SPS Supportive Tools ..................................................... 50
  7.1.1. Property Elucidation (PROPEL) ........................................... 50
  7.1.2. Property ASSistant (PASS) .................................................. 50
  7.1.3. Property Specification (Prospec) .......................................... 50
  7.1.4. CHARMY ........................................................................... 51
7.2. SPS Supportive Tools Comparison ........................................... 51
Summary.......................................................................................... 53

Chapter 8
Conclusions and Future Work.......................................................... 55
  8.1. Conclusions .............................................................................. 55
  8.2. Future Work .............................................................................. 57

References........................................................................................ 58
Appendices ....................................................................................... 62
  A. Modified LR SPS Pattern Semantics Representations..................... 63
  B. Template Based Diagram SPS Pattern Semantics Representations .......... 67
  C. Modified Timing Diagram SPS Pattern Semantics Representations .......... 72
  D. Case Study Questionnaire............................................................. 75
      D.1. Introduction ........................................................................... 75
      D.2. Assignments ................................................................. 75
      D.2.1. Formalizing Requirements ........................................... 75
      D.2.2. Select the Pattern that Conveys the Correct Requirement Behavior ....... 76
      D.3. Questions ........................................................................... 76
  E. Demo-Day Scenario...................................................................... 77
  F. Scania Requirements used for Evaluating the Specifier .................... 78
List of Figures

Figure 2.1: Verification Overview [26]................................................................. 6
Figure 2.2: Model Checking Method [28] .......................................................... 7
Figure 2.3: Qualitative Property Patterns Hierarchy ........................................... 8
Figure 2.4: Property System Patterns Scopes [8]. The diagram is further enhanced with text to illustrate where patterns can hold or not hold. .......................... 9
Figure 2.5: SPS Classification [12] ..................................................................... 10
Figure 2.6: Requirement Patterning Comparison Table [17] .............................. 10
Figure 3.1: Data Collection Cycle by Creswell [33] .......................................... 13
Figure 3.2: Data Analysis Spiral by Creswell [33] ............................................. 14
Figure 4.1: Simple LR Real-Time Cause-Effect Relation [46] .......................... 17
Figure 4.2: MLR Representations of the Response Pattern. This Figure shows a Textual Approach to Address CNL Ambiguities ........................................ 18
Figure 4.3: RTGIL Representations of a Response Pattern [50] ....................... 20
Figure 4.4: Response Pattern Semantics Represented in USD [52] ................. 21
Figure 4.5: Response Pattern Semantics Represented in UCMPPS [56] .......... 22
Figure 4.6: Tracing the Response Pattern through the TBATLV Template Flow-Diagram ................................................................. 24
Figure 4.7: FSAPPT Representation of the Response Pattern ......................... 26
Figure 4.8: Example of an MTD Representations of a Response Pattern ......... 27
Figure 5.1: Case Study Design Diagram ............................................................ 31
Figure 5.2: Preliminary Interviews Structure. The Interviews used the Interview Protocol ...... 32
Figure 5.3: Case Study Interviews Structure. The Interviews used the Interview Protocol ..... 34
Figure 5.4: Simple Example Explaining the Response Pattern ..................... 35
Figure 5.5: Simple Example Explaining the Precedence Pattern ................... 36
Figure 5.6: Graphical Representation of Interviewees Final Supportive Methods’ Selections... 39
Figure 5.7: UCMPPS Representation of the Precedence Pattern .................... 41
Figure 6.1: Pattern Information and Views Window. Adding a Pattern to the Pattern List ....... 43
Figure 6.2: Pattern Information and Views Window. Adding Views to a New Pattern .......... 44
Figure 6.3: Requirement Patterning using the Globally Scope and the Response Pattern with Empty Proposition Fields .................................................... 45
Figure 6.4: Requirement Patterning using the “Between Q and R” Scope and the Real-time Bounded Response Pattern .............................................. 46
Figure 6.5: Pattern User-Guidance ..................................................................... 46
Figure 6.6: Scope User-Guidance ...................................................................... 47
Figure A.1: MLR Representations of Different Patterns with Response Semantics. These Figures show a Textual Approach to Address CNL Ambiguities .......................... 63
Figure A.2: MLR Representations of Different SPS Patterns with Precedence Semantics. These Figures show a Textual Approach to Address CNL Ambiguities ......................... 64
Figure A.3: Occurrence Pattern Semantics Representation in MLR .................. 65
Figure A.4: Real-Time Pattern Semantics Representation in MLR .................. 66
Figure B.1: TBATLV Diagram of the Main Branch in the Flow-Diagram .......... 67
Figure B.2: TBATLV Diagram of the Real-Time Patterns Branch .................. 68
Figure B.3: TBATLV Diagram of the Qualitative Patterns Branch ................. 69
Figure B.4: TBATLV Diagram of the Order Pattern Branch ......................... 70
Figure B.5: TBATLV Diagram of the Pattern Scopes ....................................... 71
Figure C.1: MTD Representation of the Precedence Pattern - The Cause S Precedes and Stimulates the Effects V, T and P. This Figure shows A Graphical Approach to Address CNL Ambiguities .................................................. 72
Figure C.2: MTD Representation of the Response Pattern with “After Q” Scope - The Pattern Always Holds After the First Occurrence of Q but Not Before it. This Figure shows A Graphical Approach to Address CNL Ambiguities ........................................ 73
Figure C.3: MTD Representation of the Response Pattern with the “Before R” Scope - The Pattern Always Holds Before the First Occurrence of R but Never After it. This Figure shows A Graphical Approach to Address CNL Ambiguities. ............................................................... 74

List of Tables

Table 2.1: Dwyer et al. Patterns’ Categories and Patterns’ Descriptions [8]................................. 9
Table 4.1: LR Graphical Notation.................................................................................................. 17
Table 4.2: MLR Notation Diagram Symbols................................................................................... 18
Table 4.3: RTGIL Method Diagram Symbols.................................................................................. 19
Table 4.4: UCMPPS Diagram Symbols............................................................................................ 22
Table 4.5: FSA Property Pattern Templates Graphical Notation.................................................... 25
Table 4.6: Supportive Methods Feature Comparison Table.......................................................... 28
Table 5.1: Preliminary Interviews Participants List......................................................................... 32
Table 5.2: Case Study Interviews Participants List.......................................................................... 33
Table 5.3: Preliminary Interviews Questionnaire Results............................................................... 34
Table 5.4: Interviewee Final Supportive Methods’ Selections......................................................... 39
Table 6.1: Specifier Tool Evaluation Results.................................................................................. 48
Table 7.1: SPS Tool Comparison Table.......................................................................................... 53
Table F.1: Scania Requirements used for Specifier Tool Evaluation. These Requirement are taken from Westman and Nyberg [65]................................................................. 78
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E/E</td>
<td>Electrical and Electronic</td>
</tr>
<tr>
<td>FV</td>
<td>Formal Verification</td>
</tr>
<tr>
<td>VeriSpec</td>
<td>Structured Specification and Automated Verification for Automotive Functional Safety</td>
</tr>
<tr>
<td>SPS</td>
<td>Specification Pattern System</td>
</tr>
<tr>
<td>CNL</td>
<td>Constrained Natural Language</td>
</tr>
<tr>
<td>PROPEL</td>
<td>Property Elucidation</td>
</tr>
<tr>
<td>PASS</td>
<td>Property ASSistant</td>
</tr>
<tr>
<td>Prospec</td>
<td>Property Specification</td>
</tr>
<tr>
<td>RESA</td>
<td>Vehicle electrical Architecture and Chassis System Software</td>
</tr>
<tr>
<td>LTL</td>
<td>Linear Temporal Logic</td>
</tr>
<tr>
<td>CTL</td>
<td>Computational Tree Language</td>
</tr>
<tr>
<td>TCTL</td>
<td>Timed Computational Tree Language</td>
</tr>
<tr>
<td>LR</td>
<td>Graphical Property Specification Language</td>
</tr>
<tr>
<td>MLR</td>
<td>Modified Graphical Property Specification Language</td>
</tr>
<tr>
<td>RTGIL</td>
<td>Real-time Graphical Interface Logic</td>
</tr>
<tr>
<td>GIL</td>
<td>Graphical Interface Logic</td>
</tr>
<tr>
<td>FIL</td>
<td>Future Interval Logic</td>
</tr>
<tr>
<td>RTFIL</td>
<td>Real-time Future Interval Logic</td>
</tr>
<tr>
<td>RMF</td>
<td>Runtime Monitoring Framework</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>USD</td>
<td>UML 2.0 Sequence Diagram</td>
</tr>
<tr>
<td>FSA</td>
<td>Finite State Automata</td>
</tr>
<tr>
<td>UCMPPS</td>
<td>Use case maps property pattern system</td>
</tr>
<tr>
<td>ArTCTL</td>
<td>Architectural Timed Computational Tree Language</td>
</tr>
<tr>
<td>TBATLV</td>
<td>Template Based Approach for Temporal Logic Visualization</td>
</tr>
<tr>
<td>FSAPPT</td>
<td>Finite State Automata Property Pattern Templates</td>
</tr>
<tr>
<td>MTD</td>
<td>Modified Timing Diagrams</td>
</tr>
<tr>
<td>TD</td>
<td>Timing Diagrams</td>
</tr>
<tr>
<td>KTH</td>
<td>Kungliga Tekniska Högskolan</td>
</tr>
<tr>
<td>DNL</td>
<td>Disciplined Natural Language</td>
</tr>
<tr>
<td>QT</td>
<td>Question Tree</td>
</tr>
<tr>
<td>CP</td>
<td>Composite Propositions</td>
</tr>
<tr>
<td>Promela</td>
<td>Protocol Meta Language</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>SPIN</td>
<td>Simple Promela Interpreter</td>
</tr>
<tr>
<td>PSC</td>
<td>Property Sequence Charts</td>
</tr>
</tbody>
</table>
1. Introduction

The automotive industry is witnessing an increasing dependency on software systems. Software is becoming an integral part of the Electrical and Electronic (E/E) systems leading to escalating systems’ complexity and failure probability. At the same time, the software systems developed in the automotive industry are increasingly being labeled as safety-critical systems, whose failure can cause loss of life. As a result, there is a growing demand in automotive industry to ensure the reliability and correctness of safety-critical software systems [1, 2].

1.1. Background and Thesis Motivation

The road vehicles' functional safety standard ISO 26262 [3] is a standard introduced to address concerns regarding safety-critical software systems in the automotive industry. The standard promotes applying Formal Verification (FV) techniques at every level of system abstraction in an aim to reduce the functional safety concerns to a tolerable level [4, 5]. FV exhaustively and automatically checks system correctness to ensure the exclusion of undetected failures in the system’s behavior. FV is therefore gaining an increased attention because it addresses system property ambiguities, ensures quality, and minimizes failure probabilities [5-7]. However, adopting the ISO 26262 is being hindered in the automotive industry due to a number of issues. These issues are, such as, lack of expertise in formal notation, lack of tool support and system specification ambiguities [5, 8].

Scania is a leading manufacturer of heavy-vehicles, and it allocates large resources to ensure the quality and safety of its products. Scania along with Volvo, and Mälardalen University started the Structured Specification and Automated Verification for Automotive Functional Safety (VeriSpec) project. The VeriSpec project is funded by VINNOVA, which is a Swedish government agency that manages programs aiming to strengthen Sweden’s innovativeness [9]. The VeriSpec project aims to facilitate the adoption of the ISO 26262 standard in the automotive industry by introducing FV at different levels of the development process. The VeriSpec project’s main research goals address two aspects of the ISO 26262 standard. The first goal is to define a pattern-based method for specifying safety relevant functional requirements, and providing tool support for formalizing qualitative and real-time requirements. The second goal is to provide an automated FV using model checking. The VeriSpec project studies the challenges hindering the adoption of the ISO 26262 standard and proposes solutions complaint with industrial and academic needs [10]. To formally specify system requirements by manually writing formal notations that describe requirements’ behaviors is a process which requires advanced expertise in formal notations. However, such expertise are hard to find in the industry, and are expensive to learn in terms of money and time. In addition, it is difficult to find good training material for formal specification, which is a very precise and an error-prone process [8]. One method to overcome these issues is a pattern-based approach proposed by Dwyer et al. [8] called the Specification Pattern System (SPS). The Dwyer et al. SPS consists of a set of high-level, formalism-independent abstract property patterns representing commonly recurring qualitative system property behaviors. It facilitates the formal specification process by addressing the issue of lack of expertise in formal languages in the industry. However, the Dwyer et al. SPS does not support real-time system properties and lacks a textual interface to enhance user friendliness [8, 11].

Konrad and Cheng [12] extended the Dwyer et al. SPS by the addition of real-time patterns and the Constrained Natural Language (CNL). The extended SPS\(^\dagger\) can therefore express system properties' behaviors in both formal notations and CNL. CNL is an informal textual interface

\(^\dagger\) The Konrad and Cheng SPS is also referred to as the “SPS”, or the “extended SPS” in this thesis.
mapped to relevant SPS qualitative and real-time patterns, and it facilitates the use of the SPS by hiding patterns’ formalism [13]. However, Filipovikj et al. [5] identified a number of issues that are hindering the adoption of the SPS in the industry, such as, a long learning curve, CNL ambiguities, and lack of tool support for the SPS real-time patterns. Using the SPS requires a learning curve to get efficient in using the correct SPS patterns and scopes [5, 10]. CNL ambiguities cause difficulties in understanding the correct behaviors of the requirements formalized using the SPS. As a result, requirement developers have to refer to the formal notations of the SPS patterns in order to understand their correct temporal behaviors. Lack of tool support was identified by both Filipovikj et al. and Rodriguez-Navas et al. [10] who emphasized the need for tool support to facilitate a Konrad and Cheng SPS based requirement formalization process.

1.2. Problem Statement and Research Questions

The scope of this thesis is within one of the VeriSpec project’s main contributions, which targets the automated generation of pattern-based formal specification [10]. A number of tools such as the Property Elucidation (PROPEL) [11], Property ASSistant (PASS) [14], Property Specification (Prospek) [15], and CHARMY [16], have already been proposed by researchers to assist engineers in the process of formal specification. However, none of these tools supports the specification of real-time system properties. A pattern-based method which supports both qualitative and real-time patterns is the SPS proposed by Konrad and Cheng. The expressiveness of the SPS was tested on functional requirements in the automotive industry in at least two case studies with satisfactory results [5, 17]. Filipovikj et al. stated that using the SPS helps to reduce the formalized requirements’ ambiguity, improves its testability and promotes communication between stakeholders. However, Filipovikj et al. identified a number of issues hindering the acceptance of the SPS in the industry, such as, a long learning curve, CNL ambiguities, and lack of tool support for the SPS. As a solution for the first two issues, Filipovikj et al. proposed providing supportive material as feedback on the formalized requirements’ semantics. With regard to the third issue, both Filipovikj et al. and Rodriguez-Navas et al. stressed the need for a user-friendly SPS-based tool to automate the requirement formalization process [5, 10].

The Specifier is a Scania specific tool developed as part of the ESPRESSO project which is a collaboration between the Kungliga Tekniska Högskolan (KTH) and Scania. ESPRESSO aims to improve the process of developing embedded systems by enhancing the development process quality, and reducing its costs [18, 19]. The goal of the Specifier tool is to provide a model-based solution for improving the efficiency of document management. The Specifier tool implements the contract structure proposed by Westman and Nyberg [20], and allows a hierarchical and traceable requirements' structure compliant with the ISO 26262. Hence, the Specifier tool is capable of supporting Safety Goals, Functional Safety Requirements, Technical Safety Requirements, and Hardware and software requirements [21, 22].

This thesis problem statement is based on the issues identified by both Filipovikj et al. and Rodriguez-Navas et al., which were mentioned earlier in this section. Addressing these issues requires first finding state-of-the-art supportive methods that provide feedback on the formalized requirements’ semantics. The next step is to evaluate users’ (engineers) preferences with regard to supportive methods' performance in representing formalized requirements’ semantics. Furthermore, the Specifier tool has to be extended to provide support for qualitative and real-time SPS formal specification. In addition, a number of supportive methods preferred by the engineers have to be integrated into the Specifier tool. Finally, the performance and user-friendliness of the modified Specifier tool have to be evaluated with regard to supporting a qualitative and real-time SPS requirement formalization process. The evaluation should also assess the effectiveness of the supportive methods’ feedback in helping to guarantee that the formalized requirements’ semantics match the intended behavior of requirement developers.
The main research question of this thesis is:

What is required to provide tool support for an SPS requirement formalization process aided with state-of-the-art supportive methods’ feedback on the formalized requirements’ semantics?

The main research question of this thesis is further divided into five sub-questions (RQ1-RQ5):

**RQ1:** Based on the current state-of-the-art, which existing methods can provide relevant feedback about formalized requirements’ semantics to the requirement developers?

**RQ2:** What state-of-the-art supportive methods are preferred by the engineers based upon the feedback provided by the supportive methods on the formalized requirements’ semantics?

**RQ3:** What changes should be introduced into the Specifier tool to support a user-friendly qualitative and real-time SPS requirement formalization process?

**RQ4:** What is the best way to provide requirement developers with supportive methods’ feedback using the Specifier tool?

**RQ5:** In which way should the modified Specifier tool be evaluated regarding SPS requirement formalization process, user-friendliness, and the effectiveness of the supportive method’s feedback?

### 1.3. Thesis Objectives

The main objective of this thesis is to provide tool support for an SPS requirement formalization process, and to provide supportive methods’ feedback on the formalized requirements’ semantics. This objective is further divided into three sub-objectives that address the thesis problem statement sub-questions (RQ1-RQ5). The first sub-objective addresses both RQ1 and RQ2, and is concerned with finding supportive methods that can provide feedback on the formalized requirements’ semantics and selecting the supportive methods that can provide the best feedback. RQ1 is addressed by conducting a literature survey to find state-of-the-art supportive methods for providing the feedback. RQ2 is addressed by conducting a study to evaluate the feedback of the supportive methods with regard to help in guaranteeing that the formalized requirements’ semantics match the intention of the user.

The second sub-objective is to provide tool support for a requirement formalization process using the SPS qualitative and real-time patterns, and to integrate supportive methods into the Specifier tool. This sub-objective addresses both RQ3 and RQ4 of the problem statement of this thesis. RQ3 is addressed by extending and modifying the Scania Specifier tool so as to support a requirement formalization process based on the SPS qualitative and real-time patterns. RQ4 is addressed by integrating one or more supportive methods selected from the research study into the Specifier tool.

Finally, the third sub-objective addresses RQ5. The third sub-objective aims to evaluate the performance of the tool support and the feedback provided to the requirement developers in facilitating the requirement formalization process. The evaluation will be achieved by asking a number of participants for their feedback with regard to formalizing a number of Scania requirements using the Specifier tool.
1.4. Overview of the Thesis Research Methodology and Implementation

An academic research approach has been followed, and practical implementation has been performed to achieve the objective of this thesis. The academic research approach consisted of a qualitative research study with a literature survey. The literature survey was conducted to find state-of-the-art supportive methods, and criteria were applied to filter the results. The resulting supportive methods were then prepared, and a qualitative research methodology was defined. The research methodology was implemented as a case study approach to explore opinions, collect observations and feedback from participants in the study.

Practical implementation involved extending and modifying the Scania Specifier tool to support a requirement formalization process using the SPS qualitative and real-time patterns. The implementation also included integrating a number of supportive methods that resulted from the case study. Finally, the modified Specifier tool was evaluated to assess the performance of the tool support, and the feedback provided to the requirement developers.

1.5. Limitations and Delimitations of the Study

Study limitations describe the conditions that cannot be controlled by the author and therefore, restrict the scope of the study and affect its outcome [23]. The main limitation identified in this study was the limited time available to introduce the SPS and the supportive methods to the Scania interviewees. Consequently, the interviews had to be planned so as to both provide a brief introduction segment and collect the data in short interview sessions. As a result of this limitation, Scania engineers had only basic knowledge and limited exposure to the SPS and supportive methods, which influenced their supportive methods’ selections. On the other hand, this limitation provided a valuable opportunity to collect data regarding their experiences in using the SPS.

Study delimitations are the boundaries set by the author to define a manageable scope for the study and constrain its result generalizability [23]. Delimitation actions were taken regarding the supportive methods that resulted from the literature survey, which did not comply with the defined filtering criteria. None of the state-of-the-art supportive methods resulting from the literature survey addressed the CNL ambiguity issues. In addition, some of the state-of-the-art supportive methods did not provide some or any of the SPS pattern representations. As a result, three delimitation actions had to be taken. In the first action, exemptions from the literature survey criteria were made in order to include these supportive methods into the study. Second, three supportive methods that would have been otherwise excluded from the study, were modified and the modifications included addressing the CNL ambiguities. The purpose of including these three supportive methods was to illustrate the CNL ambiguities to the study participants, and to get their reaction towards these attempts. The third action was to make SPS pattern representations for some of the supportive methods. These delimitation actions resulted in including more supportive methods into the study, however, implementing these actions was both time and effort consuming.

1.6. About Scania

This master thesis was performed at the Vehicle electrical Architecture and Chassis System Software (RESA) department at the Scania Technical Centre in Södertälje, Sweden. RESA is a group within Scania responsible for the electrical/electronic systems and their architecture in trucks. Scania is a leading heavy-vehicle manufacturer founded in 1891 in Sweden with the objective of providing optimized products such as, trucks, buses and heavy engines to the customers. Scania is an international organization operating in more than 100 countries around the globe. There are over 35,500 employees working for Scania in three main departments, sales, production and research [24].

---

[ về văn bản ]

---
1.7. Outline of the Thesis

The rest of this thesis is organized as follows:
Chapter two presents information with regard to the preliminary techniques which are important for the thesis. Chapter three defines and discusses the study methodology applied, and the procedures followed in this thesis. Chapter three presents details about the research methodology and its inquiry approach, data collection, sample selection, data analysis, and the research validation and evaluation procedures. Chapter four provides a brief introduction to seven supportive methods and explains their graphical notations with examples. Chapter five presents the case study implementation and findings. The implementation section presents the literature survey, case study material preparation, preliminary interviews, and interviews. The findings section presents the interviews’ results followed by the discussions and analysis on the findings. Chapter six briefly presents a number of existing related tools that support the Dwyer et al. SPS requirement formalization process. In addition, this chapter also provides a presentation on the implementations introduced on the Scania Specifier tool to support an SPS requirement formalization process aided with supportive methods’ feedback. The final section in Chapter six presents the advantages of the Scania Specifier tool comparing it with other related tools mentioned in this thesis. Chapter seven presents the conclusions and future work of the thesis. Finally, appendices A, B, and C contain SPS representations which the author made for three supportive methods. Appendices D, E, and F contain the case study questionnaire, demo-day scenario, and the Scania requirements used for evaluating the Specifier tool, respectively.
2. Preliminaries

This chapter briefly presents background information on the technologies necessary for understanding this thesis work. The presented technologies are FV, and the SPS proposed by Dwyer et al. and then further extended by Konrad and Cheng.

2.1. Formal Verification (FV)

During the system development process, almost 75% of the total effort is allocated for the verification and debugging of the development process. The system verification process is expensive, and it even becomes more resource-exhaustive as the complexity of the system increases. Figure 2.1 shows an overview of functional verification, which consists of the following techniques: simulation, FV, and hardware emulation. Simulation is a simple and a commonly practiced technique for system verification in which output values are monitored for a given input data sequence. The simulation technique works fine for the simpler systems; however, it is not preferred for the more complex systems due to a number of drawbacks. Among these drawbacks are time and resource consumption, late application in the development process, and vulnerability to human error since it is usually performed manually [25, 26].

Another technique in system verification is FV, which identifies system issues related to property ambiguities, and inconsistencies [27]. FV has a number of important features, such as its high automation and exhaustive system state exploration. In addition, another significant feature of FV is that it can be applied at the early stages of the development process [2, 6].

![Figure 2.1: Verification Overview [26].](image)

FV uses mathematical proof to prove or disprove the system’s functional correctness [2, 6]. The process checks the conformance of the system’s formal model against its formally specified properties [28]. There are a number of FV techniques, and they are grouped into three main categories [26]:

- Arithmetic verification: It proves or disproves that an implementation is fulfilling its design purpose operation-wise.
• Property checking: It proves or disproves that a design satisfies the intended behavior. This category can be further divided into several models, each providing its own formalism method and FV procedure. These models are: theorem proving, model checking, language containment, and symbolic trajectory evaluation.

• Equivalence checking: It proves or disproves the functional equivalence of two given system designs.

A prerequisite for formalizing a system’s properties is ensuring the correctness of the system specification. In fact, the lack of an accurate system specification is one of the obstacles hindering the adoption of methods such as the SPS in the industry [5]. Lamsweerde [29] defined criteria for a correct system specification. According to these criteria, a specification has to be adequate, unambiguous, complete, internally consistent, and finally, correctly satisfied by its lower-level properties. An adequate and unambiguous specification, is a specification that correctly represents the system properties' behavior, and has no more than one interpretation. A complete specification correctly describes both the properties’ decomposition to their lower-level properties, and the properties’ composition of their higher-level sources. Finally, an internally consistent system specification means that the semantic behavior of all system properties will remain correct when combined together [29].

The FV technique of interest in the VeriSpec project which this thesis is part of is the model checking method [10]. This method uses a tool called the Verifier to perform model checking. The Verifier is based on a state-space search method which explores each system state and checks, whether it satisfies a desired property behavior or not. The model checker shown in Figure 2.2 takes a system’s formal model and its formal specification as inputs. The model checker afterwards determines all the potential reachable system states, including deadlock states where no further progress is possible. If the model conforms to the specification, then the Verifier returns
“true” otherwise, it returns “counter example”. The Counter Example provides the details of the mismatch, which are then analyzed. Based on the analysis, the causing errors can be found and corrected. The automated model checker makes it easy to re-check the modified properties against the system’s formal model [25, 28, 30].

2.2. Specification Pattern System (SPS)

Requirements written in natural language might not capture all the necessary information required for describing the correct behavior of the system [5]. Nevertheless, requirement developers prefer using natural language to specify requirements over formalizing them with formal notations, which promote unambiguity. The reason is that natural language is easier to use in comparison to the more rigorous temporal logic used in formal specification [31]. The lack of expertise in using formal notations was one of the obstacles identified by Dwyer et al. that were hindering the use of FV in the automotive industry. Another obstacle identified by Dwyer et al. was the lack of supportive tools [8, 31].

Dwyer et al. studied a number of functional qualitative requirements written in natural language, and noticed reoccurring behaviors, and categorized them into patterns. Figure 2.3 shows the hierarchy of the qualitative patterns proposed by Dwyer et al. The patterns are classified according to their semantics, and are grouped into three main categories Occurrence, Order, and Compound. Patterns in the Occurrence category describe properties that are dependent on the occurrence of certain states or events within a defined scope. Order category contains patterns that describe a cause-effect relation between a pair of states or events within a defined scope. Compound category contains Chain Precedence and Chain Response patterns within a defined scope. Another type of Compound patterns is the Boolean pattern which combines patterns using Boolean operators. Table 2.1 presents the patterns different categories with patterns’ descriptions.

![Figure 2.3: Qualitative Property Patterns Hierarchy](image)

The Dwyer et al. SPS patterns are high level, formalism-independent, and abstract property patterns, which can be mapped to different formal notations. Based upon these patterns, Dwyer et al. proposed the SPS where, property patterns mapped to formal notations are used to represent a requirement formally in terms of behavior and scope. Each SPS pattern consists of literal and non-literal terminals. The literal terminal is surrounded by quotation marks, describes the pattern behavior, and is not changeable, while the non-literal terminals represent Boolean propositions or time constraints. The Dwyer et al. SPS scopes are shown in Figure 2.4, and they represent the extension of the program execution where a behavior is expected to hold. Using Dwyer et al. SPS, developers can formalize requirements without the need to write formal notations. However, in order to use the Dwyer et al. SPS efficiently, it is necessary that developers receive sufficient training [8].

---

8
Table 2.1: Dwyer et al. Patterns’ Categories and Patterns’ Descriptions [8].

<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrence</td>
<td>Absence</td>
<td>A given state/event does not occur within a scope.</td>
</tr>
<tr>
<td></td>
<td>Universality</td>
<td>A given state/event occurs throughout a scope.</td>
</tr>
<tr>
<td></td>
<td>Existence</td>
<td>A given state/event must occur within a scope.</td>
</tr>
<tr>
<td></td>
<td>Bounded-Existence</td>
<td>A given state/event must occur k times within a scope.</td>
</tr>
<tr>
<td>Order</td>
<td>Response</td>
<td>A state/event P must always be followed by a state/event Q within a scope.</td>
</tr>
<tr>
<td></td>
<td>Precedence</td>
<td>A state/event P must always be preceded by a state/event Q within a scope.</td>
</tr>
<tr>
<td>Compound</td>
<td>Chain Response</td>
<td>A sequence of states/events P₁,..., Pₙ must always be followed by a sequence of states/events Q₁,..., Qₘ.</td>
</tr>
<tr>
<td></td>
<td>Chain Precedence</td>
<td>A sequence of states/events P₁,..., Pₙ, must always be preceded by a sequence of states/events Q₁,..., Qₘ.</td>
</tr>
<tr>
<td></td>
<td>Boolean</td>
<td>Combining patterns using Boolean operators.</td>
</tr>
</tbody>
</table>

Figure 2.4: Property System Patterns Scopes [8]. The diagram is further enhanced with text to illustrate where patterns can hold or not hold.
Konrad and Cheng extended the work of Dwyer et al. with the introduction of real-time patterns and the informal textual interface CNL which hides the patterns’ formalism. Konrad and Cheng SPS supports both qualitative and real-time patterns, and can represent system properties' behaviors on two levels. The first level uses formal notations to formally represent system properties' behaviors, while the second level is textual using where CNL is mapped to the formal notation of each pattern [12]. Figure 2.5 shows the Konrad and Cheng SPS classification which consist of two main categories of patterns, qualitative and real-time patterns. An example of an SPS real-time pattern is the “Bounded Response” pattern, and its CNL states, “it is always the case that if P holds, then S holds after at most c time units”. P, S, and c are non-literal terminals, where P is the cause, S is the effect and c is a time constraint [32].

![Figure 2.5: SPS Classification](image)

The expressibility of the extended SPS in the automotive industry has been tested in at least two case studies with satisfactory results. The first case study was performed by Post et al., who further extended the Konrad and Cheng SPS, and compared performance of different versions of the SPS on 289 functional automotive requirements at BOSCH. These SPS versions were Dwyer et al., Konrad and Cheng, and Post et al. SPS. The case study results are displayed in Figure 2.6, and the inexpressible requirements are shown in purple and red. None of the three SPS versions succeeded in patterning 100% of the requirements, but extending the Dwyer et al. SPS effectively decreased the number of inexpressible requirements [17].

![Figure 2.6: Requirement Patterning Comparison Table](image)
The second case study was performed by Filipovikj et al. at Scania. The case study assessed the applicability of the Konrad and Cheng SPS on E/E systems’ functional requirements written in natural language. The results of the case study showed that 70% of all requirements and 92% of the behavioral requirements were formalized using Konrad and Cheng SPS. Another outcome of the case study showed that the SPS was effective in reducing requirement ambiguity and enhancing communication between stakeholders [5].
3. Study Design and Methodology

This chapter describes the methodology used, and the procedures followed in conducting the research study. The chapter is organized into a number of sections; the first section discusses selecting the research methodology and its inquiry approach. The second section is the data collection procedures, which briefly describes the different cycles of the data collection process. Afterwards, the third section presents the sample selection procedures and details about the participants who took part in the study. Finally, the data analysis, and the study validation and evaluation sections present the processes, and techniques followed to analyze the data, and validate and evaluate the study.

3.1. Selecting the Study Methodology and its Approach

The first sub-objective of the thesis is concerned with finding state-of-the-art supportive methods that can provide feedback on the formalized requirements’ semantics. This is achieved by performing a literature survey to search academic databases for relevant literature. In addition, the first sub-objective also aims to identify the supportive methods that provide the best feedback on the formalized requirements’ semantics. This goal is achieved by performing a qualitative research study which is useful in investigating the participants’ preferences regarding the supportive methods’ feedback. An advantage of the qualitative research is that it involves exploring the human behavior and feedback providing a complex and a detailed understanding of the issue being studied. This level of detail can only be gained by directly communicating with the engineers involved in the study. Another key characteristic of the qualitative research study is its emergent design. Emergent design means that the research study cannot be tightly defined since different elements of the process may change as the process is stated and data is collected. Examples of these elements are case study questions, data collection forms and participants. The purpose of the qualitative research is to understand the problem from the interviewees and to adjust the study in order to obtain the required information [33]. The researcher is a key instrument in the qualitative research. It is the researcher’s role to find state-of-the-art material and create the case study questionnaires. Afterwards, the researcher conducts in-person interviews while taking the role of an observer, collecting data and feedback. Finally, the researcher analyzes the collected data and reports its findings [33, 34].

There are different approaches within the qualitative research to perform a study, and the approach selected for this qualitative research is the case study approach. The case study approach focuses on gathering empirical data, and on performing an in-depth and detailed exploration of the problem within a specific context [33]. The case study category is exploratory because it is exploring the opinions of one group of engineers under the same context. In addition, the case study type is defined as single [35], because it investigates a single case concerning the participants’ preferences regarding supportive methods’ feedback. Furthermore, the case study is defined as holistic because it uses perspectives from different sources such as interview questionnaire, observations and feedback. These multiple perspectives help in building a detailed understanding of the study problem. Finally, the case study is classified as instrumental [36] because the data gathered from exploring the case helps in understanding and improving the experience of using the SPS. The ideal setup for the case study is to conduct it in natural settings. Natural settings include having in-person interactions with the participants and collecting the data in the field where the participants are experiencing the problem [33]. Natural settings also mean that the case environment is not manipulated, while taking into account real-world complexity [37, 38]. At the same time, the interviewer should play the role of an observer, and have as little input as possible during the case study [38].
3.2. Data Collection Procedures

Creswell [33] proposed a data collection circle that consists of a number of interrelated activities with the purpose of collecting detailed information regarding the research question. Figure 3.1 shows the activities involved in the data collection process. The first activity is locating a resource which could be a person, a website, a company or a book. The second activity is to gain access to the resource, and if individuals are involved in providing the data, then, the researcher should establish cooperation and rapport with them. The third activity involves using a sampling technique to ensure that the selected resources can provide the best information related to the research problem. The fourth and fifth activities involve collecting the data from the selected resources and documenting it respectively. The sixth activity deals with field issues related to the data collected from the resources. Resolving these issues contributes to evolve the case study design. Finally, in the seventh activity, the collected data is stored.

![Figure 3.1: Data Collection Cycle by Creswell [33].](image)

The case study preliminary interviews and interviews were in-person, open-ended, followed an interview protocol and took place at Scania where this thesis was performed. All the interviews of the case study were organized to accommodate the interviewees’ schedules and took place at the locations they suggested.

Data collection was performed during both the preliminary interviews and the interviews. Data collection followed specific strategies with regard to conducting observations, collecting feedback and documentation. The observations were taken from the point of view of an observer as hand written filed notes. The notes collected from both the preliminary interviews, and the interviews were then stored in the form of Microsoft Word files, and backup copies were made. One of the features of a qualitative interview is that it is an extended form of conversation. Consequently, the interviewees were looked upon more as partners rather than subjects of test during these qualitative interviews [39]. This partnership setup was convenient, and welcomed during the interviews. As a result, it helped in uncovering important issues with regard to the use of SPS. The feedback from the interviewees with regard to these issues and on the questionnaire was recorded as written notes.

3.3. Sample Selection Procedures

The selected site for the case study was at RESA department in Scania Research Center where this thesis was performed. There were only three participants in the preliminary interviews, two
of them were fellow engineering students who also worked with the SPS, while the third was a Scania project supervisor. The author contacted a RESA manager and requested candidate participants for the case study interviews. RESA management provided a list of seven engineers having different roles in the product-development process from different departments at Scania. The sample provided by Scania fulfilled the criteria for maximum variation sampling, which aims to sample a diversity of individuals [33], and all the seven engineers took part in the case study. All the participants in the interviews were experienced engineers; however, none of them had any previous experience with the SPS, or any of the supportive methods. Three of the participants worked as embedded software developers, one was a tester; another had a managerial and a research position, and the last participant worked as a product owner. All the participants were involved in working with real-time software systems. The reason for limiting the case study to a Scania site and employees was that the thesis was conducted at that site. Furthermore, there was not enough time to visit other companies involved in the VeriSpec project and meet their employees.

3.4. Data Analysis Procedures

![Data Analysis Spiral by Creswell](image)

Creswell proposed an analytical spiral approach for the analysis of qualitative data. Figure 3.2 shows that the spiral process consists of several procedures that have to be conducted before the analyzed data is converted into an account or a narrative. The first procedure in the spiral is data management where the data collected from the interviews is arranged and prepared for analysis. The hand-written notes from the interviews were converted and processed into easy to locate text units organized as a database stored in a Microsoft Word file. The second procedure in the spiral process after organizing the data is getting a sense of the whole database transcript [33]. Agar [40] suggested reading the entire transcript several times, and adding notes and remarks while getting immersed into the details of the interviews. The third procedure consisted of classifying, interpreting the database entries, and drawing the conclusions of the study. Conclusions on the results of the study were based on the interviewees’ answers, feedback, and author’s observations made during the interviews. The conclusions were derived by applying the direct interpretation.
3.5. Validation and Evaluation Procedures

Validation is one of the strong features of a qualitative research study because its accuracy is enhanced while the study is ongoing for a number of reasons. These reasons are, the amount of time spent on the case study, detail level of the collected data, and the close relation established with the participants. A strategy that was used to further validate the study was collecting data using the triangulation approach. In this approach, more than one data source was used to collect data with regard to the case study problem. The main purpose of triangulation approach is to inspect different dimensions of the same problem accurately rather than cross validating the data [33]. The author used the interview questionnaire, interviewee feedback, and observation as sources of data during the case study interviews.

Creswell proposed criteria for evaluating a qualitative research study which included clear identification, understanding and description of the problem being investigated. Furthermore, Creswell emphasized on the importance of correctly identifying the patterns emerging from the case analysis and the role of the researcher in the study [33]. Following Creswell's criteria, the author relied on state-of-the-art material to build an understanding of the problem statement of the thesis and for generating the material used in the case study. In addition, the author developed rapport with the study participants to enhance communications with them. This resulted in improving the quality of the collected data, which combined with reflections, and in-depth analysis contributed to successfully identify the case patterns. Finally, the author’s role in the qualitative study was that of an observer in order not the effect the opinion of participants and results of the case study.

Summary

This chapter defined the type of research methodology used as a qualitative research study and the applied approach as a case study approach. The case study characteristics are defined as a single, holistic, instrumental and exploratory case study. Thereafter, the chapter presented the different procedure followed in conducting the qualitative case study. The data collection procedure followed the Creswell data collection cycle and relied on the author’s observations, interviewees’ feedback and the interview protocol questionnaire as data sources. The one-to-one interviews used an interview protocol and were open-ended. Author’s role during the interviews was that of an observer, and observations were recorded as hand-written notes. In addition, a participant sample of seven engineers was provided by the Scania management and fulfilled the maximum variation sampling approach. The data analysis procedures applied the Creswell spiral process to analyze and interpret the collected data. The conclusions were derived from applying the categorical aggregation techniques and the direct interpretation analysis. Finally, the validation and evaluation section applied both the triangulation approach, and criteria proposed by Creswell to validate respectively evaluate the case study.
4. Supportive Methods

This chapter presents the supportive methods that resulted from the literature survey and the feedback from the preliminary interviews. The intended use of these supportive methods is to provide feedback on the behavior of the formalized requirements' semantics. Each supportive method is presented with a brief introduction, and a description of its graphical notation. The final section of this chapter presents a table that provides a comparison between the different features of the supportive methods.

For each supportive method in this chapter, a representation example consisting of a “Response” pattern has been provided. The SPS “Response” pattern represents a qualitative cause-effect relation between two propositions P and S. The “Response” pattern CNL states that “If P holds, then S will eventually hold” [12]. A simple requirement example that follows this pattern would be “if Signal_A = True, then Signal_B = False”, where “Signal_A = True” represents the proposition P, while “Signal_B = False” represents the proposition S. Propositions can be either simple containing a single logical relation, or compounded containing more than one logical relation connected by logical connectors. Some of the logical connectors that can be used are such as and (&), or (|), and negation (!).

4.1. Supportive Methods from the Literature Survey

A literature survey was performed to find the state-of-the-art supportive methods and six of them were selected after applying criteria. Most of these six supportive methods were originally proposed by researchers to facilitate a requirement formalization process by using the Dwyer et al. SPS. Generally, each of the six supportive methods uses two levels to represent a specification property behavior. The first level uses formal notation to represent the patterns’ temporal behaviors. Examples of the formal notations are, such as Linear Temporal Logic (LTL) [43], Computational Tree Language (CTL) [44], and Timed Computational Tree Language (TCTL) [45]. The second level is the user level, and consists of graphical notations mapped to relevant formal notations. This study will only use the graphical level of the supportive methods to provide feedback with regard to requirement formalization semantics.

4.1.1. A Graphical Property Specification Language

Lee and Sokolsky [46] proposed a method called Graphical Property Specification Language (LR) to facilitate the formalization process of system properties. LR expresses a system property behavior on two different levels. The first level is the expert level which uses the formal notation $\mu$-calculus which is an extension of the propositional modal logic [47]. The second level is the user level which uses an extendable graphical notation mapping the behavior of the formal notation. The user level hides the formal logical expressions, and provides a user-friendly interface for the practitioners who are not experts in formal logics.

The graphical notations of the user level in LR are presented in Table 4.1 and are inspired by the concept of constraint graphs [48]. The LR consists of nodes sequentially placed along a progression line that resembles the process. This method allows multiple paths between the nodes which represent different elements and actors such as propositions, logical connectives, modal and temporal operators. A logical connective can be such as an “and”, “or”, or a negation “!””. A modal operator is used to express a next state input or an output event. A temporal operator represents a path property such as “Eventually”, “Next” or “Always”. Temporal operators can also be supplemented with modifiers, which can add constraints on time, path, or for path termination [46]. Figure 4.1 shows an example of an LR representation of a simple real-time cause-effect relation. This example was provided by Lee and Sokolsky.
Lee and Sokolsky suggested constructing libraries containing LR representations of common reoccurring system properties. Users with no experience in formal languages can then formalize requirements by matching user level pattern expressions from the libraries to an intended behavior.

Table 4.1: LR Graphical Notation.

<table>
<thead>
<tr>
<th>LR Example Diagram Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>always</td>
</tr>
<tr>
<td></td>
<td>eventually</td>
</tr>
<tr>
<td>3</td>
<td>request!</td>
</tr>
<tr>
<td></td>
<td>response?</td>
</tr>
<tr>
<td>4</td>
<td>within [0,5]</td>
</tr>
<tr>
<td>5</td>
<td>avoiding reset</td>
</tr>
</tbody>
</table>

Figure 4.1: Simple LR Real-Time Cause-Effect Relation [46].

The LR method is user-friendly, and has different types of operators making it easy to represent formal expressions. However, it should be noted that the LR does not support the SPS. In addition, there was not enough information available about the LR method, for example, not all the graphical notation information was provided. Lee and Sokolsky referred to a research paper containing detailed information about the LR, but the author was not able to locate that paper. To use this supportive method in the study, the LR had to be modified to further simplify and adapt it for representing the SPS patterns. Modifications included a textual approach to address the SPS CNL ambiguities, and consisted of using textual comments. This modified graphical notation is referred to as the Modified Graphical Property Specification Language (MLR) supportive method. Table 4.2 presents the graphical notation of the MLR. The following modifications were introduced on the LR user level to enhance clarity and improve the information communicated to the user:

1. Cause-effect indicator: To express a cause-effect relation.
2. Pattern template: Pattern expressions are encapsulated in a template which is used to provide information about the pattern. This information includes pattern name, pattern description and any additional remarks, which could be difficult to visualize such as CNL ambiguities.
3. Elements container: Is a container for encapsulating and grouping diagram elements for the purpose of enhancing clarity.
4. Coloring: Element coloration is introduced to easily distinguish different elements.
5. Using textual temporal representations only instead of both symbolic and textual.
6. Enhancing clarity through using different geometrical shapes for different operator types.

Table 4.2: MLR Notation Diagram Symbols.

<table>
<thead>
<tr>
<th>Modified LR Diagram Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

Figure 4.2 shows an example of the “Response” pattern semantics represented in MLR. P is the cause proposition while S is the effect proposition. The cause-effect indicator displays “Always followed by” to indicate a “Response” type relationship between the cause and effect. The effect S in Figure 4.2 is preceded by the temporal operator “Eventually” to indicate the expectation of when S holds. The pattern template in Figure 4.2 shows the pattern name and displays the behavior of the pattern. The message “The effect S can be stimulated by other different causes and not only by P” is displayed to alert users to a behavior in the “Response” pattern which is not described by its CNL.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>P must always be followed by S</td>
</tr>
</tbody>
</table>

The author made MLR representations of the SPS qualitative and real-time patterns, and are found in Appendix A. Using the MLR language requires basic knowledge in MLR notation, and the temporal operators “Eventually”, “Always,” “Next”…etc. Lee and Sokolsky did not provide any
evaluation information comparing the LR method to other methods with regard to the requirement formalization.

4.1.2. Real-Time Graphical Interval Logic

Ramakrishna et al. [49] proposed a Real-Time Graphical Interval Logic (RTGIL) method for representing the temporal behavior of Konrad and Cheng SPS patterns. The method provides two levels mapped to each other for representing system properties. The first level is called the Graphical Interval Logic (GIL) which provides a visual representation of the system properties' behaviors. The second level is called the Future Interval Logic (FIL) which provides temporal logic expressions for the system properties’ behaviors represented by GIL. Moser et al. [50] extended the work of Ramakrishna et al. [49] by developing the RTGIL, and the Real-time Future Interval Logic (RTFIL) to represent real-time properties [13, 49].

Table 4.3: RTGIL Method Diagram Symbols.

<table>
<thead>
<tr>
<th></th>
<th>RTGIL Diagram Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Execution context: Represents an interval between two states where a certain property represented by a proposition can be evaluated.</td>
</tr>
<tr>
<td>2</td>
<td>Weak search: The RTGIL formula is true even when the search fails in finding a required state. Meaning the proposition may or may not hold on the interval.</td>
</tr>
<tr>
<td>3</td>
<td>Strong search: The RTGIL formula is false if the search fails in finding a required state. Meaning the proposition must hold on the interval.</td>
</tr>
<tr>
<td>4</td>
<td>Proposition will hold at every state.</td>
</tr>
<tr>
<td>5</td>
<td>Proposition will hold happen at next state.</td>
</tr>
<tr>
<td>6</td>
<td>Time-duration constraint where the duration (len) is defined by a minimum (d) and maximum (D) time values.</td>
</tr>
</tbody>
</table>

RTGIL categorizes system properties according to the location where they hold in an interval. In the "Initial property" category, the property holds at the beginning of the interval. The "Henceforth Property" category denotes a property that holds during the entire interval. The "Eventuality Property" category expresses a property that holds at some point along the interval starting from the beginning of the interval up to but not including the last point of the interval. Finally, the "Real-Time Duration Constraints" category expresses a time-duration constraint len(d, D) where the duration (len) is defined by minimum (d) and maximum (D) time values [13, 49].
Figure 4.3 shows RTGIL representations of the “Response” pattern semantics with a "Globally" scope. The diagram’s graphical notation is explained in Table 4.3, and the diagram is read along both its horizontal and vertical axes. It is read horizontally from left to right and vertically from top to bottom. The behavior of the system property is represented by a sequence of events and time constraints. These events and constraints are distributed along a time-line describing the state progression of the system. The section of the time-line between two states is called an interval, which is the basic building block of GIL/RTGIL. An interval is the range where a property is checked for whether it holds or not. Each interval consists of a sequence of events aligned in a linear manner. Events of non-timed properties are described as discrete, while events of real-time properties are described as dense. Figure 4.3 shows a weak search represented by a single headed arrow which is preformed to check for whether P holds or not. Since the scope used here is “Globally”, the cause P can hold at any state along the progression line. Once P holds, then S must eventually hold. This is expressed by a second interval extending from where P holds and till the end of the first interval, meanwhile S is designated with the diamond symbol representing the temporal operator “Eventually” [13, 49].

Using the RTGIL method requires knowledge in in RTGIL notation, and in temporal operators such as “Eventually”, “Always”, “Next” … etc. The researchers did not provide any data with regard to evaluating the RTGIL performance in requirement formalization. SPS real-time pattern representations in RTGIL were provided by Konrad and Cheng in [13] and GIL SPS qualitative pattern representations are available in an online repository setup by Alavi et al. [51].

4.1.3. Runtime Monitoring Framework for Property Specification Patterns

Simmonds et al. [52] proposed the Runtime Monitoring Framework for Property Specification Patterns (RMF) method, which was used to check the correctness of website behaviors. RMF implemented the Unified Modeling Language (UML) 2.0 [53] sequence diagrams (USD) for representing system specification properties. Simmonds et al. introduced modification to the USD so that the RMF was suitable for representing Dwyer et al. SPS property patterns. The USD diagrams were then interpreted into Finite State Automata (FSA) [54] which was used as a formal notation to formalize property semantics. The RMF does not support Konrad and Cheng SPS real-time patterns, and representations of the Dwyer et al. SPS can be found in [52].

Figure 4.4 shows an example of a USD representation of an SPS “Response” pattern which is a qualitative cause-effect relation. In USD, the vertical dimension represents the relative time while the horizontal dimension represents the different states of the system. Each state/object is represented by the rectangular box with a vertical dashed line called a lifeline, and messages between objects are represented by arrows. The loop-block in Figure 4.4 continuously checks for the existence of P while the assert-block indicates that S must hold after P has held. Simmonds et al. have extended the USD by adding operators from the UML 2.0 [53] to enhance capturing of system properties. The added operators can be categorized into four groups. The first group is the “Compositional operators”, which contains the following operators: alternatives “alt”, parallel
(par), weak sequencing “weak seq”, and strict sequencing “strict seq” are used to join two USDs together. The loop, opt and critical operators are used for scenario repetition, scenario designation and atomicity (preventing parallel thread access) respectively. Furthermore, it is possible to modify the messages communicated between USD objects through using the “Alphabet changing operators” like (consider) and (ignore). A third group is “Assertion and negation operators”, which can enforce or disallow certain scenarios, by using the “assert” and “negate” operators, respectively. Finally, a new feature provided through the “ref” operator called “Interaction use operator” in UML 2.0 enables operators to invoke other USDs by referencing their names [52]. For a list of the operators belonging to each group, refer to [52], section 2. Further details regarding the operators are found in [55].

![SD response](image)

Figure 4.4: Response Pattern Semantics Represented in USD [52].

Simmonds et al. did not perform an evaluation on the effectiveness of the RMF method in comparison to other methods with regard to facilitating requirement patterning. Finally, using the graphical notation requires knowledge in USD and UML operators, and in LTL.

### 4.1.4. Use Case Maps Property Pattern System

Hassine [56] proposed the Use Case Maps Property Pattern System (UCMPPS) method to facilitate the process of requirements formalization process using Konrad and Cheng SPS. The UCMPPS is an extension of the Use Case Maps [55], and has graphical templates that can express functional, timing and even architectural properties. This method facilitates precise behavior analysis, while at the same time maintains a high level of abstraction. Each graphical template is mapped to a relevant formal notation which captures the temporal behavior of the SPS properties' patterns. Depending on the type of the template, the mapped formal notation can be either CTL, or TCTL temporal languages, or to Architectural TCTL (ArTCTL) [56].

The goal of UCMPPS method is to provide users who have little or no experience in formal notation with an easy to use graphical method to formalize requirements. Hassine provided a UCMPPS library of representations expressing the SPS pattern semantics on two levels. The first level consists of user-friendly graphical templates, and the second level consists of formal logical expressions representing the patterns’ temporal behaviors. Each element in the first level is
mapped to a relevant element in the second level. A list of UCMPPS graphical notation symbols is provided in Table 4.4.

Figure 4.5 shows a UCMPPS representation of the “Response” pattern semantics which is a cause-effect relation. In this figure the (*) appended to the proposition P (cause) indicates that when P holds then S (effect) should eventually hold. Hassine’s study compared features of different methods and resulted in the development of the UCMPPS [56]. However, Hassine did not provide an evaluation of the UCMPPS in comparison to other methods with regard to facilitating requirement formalization. Using this method requires knowledge in the UCMPPS notation. UCMPPS SPS representations made by Hassine, are be found in [56].

Table 4.4: UCMPPS Diagram Symbols.

<table>
<thead>
<tr>
<th>UCMPPS Diagram Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

Figure 4.5: Response Pattern Semantics Represented in UCMPPS [56].

4.1.5. Template Based Approach for Temporal Logic Visualizing

Comella-Dorda et al. [57], developed a template based method to facilitate the requirement formalization process for users who have little or no experience in formal notations. The method is called Template Based Approach for Temporal Logic Visualizing (TBATLV) and represents system properties on two levels. The first level is a CTL formal notation representing the property temporal behavior. The second level consists of a graphical representation in the form of a flow-diagram that consists of segments called “Specification Templates”. Each specification template is mapped to a relevant CTL expression on the first level [57]. The specification templates are fine grained patterns based on Dwyer et al. SPS. The name specification templates was coined by Comella-Dorda et al. in order to distinguish them from Dwyer et al. SPS property patterns. Following, are some of the characteristics of the specification templates [57]:
- Specification templates have names and contain all the necessary information for the practical implementation.
- Specification templates capture the basic structure of temporal logic and use symbols instead of notations to generalize the problem.
- Specification templates represent solutions to reoccurring problems.
- Specification templates can be developed to represent properties of specific systems. Consequently, they will have different structures.

The specification templates express the basic behavior of Dwyer et al. SPS patterns, and are classified accordingly. Comella-Dorda et al. applied criteria for choosing the specification templates to include in the flow-diagram. The first criterion is “Simplicity”, and the second criterion is “Correspondence to an expected property”. According to the first criterion, templates which are too complex or too simple should not be included. The second criterion, ensures that the selected templates should express common behaviors shared by different systems. Therefore, TBATLV method aims to provide a simple and fast solution for requirement patterning, but not a complete solution. The reason is that, the specification templates developed by Comella-Dorda et al. aim to represent the most commonly reoccurring system properties only. Hence, completeness of the specification templates is not guaranteed as there are many system properties not covered by this method [57].

Comella-Dorda et al. did not provide representations for any of the SPS patterns. To use this supportive method in the study, specification template representations of the SPS scopes, and patterns were made by the author and are found in Appendix B. Additionally, this supportive method has been modified to address the SPS CNL ambiguities using a textual approach, which consists of textual comments to alert the user.

Figure 4.6 shows a highlighted path traversing the template flow-diagram to represent the “Response” pattern temporal behavior. TBATLV is useful in guiding users and especially the beginners through the process of building up claims. A claim represents a system property behavior, which can be built by traversing the flow-diagram [57]. TBATLV does not require knowledge in temporal logics, and during the study, it was observed that this supportive method was easy to use and could be a good learning tool.
Figure 4.6: Tracing the Response Pattern through the TBATLV Template Flow-Diagram.
4.1.6. Finite State Automata Property Pattern Templates

Finite state machine or FSA is a mathematical computational model used for representing state driven systems. When system events or conditions change, the FSA reacts to these changes by switching from one state to another. Conceptually, an FSA consists of a finite set of states, input/output symbols, mapping function(s), output function(s), and a set of accepted states. The FSA state at any given time instant is called a “Current State”, and changing from one state to another is called a “Transition”. FSA has a finite number of states and finite number of possible transitions [54, 58]. There are two types of FSAs, deterministic and non-deterministic. In a deterministic FSA, an input is required to initiate a state transition. For a combination of a given input and a given transition condition, there is only one possible output state. On the other hand, a non-deterministic FSA can have more than one output state for a combination of a single input and a transition condition. Additionally, a state transition in a non-deterministic FSA can be initiated by a transition condition change only, without necessarily the need for an input change. This is called an epsilon transition [59, 60].

Table 4.5: FSA Property Pattern Templates Graphical Notation.

<table>
<thead>
<tr>
<th>Modified Finite State Automata Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

The initial inclination was not to include the FSA as a supportive method in the case study material. The reason was that during the literature survey, attempts that were made to find FSA diagrams representing the real-time SPS patterns’ temporal behaviors did not produce results. Similarly, results of attempts aimed at finding tools or programs, which could generate real-time SPS FSA representations were unsuccessful too. However, the FSA was finally included into the study because some Scania engineers used the FSA in their work. Additionally, temporal logic formulae are commonly translated into FSA diagrams. This thesis uses the deterministic FSA SPS pattern representations generated by the PROPEL tool, which was developed by Cobleigh [11]. The PROPEL tool uses an extended form of FSA called the Finite State Automata Property Pattern Templates (FSAPPT) to represent the behavior of the qualitative SPS patterns. Table 4.5
presents the graphical notation of the FSAPPT. The reason for using PROPEL tool representations was because PROPEL generates clearer SPS qualitative pattern diagrams than any other tool tested in this study. FSAPPT does not support SPS real-time patterns; however, if the FSAPPT method were chosen by the interviewees, then FSA SPS real-time representations have to be developed.

Figure 4.7 shows an FSAPPT representation of the SPS “Response” pattern semantics. The diagram displays two accepting states, which are marked with concentric circles. The concentric circle to the left represents the start state, and it is marked with an incoming arrowhead. Transitions P (cause) and S (effect) indicate the direction of the flow in the FSA. If P does not hold, then there is no transition of states and the start state will be the accepting state. However, if P holds, then the FSA state transitions to the second state and remains there as long as S does not hold. Finally, the FSA reaches the second accepting state when S holds, and the state remains unchanged until P holds leading to a transition to the previous state.

![Figure 4.7: FSAPPT Representation of the Response Pattern.](image)

Using FSAPPT as a supportive method requires knowledge in the FSAPPT graphical notation. Cobleigh did not provide an evaluation of the PROPEL method in comparison to other methods with regard to facilitating requirement patterning. More FSAPPT SPS representations can be generated using the PROPEL tool [11].

4.2. Supportive Methods from the Preliminary Interviews Feedback

The second source for the supportive methods was from the feedback gathered during the case study preliminary interviews. This feedback resulted in the addition of one more supportive method the MTD.

4.2.1. Modified Timing Diagram

The idea of the Modified Timing Diagram (MTD) was suggested by a case study participant with the aim to use it as an aid to represent CNL ambiguities for the study participants. The MTD draws inspiration from both the Timing Diagram (TD) and the RTGIL. The TD is an informal graphical method usually used in hardware design to represent signal levels [61, 62]. The use of TD is not limited to representations of hardware specification; it is also used by logicians to represent temporal logic formulae [63, 64]. Dillon et al. [63] concluded that linear textural representations display complex timing properties clearer than other types of visualizations.

Figure 4.8 shows an MTD representation of the SPS “Response” pattern semantics with the goal of displaying two of its temporal behavior’s features. The first feature denotes that if any of the causes X, Y, or P held, then the effect S will eventually hold following any one of them. The second feature denotes that the effect S is not dependent on a single cause, but can be stimulated by any of the causes X, Y, or P. The vertical axis in the figure is used to display the Boolean value of the cause proposition which is either true or false, while the horizontal axis represents the program execution time. The line below the horizontal axis shown in Figure 4.8 represents an interval where the effect S is represented.
Using the MTD as a supportive method requires knowledge in the TD, RTGIL graphical notations, and in temporal operators. The author created several MTD representations for the SPS, and they are available in Appendix C.

4.3. Supportive Methods Features Comparison Table

The supportive methods that were found through the literature survey, and the preliminary interviews varied in their features and suitability to be used as supportive methods. Some of these supportive methods required some work, to be included in this study. Table 4.6 provides feature comparison between different supportive methods, which were used in this study. The supportive methods' features included in this table are, supported SPS patterns' types, method modifications, and if the supportive method addressed CNL ambiguities or not. In addition, the table includes brief comments by the author on each supportive method.

Summary

This chapter presented seven supportive methods, which resulted from the literature survey and the feedback of the preliminary interviews. These seven supportive methods were briefly presented, and their graphical notations were explained. Furthermore, the supportive methods' presentations included an SPS “Response” pattern example using each method's graphical notation. Finally, the last section in the chapter presented a feature comparison table between the supportive methods used in this study.
Table 4.6: Supportive Methods Feature Comparison Table.

<table>
<thead>
<tr>
<th>S.N</th>
<th>Supportive Method Name</th>
<th>Represented SPS Patterns</th>
<th>Thesis Author Modifications</th>
<th>Can Represent CNL Ambiguities</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Qualitative Patterns</td>
<td>Real-Time Patterns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MLR</td>
<td>Yes, added by thesis author</td>
<td>Yes, added by thesis author</td>
<td>Yes</td>
<td>Yes, with textual descriptions</td>
</tr>
<tr>
<td>2</td>
<td>RTGIL</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>High level of detail. Could be experienced more sophisticated than other methods, requires knowledge in temporal logic and RTGIL notation.</td>
</tr>
<tr>
<td>3</td>
<td>RMF</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Could be experienced as less user-friendly because it encapsulates diagrams and uses math expressions. Requires knowledge in SD, SD operators and LTL.</td>
</tr>
<tr>
<td>4</td>
<td>UCMPPS</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>User-friendly method, and requires knowledge in UCM notation and LTL operators.</td>
</tr>
<tr>
<td>5</td>
<td>TBATLV</td>
<td>Yes, added by thesis author</td>
<td>Yes, added by thesis author</td>
<td>Yes</td>
<td>Yes, with textual descriptions</td>
</tr>
<tr>
<td>6</td>
<td>MTD</td>
<td>Yes, added by thesis author</td>
<td>Yes, added by thesis author</td>
<td>Yes</td>
<td>Yes, with graphical representations</td>
</tr>
<tr>
<td>7</td>
<td>FSAPPT</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
5. Case Study Implementation and Findings

This chapter discusses the case study implementation procedures which consist of conducting a literature survey, preparing the case study material, and performing both the preliminary interviews and the interviews. This chapter also presents the findings of the case study which includes discussions, analysis and conclusions on the results of the study, and presents its account.

5.1. Case Study Implementation Procedures

One of the objectives in this thesis is to facilitate the requirement formalization process by providing supportive methods’ feedback with regard to requirement formalization semantics. The purpose of the supportive methods’ feedback is to guarantee that the formalized requirements’ semantics matched the intended behavior of the requirement developers. An approach described in Figure 5.1 was followed, which consisted of a number of procedures. A literature survey was performed, and criteria were applied to find supportive methods, which were then prepared to be used in the case study. Next, an interview protocol was developed and preliminary interviews conducted to evaluate it and get general feedback on the case study material. Afterwards, interviews were conducted with Scania engineers, and the data was collected from questionnaire answers, feedback, and author’s observations. The collected data was then analyzed, and the findings were in the form of emerged patterns and suggestions from the interviewees. Finally, the case study account was provided.

5.1.1. Literature Survey

A literature survey was conducted as the first step of the case study implementation. The purpose of the literature survey was to find supportive methods that could provide informative feedback with regard to the formalized requirements’ semantics. The primary source for the material of the literature survey came from research journals and conference articles found on Google Scholar. Other databases used to collect data included Springer Link, ACM Digital Library and IEEE Xplore Digital Library. The search was performed using combinations of the following keywords “Formal notation”, “Graphical view”, “Property patterns”, “Finite state”, semantics”, “Konrad”, “Dwyer”, “FSA”, “Sequence diagram”, “Precedence”, “Response”, “Time” and “LTL”.

The gathered data from the literature survey was examined and then filtered according to criteria to narrow the material selection for the case study. The conditions of these criteria were as follows:

1. Candidate supportive method should be a state-of-the-art scientific method.
2. Candidate supportive method should be user-friendly, and should not be demanding with regard to having a long learning curve.
3. Candidate supportive method should support SPS qualitative and real-time patterns, or could be modified to represent them.
4. Candidate supportive method should address SPS CNL ambiguity issues, which were mentioned by Filipovikj.

These criteria conditions were not rigorously applied, because the literature survey did not yield sufficient results that satisfied the criteria condition, and there was a time limit for performing the literature survey. As a result, exemptions were made with regard to almost every criteria condition and specifically condition 4 which no state-of-the-art supportive method satisfied. On the other hand, the only supportive methods that met all of the criteria conditions except for condition 4 were the RTGIL and the UCMPPS. The RMF and the FSAPPT did not fully meet conditions 2, 3 and 4. However, they were included because some Scania engineers were familiar with using the USD and FSA.
The MLR and the TBATLV supportive methods did not meet conditions 3 and 4. They were included in the study because of their user-friendliness, and were modified to satisfy both conditions 3 and 4. They were modified among other things to address CNL ambiguities using a textual approach. However, by modifying the MLR, and the TBATLV they then had to be exempted from condition 1. The MLR, and the TBATLV were included in the study to test the reaction of the study participants towards CNL ambiguities. Another supportive method that was exempted from condition 1 was the MTD, which was suggested by a case study participant to address CNL ambiguities using a graphical approach. The MTD is a combination of both of a TD and the RTGIL to illustrate CNL ambiguities.

5.1.2. Case Study Material Preparation

The literature survey resulted in six state-of-the-art supportive methods and their usability as material for the case study was investigated. The investigation resulted in creating SPS representations, and introducing modifications for some of the supportive methods. However, results from the investigation also showed that none of the state-of-the-art supportive methods addressed the CNL ambiguity issues which were discussed by Filipovikj.

The RTGIL and the UCMPPS were the only state-of-the-art supportive methods that provided support and representations for both of the SPS qualitative and real-time patterns. With regard to the LR supportive method, its graphical notation was not fully explained, and no SPS pattern representations were provided by the researchers. The LR could therefore, not be included in the case study. However, this method appeared to be user-friendly and was therefore modified into the MLR supportive method. The modifications included adding textual information to the MLR to address the SPS CNL ambiguities. In addition, MLR representations for both of the qualitative and real-time SPS patterns’ semantics were created. The TBATLV supportive method was another user-friendly supportive method, which lacked representations for both the qualitative and real-time SPS patterns. These SPS representations were made, and textual information was embedded in the TBATLV supportive method to address the SPS CNL ambiguities. The RMF and FSAPPT supportive methods provided only qualitative SPS pattern representations, but no real-time SPS pattern representations were made for these supportive methods due to the lack of time. In general, the supportive methods included within this study differ in how they represent the SPS patterns’ temporal behaviors and in the complexity level of their representations. Using these supportive methods requires different learning curves, and they provide different levels of user-friendly experiences.

The supportive methods that resulted from the literature survey were used in developing an interview protocol which was used for gathering data during all the case study interviews. The interview protocol consisted of an introduction segment and a questionnaire segment. The introduction segment introduced and explained both the SPS and the supportive methods to the interviewees. The introduction segment was verbal, but it also relied heavily on diagrams and examples. The second segment of the interview protocol was a questionnaire which consisted of a number of assignments. The assignments were designed to guide the users to explore the effectiveness of different supportive methods with regard to providing feedback expressing the SPS patterns’ temporal behaviors. The case study questionnaire is provided in Appendix D.
Figure 5.1: Case Study Design Diagram.
5.1.3. Preliminary Interviews

Preliminary interviews were conducted to evaluate the interview protocol and provide an opportunity for the interviewer to practice. Table 5.1 shows a list of preliminary interviews' participants. Two of the interviewees who took part in the preliminary interviews were fellow students working on a thesis that involved using the SPS and were therefore experienced in it. A third interviewee was one of the engineering supervisors at Scania, and had no previous experience with the SPS.

All the preliminary interviews took place at Scania, and each session lasted between 60 to 90 minutes, as shown in Figure 5.2. The preliminary interviews started with a general introduction presenting the purpose of the case study, and followed the interview protocol which was developed during the case study material preparation phase. The interview protocol contained two segments, an introduction and a questionnaire. The introduction segments introduced the interviewees to the SPS and the supportive methods. The questionnaire segment at that time had only Assignment 1 available in Appendix D, which was accompanied with discussion periods. Finally, the questionnaire segment was ended with discussions on the SPS and supportive methods. The results of the preliminary interviews helped in improving the interview protocol and added additional supportive method to the case study material.

Table 5.1: Preliminary Interviews Participants List.

<table>
<thead>
<tr>
<th>Interviewee Number</th>
<th>Position</th>
<th>Previous SPS Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software Engineer doing Master Thesis.</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Software Engineer doing Master Thesis.</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Software Engineer at Scania.</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 5.2: Preliminary Interviews Structure. The Interviews used the Interview Protocol.
5.1.4. Case Study Interviews

The main purpose of the interviews was to find the supportive methods preferred by the engineers based upon the methods’ feedback on the formalized requirements’ semantics. The case study interviews were conducted with a sample of seven Scania engineers as shown in Table 5.2. Figure 5.3 shows the structure of the case study interviews that took place at Scania and lasted between 60 to 90 minutes for each session. Interviewees’ comfort was of high priority during the interviews which were organized to accommodate the interviewees’ schedules, and took place at the location they suggested. The interviews used the interview protocol given in Appendix D, which consisted of two segments, an introduction, and a questionnaire. An informative introduction segment was necessary to introduce the interviewees to requirement formalization, SPS, and the supportive methods. The introduction segment took on average between 30 to 45 minutes and was in two parts. In the first part, the engineers were given information about the SPS, temporal operators, pattern types, pattern categorization, scope types, CNL ambiguities, and examples on how to use the SPS. In the second part of the introduction segment, supportive methods were introduced and explained to the interviewees along with examples on how to use them.

The second segment of the interview’s protocol was the questionnaire segment. The main purpose of this segment was to evaluate the supportive methods’ feedback in representing formalized requirements’ semantics, and it consisted of two main assignments accompanied with discussion periods. The first assignment was a requirement patterning assignment where the interviewees read a requirement, then selected a proper pattern to formalize it. The interviewees were also asked to use the supportive methods’ feedback to guarantee that the formalized requirement semantics matched the intended behavior. This assignment was designed to guide the interviewees to formalize requirements without external help. The second assignment was a comparison between formalized requirements’ behaviors. The purpose of second assignment was to investigate the interviewees understanding of the resulting effects of applying different SPS patterns on the same requirement. Another goal of the second assignment was to investigate whether the interviewees could detect CNL ambiguities or not. All assignments provided step-by-step instructions on how to perform the exercises. However, the interviewees had difficulties with the second assignment, and needed guidance from the author to complete it.

Table 5.2: Case Study Interviews Participants List.

<table>
<thead>
<tr>
<th>Interviewee Number</th>
<th>Position</th>
<th>Previous SPS Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RESA Manager</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Software Architect</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Product Owner</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Software Developer</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Software Developer</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Software Tester</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Software Developer</td>
<td>No</td>
</tr>
</tbody>
</table>
5.2. Case Study Findings

This section presents the outcome of the case study and provides discussions, analysis and conclusions on the collected data during the study. The case study preliminary interviews and interviews findings are based on the questionnaire results, author’s observations and the feedback gathered from the interviewees.

5.2.1. Preliminary Interviews Results

The feedback and observations gathered from the preliminary interviews resulted in a number of changes, which were introduced into the interview protocol, and the case study material. The feedback led to the addition of one more supportive method the MTD to the case study material. The concept of the MTD was suggested by a fellow engineering student and further developed by the author of this thesis. The author created a number of MTD SPS diagrams, which are found in Appendix C. A second outcome of the preliminary interviews was the addition of Assignment 2 to the questionnaire segment of the interview protocol. The purpose of Assignment 2 was to investigate if the interviewees could detect CNL ambiguities when applying different SPS patterns on the same requirement. A third outcome of the preliminary interviews was to provide step-by-step instructions on how to perform the assignments.

Results of the interview protocol questionnaire are shown in Table 5.3. The two interviewees who were experienced with the SPS chose the MLR as their selected supportive method. On the other hand, the third interviewee who was a Scania engineer and not experienced in the SPS selected the FSAPPT, because he usually used it at his work.

<table>
<thead>
<tr>
<th>Interviewee Number</th>
<th>Previous SPS Experience</th>
<th>Selected Method</th>
<th>Addresses CNL Ambiguities?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>MLR</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>MLR</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>FSAPPT</td>
<td>No</td>
</tr>
</tbody>
</table>
5.2.1.1. Adding the Modified Timing Diagrams (MTD) Method

While conducting the preliminary interviews, an interviewee suggested using the TD to represent CNL ambiguity issues. These CNL ambiguities could neither be understood from reading the CNL, nor could they be graphically represented by any of the mentioned state-of-the-art supportive methods. The TD was modified into the MTD to make it more suitable for representing the SPS patterns’ temporal behaviors.

The first ambiguity issue to be represented by the MTD was the difference in behavior between the “Response” and the “Precedence” patterns. Both the “Response” and the “Precedence” patterns represent a cause-effect relation. However, there are important features in each pattern’s temporal behavior which the CNL does not state. The signature feature of the “Response” pattern which is not stated in its CNL is that its effect can be stimulated by more than one cause. However, the cause in the “Response” pattern can only stimulate one effect at the time. Meanwhile, the situation is vice versa in the “Precedence” pattern which effect can only be stimulated by one specific cause. However, the cause can stimulate more than one effect at the same time [8]. Following are two examples that were used to explain the first ambiguity issue for the interviewees:

Figure 5.4 shows a simple example to represent the temporal behavior of the “Response” pattern. The example consists of an electric circuit which contains one lamp and several switches all connected in parallel. If the switch S1 (cause) is in a closed position, then lamp L1 (effect) will therefore be set on. However, L1 can also be set on by other switches (S2 and S3).

![Figure 5.4: Simple Example Explaining the Response Pattern.](image)

The second issue is with regard to ambiguous CNL representation of the scopes “Before R” and “After Q”. From reading the CNL of these two scopes, the reader could be misled into understanding that the scope condition applies repeatedly during the program execution. However, this is not the case, as only the first occurrences of R and Q are taken into consideration. As a result, a pattern using the “Before R” scope holds only before first occurrence of R, thereafter, it never holds again no matter what the status of R has. Similarly, a pattern using the “After Q” scope will continue to always hold after the first occurrence of Q even when Q doesn’t hold any more [8].
5.2.1.2. User-guidance Supportive Method

The TBATLV supportive method is a visual questionnaire in the form of a flow-diagram. This supportive method is user-friendly and can help the user to identify an intended behavior through traversing a flow-diagram. The TBATLV was especially popular among the interviewees who were experienced in the SPS, and it was tested in patterning a number of difficult Scania requirements. The TBATLV was therefore used during the interviews for the purposes of training and as a learning tool for the SPS. Additionally, the TBATLV was integrated in the Scania Specifier tool to provide a user-guidance option for the requirement developers.

5.2.2. Case Study Interviews Results

Seven Scania engineers from different departments were interviewed in this case study. The interviewees were given a questionnaire and were requested to redo it more than once. These repetitions were accompanied by discussions between the interviewer and the interviewees regarding the SPS and the supportive methods. There were three reasons for the repetitions, the first reason was to provide the interviewees with more practice time, especially that they lacked experience in the SPS and supportive methods. The second reason was that some of the supportive methods lacked representations for the SPS real-time patterns’ semantics. Repetitions were necessary to give interviewees the opportunity to reconsider their decisions. The third reason was that these repetitions provided an opportunity to make observations and to gather feedback from the interviewees. Reporting the interviews results, the interviewees were coded as interviewee one, interviewee two … etc. Some interviewees initially preferred different supportive methods for representing different types of patterns, others were more consistent in their choices. All the interviewees were finally asked to select one preferred supportive method. The interviews’ results are chronologically listed below:

**Interviewee one:** Interviewee one worked in a managerial and research position, and his final selection was the RTGIL supportive method. Interviewee one evaluated several supportive methods but was consistent in his selections preferring the RTGIL over the rest of the methods. Following is interviewee one supportive methods selections were:

Supportive methods selections:
- Response: RTGIL
- Precedence: RTGIL
- Bounded Response: RTGIL
- Bounded invariance: RTGIL

Final selected supportive method: RTGIL

**Interviewee two:** Interviewee two was a software architect, and his final selection was the RMF supportive method. Interviewee two preferred using the RMF method for qualitative patterns and the MTD for real-time patterns. Interviewee two preferred the RMF method because he had...
experience in using the USD, and expressed his wish to learn more about it. With regard to real-time patterns, interviewee two preferred the MTD stating that it provided a clearer representation of complex temporal behaviors. The interviewee also stated that diagrams which did not provide users with more information than that provided by the CNL were not helpful. He also suggested providing textual examples as supportive method or to enhance clarity of existing supportive methods. Interviewee two supportive methods selections were:

Supportive methods selections:
- Response: RMF
- Precedence: RMF
- Bounded Response: MTD
- Bounded invariance: MTD

Final selected supportive method: RMF

Interviewee three: Interviewee three was a software developer and a product owner, and his final selection was the UCMPPS supportive method. With regard to the “Precedence” pattern, interviewee three had several remarks. In the first remark, he stated that the representation of the “Precedence” pattern semantics felt like reading backwards. Furthermore, he stressed the importance of the CNL to correctly describe the actual behavior of the SPS patterns’ temporal behaviors. Interviewee three also stated that, he would never use a requirement which did not correctly and accurately describe the behavior of the system property. Finally, interviewee three suggested making all diagrams available in the Specifier tool to allow users to choose the feedback which they prefer. The supportive methods selected by interviewee three were as follows:

Supportive methods selections:
- Response: UCMPPS
- Precedence: UCMPPS
- Bounded Response: UCMPPS
- Bounded invariance: UCMPPS

Final selected supportive method: UCMPPS

Interviewee four: Interviewee four was a software developer, who selected the UCMPPS supportive method despite preferring to use the FSAPPT supportive method. The reason was that interviewee four found the FSAPPT to be confusing sometimes. Interviewee four also liked the MLR method with regard to representing the SPS real-time patterns. However, his final decision was to choose the UCMPPS, because it was more user-friendly. Interviewee four supportive methods selections were as follows:

Supportive methods selections:
- Response: UCMPPS
- Precedence: UCMPPS
- Bounded Response: MLR
- Bounded invariance: UCMPPS

Final selected supportive method: UCMPPS

Interviewee five: Interviewee five was a software developer, and his final selection was the FSAPPT supportive method. Interviewee five preferred using the FSAPPT for representing SPS qualitative patterns and the MTD for representing SPS real-time patterns. In general, MTD was his second favorite choice in case FSAPPT was not available. Interviewee five also mentioned
that he would avoid writing requirements as a “Precedence” pattern, and his methods selections were:

Initial supportive methods selections:
- Response: FSAPPT
- Precedence: FSAPPT
- Bounded Response: MTD
- Bounded invariance: MTD

Final selected supportive method: FSAPPT

**Interviewee six:** Interviewee six was a software tester, and his final selected method was the MLR. Interviewee six liked both the MLR and UCMPPS methods. However, he experienced the use of temporal operators in the MLR method as confusing. Interviewee six was also confused with the representation of the “Precedence” pattern semantics in the UCMPPS because he experienced it as reading backwards. The interviewee’s methods selections were as follows:

Initial supportive methods selections:
- Response: UCMPPS
- Precedence: MLR
- Bounded Response: MLR
- Bounded invariance: MLR

Final selected supportive method: MLR

**Interviewee seven:** Interviewee seven was a software developer, and his final selected method was the RTGIL. Interviewee seven had difficulty in selecting a supportive method at the beginning, because he thought, he did not know enough about the supportive methods. The interviewee suggested providing the interviewees with more time to understand what each method can offer and to learn how to use them. The interviewee favored using the RMF with qualitative patterns despite that he did not fully understand the diagrams. The reason for the RMF selection was that the engineers at Scania were familiar with the USD. The interviewee also favored using the RTGIL with real-time patterns. After reconsideration, the interviewee selected the RTGIL method as his favorite method. Finally, the interviewee suggested providing the Specifier tool with representations from all of the supportive methods and letting the engineers decide which one to use. Interviewee seven supportive methods selections were as follows:

Initial supportive methods selections:
- Response: RMF
- Precedence: RMF
- Bounded Response: RTGIL
- Bounded invariance: RTGIL

Final selected supportive method: RTGIL
Table 5.4: Interviewee Final Supportive Methods’ Selections.

<table>
<thead>
<tr>
<th>Interviewee Number</th>
<th>Previous SPS Experience</th>
<th>Selected Supportive Method</th>
<th>Addresses CNL Ambiguities?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>RTGIL</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>RMF</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>UCMPPS</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>UCMPPS</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>FSAPPT</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>MLR</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>RTGIL</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 5.4 shows the case study interviewees’ final supportive methods’ selections. The RTGIL and the UCMPPS were the mostly favored supportive methods. The results listed in Table 5.4 are graphically displayed in Figure 5.6. During the interviews, the author discussed the interview’s results with each interviewee. All the interviewees expressed that supportive methods’ feedback helped in understanding the formalized requirements’ semantics, and made the formalization process more user-friendly.

![Graphical Representation of Interviewees Final Supportive Methods’ Selections.](image)

5.2.3. Case Study Discussions, Analysis and Conclusions

This section presents discussions, analysis and conclusions on the interviewees’ feedback and the observations made by the author during the case study interviews. The collected data was analyzed using the direct interpretation approach and resulted in recognizing a number of emerging patterns. These patterns were essential in understanding the interviewees’ experiences in using the SPS. Each emerged pattern section contains the discussions, analysis and conclusions to provide a clear context for each issue being discussed. It is important to take into consideration that the interviewees had no previous knowledge of the SPS patterns, nor the supportive methods used in this case study. This information was briefly introduced to them during the interviews. Understanding this context under which the case study interviews were conducted will help to understand some of the issues and difficulties that faced the interviewees.

5.2.3.1. Pattern I. Role of SPS experience in Supportive Methods Selections

All the case study interviewees were informed on the CNL ambiguity issues, and they were shown different examples illustrating these issues. Table 5.3, Table 5.4 and Table 6.1 list the supportive methods chosen by the interviewees in the preliminary interviews, the interviews and the evaluation of the Specifier tool. Results from the Specifier tool evaluation have been included.
because it follows the same pattern. The data from the tables show that most of the interviewees who were not experienced in the SPS chose supportive methods that did not address the CNL ambiguity issues. The exceptions were interviewee six in Table 5.4 and interviewee three in Table 6.1 who were inexperienced and selected the MLR which addressed the CNL ambiguities. On the other hand, Table 5.3 and Table 6.1 show that the two experienced SPS interviewees selected the same supportive method, the MLR. This leads to concluding that having experiences in the SPS influences the type of the supportive methods selected. It can be explained that interviewees who were experienced in the SPS knew the exact temporal behavior of patterns, and easily understood the effect of the CNL ambiguities. The SPS experienced interviewees focused on finding the feedback that adequately described the pattern temporal behavior and at the same time addressed the CNL ambiguity issues. On the other hand, the non-experienced interviewees did not have time to develop a deep and accurate understanding of the SPS patterns’ temporal behaviors. Consequently, the non-experienced interviewees did not exactly know what to expect from the graphical feedback. Thus, they were more focused on interpreting the diagrams to match them with the contents of the SPS patterns’ CNL.

The role of the participants’ knowledge in the SPS on the case study results could also be noticed when comparing the results of two previous case studies. Both Post et al. [17], and Filipovikj et al. [5] conducted case studies assessing the applicability of the SPS in the automotive domain. Post et al. performed a case study at BOSCH, and the participants of the study were engineers. The results of the case study indicated that the 80% of the engineers preferred using the “Globally” scope, and 50% preferred using the “Absence” or “Bounded response” patterns. Filipovikj et al. performed his case study at Scania, and the participants in his study were researchers. The results of the case study showed that the researchers mostly preferred using the “Precedence” and the “Universality” patterns. According to Filipovikj et al., the differences in the results were attributed to the different types of participants in the case studies. Researchers were more inclined than engineers to develop an accurate understanding of the SPS patterns and scopes, and to use as much of them as possible when formalizing requirements [5].

5.2.3.2. Pattern II. Difficulties with some SPS Patterns and Temporal Operators

The interviewees who took part in the case study were enthusiastic about inspecting the features of different supportive methods, and were eager to learn more about them. However, some interviewees were not comfortable with using the temporal operator “Eventually” experiencing it as confusing. This was noticed when using the CNL of some patterns, and the graphical notations of some supportive methods that contained the temporal operator “Eventually”. The difficulty in using the temporal operator “Eventually” can be attributed to two reasons. The first reason was that all the interviewees work with real-time systems where events are expected to be accurately scheduled. However, the operator “Eventually” means that a proposition will hold at least once at some point during the program execution. This introduces an element of uncertainty with regard to when the proposition holds, which makes the engineers uncomfortable using this operator when dealing with safety-critical systems. It was therefore, important to clarify and explain the proper use of the operator “Eventually” to the interviewees when introducing the SPS.

The second reason was that some real-time requirements were being mistakenly classified as qualitative requirements, which lead to patterning them with the wrong patterns. The requirements used as examples in this study were inspired by existing requirements used to describe real-time system properties. An issue with some of these requirements was that they could be lacking information with regard to time constraints. This is a known issue which occurs when developers omit information they consider as being common knowledge. As a result, the affected requirements were wrongly perceived as qualitative instead of real-time requirements. Consequently, the wrong patterns were used to formalize these requirements. To solve this issue it is essential that time constraints are not omitted in order to distinguish between qualitative and
the real time requirements. The issue was also discussed by Filipovikj et al. who stated that it is
difficult to choose the correct type of patterns when formalizing an ambiguous requirement [5].
Rodriguez-Navas et al. [10] stated that the design process in the automotive industry is a very
human-intensive activity and therefore, is error-prone. Rodriguez-Navas et al. mentioned the issue
of requirement ambiguity as part of the challenges facing requirement formalization in the
automotive industry.

Furthermore, the concept of propositions preceding each other with regard to holding was also
experienced as confusing to some interviewees. This issue was noticed with the “Precedence”
pattern which states, “It is always the case that if S holds, then P previously held” [12].
Interviewee five stated that he would avoid writing requirements as a “Precedence” pattern. While
interviewees three, and seven perceived a diagram in Figure 5.7 representing the “Precedence”
pattern as reading backwards from right to left.

Figure 5.7: UCMPPS Representation of the Precedence Pattern.

5.2.3.3. Pattern III. Interviewees Biased Selection Tendencies

In the early stages of the interviews, the interviewees who were less experienced in the SPS kept
selecting different supportive methods to represent the different SPS patterns. The oscillations in
the selection process could be attributed to two reasons. The first reason was that the supportive
methods had different performances with regard to representation clarity depending upon the
patterns being represented. That is why both interviewee three and interviewee seven requested
integrating all the supportive methods into the Specifier tool. Another reason for the oscillation
was that the interviewees initially favored supportive methods, which seemed familiar to them
such as the FSAPPT or the RMF which used USD. In order to overcome oscillations in supportive
method selection, the interview questions and the results were revisited several times until the
oscillations were eliminated.

5.2.3.4. Interviewees Suggestions and Study Observations

The CNL ambiguities were presented to the interviewees during the introduction segment.
Interviewee three was the most experienced requirement developer in the sample. This interviewee mentioned that he was not interested in supportive methods that did not provide more
information than what the CNL offered. However, interviewee three did not select a supportive
method that addressed the CNL ambiguity issues. It should be noted that none of the state-of-the-
art supportive methods found through the literature survey were capable of addressing the CNL
ambiguity issues. The MLR, TBATLV and the MTD supportive methods were modified in an
attempt to address CNL ambiguity issues, but none of them was favored by a majority of Scania
engineers. Interviewee three also emphasized that the requirement description should be
unambiguous and correctly describes the intended behavior of the requirement developer.
According to this interviewee, a formalized requirement with an ambiguous CNL should not be
used. Furthermore, interviewee three stated that his preferred solution for the CNL ambiguities
would be to reformulate them, so they represent the correct SPS patterns temporal behavior.

Another suggestion was made by interviewee two who suggested using textual examples of
formalized requirements to illustrate the temporal behavior of the SPS patterns. These textual
examples could be used as separate supportive methods, or as additions to existing supportive
methods to enhance their clarity. This approach was already applied to this study where simple
examples were used during the interviews to explain temporal behaviors of the patterns. The
textual examples of formalized requirements provided further clarity for the interviewees and
complemented the graphical feedback. Both interviewee three and seven suggested integrating all
the supportive methods into the Specifier tool to allow the requirement developers to have more than one supportive method to choose from.

5.2.4. Case Study Account

Findings of the case study show that the supportive methods' feedback on the formalized requirements’ semantics helped in facilitating the requirement formalization process. The feedback provided help to guarantee that the formalized requirements’ semantics matched the intended behavior of the requirement developers. The two most favored supportive methods among the Scania interviewees were the RTGIL and the UCMPPS. However, both supportive methods did not address the CNL ambiguities. On the other hand, the MLR which addressed CNL ambiguity issues was chosen by the SPS experienced interviewees in the preliminary interviews. As a result, the supportive methods selected to be integrated into the Specifier tool were the RTGIL, UCMPPS and the MLR.

The results of this case study also included interviewee feedback and author’s observations that were collected during the preliminary interviews and the interviews. These observations and feedback were analyzed and a number of patterns relating to the interviewees’ experiences in using the SPS emerged from the analysis. The first pattern was with regard to differences in supportive methods' selections between the SPS experienced and the non-experienced interviewees. The second pattern described some interviewees’ issues with using a number of SPS patterns and temporal operators. The third pattern was with regard to interviewees’ biases in selecting supportive methods. Finally, the case study’s results also included a number of suggestions from the interviewees with regard to addressing CNL ambiguities, and improving the feedback process provided by the supportive methods.

Summary

This chapter presented the case study implementation procedures and findings. The case study implementation started with conducting a literature survey to collect state-of-the-art supportive methods, which were then filtered and prepared to be used as material in this study. The case study material was then used to prepare the interview protocol which consisted of an introduction segment and a questionnaire segment. Preliminary interviews were conducted to evaluate and improve the interview protocol and the case study material. The preliminary interviews were then followed by interviews conducted with a sample of engineers from Scania. The interviewees agreed that the supportive methods helped in understanding the semantics of the formalized requirements, and in facilitating the requirement formalization process. The results of the interviewees’ selections were presented in a table and in a graph format. Furthermore, results consisting of interviewee feedback and author’s observations were analyzed resulting in a number of suggestions and in identifying a number of patterns. These patterns were then discussed to explore their connections to the case study question. Finally, this chapter also presented the author’s account of the case study.
6. The Scania Specifier Tool

This chapter describes the modification and extensions introduced into the Scania Specifier tool to provide support and feedback on the SPS requirement formalization process. Information and results of the evaluation performed on these modifications and extensions are also presented.

6.1. Specifier Modifications and Extensions

To achieve the second sub-objective of this thesis, the Scania Specifier tool was modified and extended to provide tool support for a requirement formalization process. This requirement formalization process was based on Konrad and Cheng SPS qualitative and real-time patterns. In addition, three supportive methods RTGIL, UCMPPS, and MLR were integrated into the Specifier tool to provide feedback on the formalized requirements' semantics. A fourth supportive method TBATLV was also integrated to be used for providing user-guidance service for the selecting SPS patterns and scopes.

The Specifier tool was extended by adding the “Pattern Information and Views” window to save and manage patterns’ details. On the other hand, the “Edit/Add Requirement” window was modified to provide support for the SPS requirement patterning process and offer user guidance service for the users. Additionally, three supportive methods were integrated into the Specifier tool to provide feedback, which can be accessed from this window.

6.1.1. Pattern Information and Views Window

The Specifier tool has been extended with a new window called the “Pattern Information and Views” window shown in Figure 6.1. The purpose of this window is to create a pattern library, and to manage the patterns and their contents. This window’s user-interface is divided into three main panels. The first panel is in the upper left side of the user-interface, and it presents pattern information consisting of the pattern name, description and the precondition status. The second panel is in the upper right side of the user-interface. This panel lists the available views in the pattern, and provides view management functions such as edit, add and delete. The third panel is located in the lower half of the user-interface, and it contains a table that lists the existing patterns in the library. It is possible to sort and filter the contents of this table. This panel also provides functions, such as edit, add and delete to manage the patterns.

![Figure 6.1: Pattern Information and Views Window. Adding a Pattern to the Pattern List.](image)
To add a new pattern to the library, first, right click on the pattern-list in the third panel, and a menu will be displayed with the commands “Add”, and “Delete” as shown in Figure 6.1. Select “Add” from the menu, then fill in the pattern name, description and the precondition status fields in the first panel. Next, views must be added to the new pattern, and this is done by right-clicking on the second panel which displays a menu with the commands “Add”, “Delete” as shown in Figure 6.2. Once all the pattern’s information has been filled, the window’s “Apply Changes” button is enabled, and clicking it adds the pattern to the pattern-list. Clicking the “OK” button on the “Pattern Information and Views” window closes the window, and all the library information is parsed and saved into a text file located inside the Specifier tool project.

Figure 6.2: Pattern Information and Views Window. Adding Views to a New Pattern.

6.1.2. The Edit/Add Requirement Window

The “Edit/Add Requirement” window shown in Figure 6.3 originally supported adding and managing requirements written in natural language only. In this thesis, the window’s functionality was enhanced to support a requirement formalization process using the Konrad and Cheng SPS, and to provide feedback on the formalized requirements’ semantics. In addition, a signal name auto-completion and user-guidance services were also added to support the process of requirement formalization. The user-interface of this window is divided into four horizontal rows or sections. The first section is at the top and is used to configure and display the requirement’s ID, ASIL, pattern scope/format, and hierarchy information. This section also provides access to the user-guidance service for aiding users in selecting scopes and patterns. The second section contains the pattern’s proposition fields which types and numbers automatically change to match the types and numbers of proposition in the selected pattern. Figure 6.3 shows the second section containing two proposition fields, the “Precondition” and the “Postcondition” which represent the cause and effect propositions of the “Response” pattern, respectively.

The third section is the “Formalized Requirement” and it contains the formalized requirement’s CNL view. The fourth section is the “Other Views” section, and it contains the formalized requirement’s formal notations view. As the proposition fields in section two are populated, the application automatically copies the values to the relevant proposition labels in both the “Formalized Requirement” and the “Other Views” views. These labels represent the pattern condition propositions’ values, and are displayed as expressions with special formatting depending on the type of the condition. Each label expression is surrounded with curly brackets.
accompanied with a relevant symbol or letter to indicate the type of the pattern’s condition proposition. In Figure 6.4, an empty cause proposition is displayed as “{Precond}” label, while an empty effect proposition is displayed as “>{Postcond}” label. The empty Q, R, T pattern’s conditions propositions, and the C time-constraint are shown as “Q{Q_cond}”, “R{R_cond}”, “T{T_cond}”, and “C{C_cond}” labels, respectively.

Figure 6.3: Requirement Patterning using the Globally Scope and the Response Pattern with Empty Proposition Fields.

Figure 6.4 shows an example of requirement patterning using the “Real-time Bounded Response” pattern, with a “Between Q and R” scope. To create this formalized requirement, first insert the basic requirement information regarding requirement’s ID, ASIL and hierarchy information. Then, select a pattern scope from the “Pattern Scope” combo-box. Once a scope has been selected, the “Pattern Format” combo-box is populated with a list of the SPS patterns available in the patterns’ library of the Specifier tool. Second, select a pattern from the “Pattern Format” combo-box, this will cause relevant proposition fields to appear in the second section of the user-interface. The type and number of the propositions fields depend on both the scope and pattern selected. Afterwards, fill in the proposition fields with proposition logic. The proposition fields support two ways for inserting signal names, either by manually writing them, or by using the auto-completion feature for signal names. To use the auto-completion feature, the mouse cursor should be placed in a proposition field, then the symbol “#” is typed. This opens a list containing the available signal names belonging to the project loaded in the Specifier tool. Selecting a signal will insert it in the proposition field, and displays it in a green color to distinguish it from the rest of the text. It should be mentioned that the auto-completion service is only available if the Specifier tool is connected to the database. The alternative method for inserting signal names is to insert the symbol “#” and write the desired name.
The “Edit/Add Requirement” window provides a user-guidance service to aid users in selecting the SPS patterns and scopes. It is possible to access this service through clicking on the “User Guide” button located in the first section of the user-interface. The “User Guide” uses the
TBATLV supportive method to provide guidance in the form of flow-diagrams for selecting both patterns and scopes. When the “User Guide” button is clicked, a window is displayed starting in the pattern-guidance mode by default as shown in Figure 6.5. It is possible to navigate by answering the questions in the diamond-shaped figures of the flow-diagram. There are answer options available in the diagram and at the same time on the window. Clicking directly on these options will traverse the flow-diagram to the stage of the selected option. When the end of the flow-diagram has been reached, the answer options will link back to the root of the previous stage. It is possible to change the user-guidance mode to the scope-guidance mode by selecting “Scope guidance” from the drop list shown in Figure 6.5. The scope guidance mode has only one flow-diagram with no navigation possibilities as shown in Figure 6.6.

![Scope User-Guidance Diagram](image)

Figure 6.6: Scope User-Guidance.

### 6.2. Evaluation of the Specifier Tool SPS Support

To achieve the third sub-objective of the thesis, the Scania Specifier tool was evaluated at two different stages of its development during this thesis. The evaluation was concerned with the performance of the Specifier tool in facilitating a requirement formalization process using the SPS.

#### 6.2.1. The Evaluation Events

During the Scania Demo-day which was held in February 2015, students working in different departments within the Scania Research Centre exhibited their projects to Scania engineers. During this event, the visitors were shown demonstrations of a requirement formalization process using the Specifier tool. At this stage, the Specifier tool only supported a limited number of the SPS patterns, and no supportive methods were yet integrated. The demonstrations presented adding new patterns to the patterns’ library, formalizing requirements, and displaying the results of the formalization process in CNL and formal notations. The feedback from the event was encouraging, and the scenario that was used during the Demo-day can be found in Appendix E.
The final evaluation of the Specifier tool was performed after all developments were completed including the integration of supportive methods into the tool. The participants who took part in the evaluation were four fellow student engineers. Two of them had previous experience with the SPS and participated in the preliminary interviews. The evaluation sessions provided an introduction segment about the SPS and the relevant supportive methods to the non-experienced interviewees. The requirements used in the evaluation process were Scania requirements taken from Westman and Nyberg [65], and are available in Appendix F. The evaluation focused on the performance of the Specifier tool in facilitating a requirement formalization process using the SPS. The evaluation also investigated the effectiveness of the supportive methods' feedback to help in guaranteeing that the requirement formalization semantics matched the intended behavior of the requirement developers. The interviewees were asked to formalize requirements using the Specifier tool with, and without the feedback provided by the supportive methods. They were then asked the following questions:

Q1. What is your preferred supportive method?
Q2. Does the feedback provided by the supportive methods help to guarantee that the formalized requirements’ semantics match the intended behavior?
Q3. Does the user-guidance service help in the process of selecting patterns and scopes?
Q4. Does the supportive methods’ feedback help in learning the temporal behavior of the SPS patterns faster?

6.2.2. Final Evaluation Results and Discussions

A pattern that emerged from the results of the case study preliminary interviews and interviews show a relationship between having an SPS experience and addressing CNL ambiguities. Interviewees who had experience with the SPS selected supportive methods that addressed CNL ambiguities. On the other hand, most of the interviewees who did not have experience in the SPS selected supportive methods that did not address the CNL ambiguities. During the evaluation sessions, the interviewees were again asked about their preferred supportive methods to see if the same pattern still holds. Table 6.1 shows the results of the evaluation sessions. The table shows that the interviewees one and two who were experienced in the SPS did not change their previous choices during the preliminary interviews and still favored the MLR. Interviewee three was not experienced in the SPS and also selected the MLR. The interviewees one, two and three agreed that the information provided by the supportive methods’ feedback conveyed the intended behavior of the requirement developer. Interviewee four selected the UCMPPS, and also stated that UCMPPS conveyed the intended behavior of the requirement developer, despite that it did not address the CNL ambiguities. The reason was attributed to that this interviewee did not use scope and pattern combinations that exposed the formalization process to the CNL ambiguities. Evaluations results also showed that the evaluators found the user-guidance service easy to use and helpful in aiding them in the process of selecting scopes and patterns. The evaluators also stated that providing feedback on the temporal behavior of the formalized requirements made it easier to learn about the SPS.

**Table 6.1: Specifier Tool Evaluation Results.**

<table>
<thead>
<tr>
<th>Interviewee Number</th>
<th>Previous Experience in SPS</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>Yes</td>
<td>MLR (Addresses CNL ambiguities)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Two</td>
<td>Yes</td>
<td>MLR (Addresses CNL ambiguities)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Three</td>
<td>No</td>
<td>MLR (Addresses CNL ambiguities)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Four</td>
<td>No</td>
<td>UCMPPS (Does not address CNL ambiguities)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Summary

This chapter presented the modifications and extensions implemented on the Scania Specifier tool to support a requirement formalization process using the SPS qualitative and real-time patterns. The modifications also included the integration of three supportive methods into the Specifier to provide feedback on the formalized requirements’ semantics. An evaluation has been performed to assess the performance and user-friendliness of the Specifier tool in supporting an SPS requirement formalization process. In addition, the evaluation also assessed the performance of the supportive methods' feedback to help in guaranteeing that the formalized requirements’ semantics matched the intended behavior of the users.
7. Related Work

This chapter briefly presents a number of related tools that support the requirement formalizing process using the Dwyer et al. SPS. The final section in this chapter presents a comparison between the Scania Specifier and the related tools in terms of feature differences and similarities highlighting the advantages of the Specifier tool.

7.1. Related SPS Supportive Tools

This section briefly describes a number of related tools that are used to support a requirement formalization process based on Dwyer et al. SPS. However, Dwyer et al. SPS supports qualitative patterns only, and no tool was found during this study that could support Konrad and Cheng SPS.

7.1.1. Property Elucidation (PROPEL)

PROPEL was proposed by Cobleigh [11] to provide tool support for the Dwyer et al. SPS formal specification process. The final formalized requirement is represented by both the DNL and the FSAPPT interfaces. The Disciplined Natural Language (DNL), is a textual interface describing the requirement behavior in a similar manner to using natural language. The FSAPPT [54, 58], is an editable graphical interface for representing and editing the requirement temporal behavior [66]. The tool also provides the Hierarchical Question Tree (QT), which is a questionnaire that guides the users through the process of requirement formalization. A main limitation of PROPEL is that it does not support real-time patterns of the SPS. Furthermore, PROPEL captures precise details about the property behavior through its QT, however, using the QT is a lengthy process. In addition, using the FSAPPT template to edit the formalized property requires a high level of expertise.

7.1.2. Property ASSistant (PASS)

Remenska et al. [14] proposed a tool called the PASS specifically aimed for software engineers who use UML in the development process. PASS aims to facilitate the requirement formalization process by automating as much as possible of the process through using the UML design information of the system as an input. PASS provides a question tree to capture the subtle details about system property behaviors, and the final results of the patterning process are displayed to the users in four different interfaces. The first interface is a description of the formalized properties given in natural language. The second and third interfaces show a μ-calculus formula, and a USD, respectively, that represent the temporal behavior of the patterned requirement. The fourth interface is an FSA diagram which is used for runtime verification. The researchers plan to enhance the functionality of PASS so as developers can directly write sequence diagrams and get a μ-calculus formula and an FSA monitor as a result [14]. The main limitation of PASS is that it does not provide support for the SPS real-time patterns. In addition, it can only be used with systems that provide UML design information.

7.1.3. Property Specification (Prospec)

Mondragón et al [15] proposed a tool called the Prospec for facilitating requirement patterning. Prospec aims to both enhance and facilitate formal requirement specification by dividing the patterning process into well-defined and well-supported stages. Each stage provides the users with support in the form of diagrams and decision trees. Prospec offers users interactive guidance throughout the patterning process. Prospec also introduces Composite Propositions (CP) which captures concurrent or sequential propositional behavior of events and conditions enabling the expression of more complex behaviors.

The requirement patterning process in Prospec starts with specifying the scope. The user is provided with diagrams representing available scopes and their descriptions. The user can either
select the scope directly, or use the “Guided Selection” option which provides a decision tree to guide the user through the process of selecting a scope. The second stage of the patterning process involves specifying a pattern. Prospec provides users with pattern representations to aid the process. However, unlike the scope specification process, users here are not provided with a “Guided Selection” option. In the third stage of the patterning process, users get to specify the CP which are categorized into several classes represented by graphical diagrams. As in the case of specifying a scope, the user can either directly select a CP or use the help of the Guided Selection function. The fourth stage of patterning is concerned with defining behaviors of concurrent systems like, synchronization fork, synchronization join...etc. This stage does not have a “Guided Selection” option; however, it does provide three different diagrams to represent each behavior. Finally, the last stage of the patterning process presents the results of the formalization process using two views; the first view is a natural language interface. The second view is a formal logic representation which can be shown either as a FIL, or an LTL formula [15]. The main disadvantage with using Prospec is that it does not support the real-time patterns of the Konrad and Cheng SPS.

7.1.4. **CHARMY**

Inverardi et al. [67] proposed a tool called CHARMY, which is a UML-based tool that can be used with industrial UML projects. CHARMY provides tool support to facilitate FV without the need for expertise in formal notations. It uses a model checking method to validate software architecture correctness against system properties. The tool is designed to accept plugins allowing the addition of new features, and integration with other tools [67]. CHARMY specifies the software architecture as state diagrams, which are then translated into a verification modelling language called Protocol Meta Language (Promela) [68]. CHARMY uses USD to represent system properties, which are then translated into Buchi Automata [69]. The software architectural model produced in Promela code can then be validated against the formally specified properties using the Simple Promela Interpreter (SPIN) model checker, which is a tool used for FV of distributed system [70, 71].

The CHARMY tool consists of three main parts; the first part is a graphical editor for specifying software architecture. The second part is a translator utility to interpret state diagrams into Promela code. The third part is a repository component to store models for the purpose of reuse. There are three sub-components in the graphical editor, the first is the “Topology Editor” which is used to specify software architectural topology as components, connectors and interactions. The second sub-component is the “Thread Editor” which is used to specify the architectural components’ internal behavior into state machines. Afterwards, a translator utility automatically transforms the state diagrams of the “Thread Editor” into Promela code. The third sub-component is the “Sequence Editor” which is used to model temporal properties representing component interactions and are represented in the Property Sequence Charts (PSC). PSC is a scenario based language extended from the USD. PSC was presented by M. Autili et al. [16] and used for expressing temporal properties. The PSC generated from the “Sequence Editor” is then translated into a Buchi Automata [69] using the translation algorithm PSC2BA. Finally, the SPIN analyses the information in the Buchi Automata against that of the Promela code to validate the temporal properties [67]. The main limitations of CHARMY tool are that it does not provide support for the SPS real-time patterns, and it can only be used with systems that provide UML design information.

7.2. **SPS Supportive Tools Comparison**

This section briefly compares some of the similarities and differences between the Scania Specifier tool, and the other related tools mentioned in this chapter. The purpose of the comparison shown in Table 7.1 is to highlight some of the features of the Specifier which give it
advantage over the other mentioned related tools. One of the most important features in this comparison is with regard to the type of the SPS patterns supported by the different tools. The main advantage of the Specifier tool is that it provides requirement formalization support using both the qualitative and the real-time SPS patterns. Meanwhile, all the other mentioned tools support the qualitative SPS patterns only.

Another feature is the type of views or feedback provided to the user on the temporal behavior of the formalized requirements. In general, there are three types of views for representing or giving feedback, and they are textual, formal notation, and graphical. The tools mentioned in this chapter represent the formalized requirements using different combinations of these views. With respect to the textual view, the Specifier tool uses CNL, which was proposed by Konrad and Cheng to describe the formalized requirement. PROPEL’s textual view uses DNL, which is mapped to the FSAPPT representation. On the other hand, PASS and Prospec both use natural language to describe the elicited properties, while CHARMY does not provide any textual view. With regard to the formal notation view, the Specifier has the advantage of being able to configure the number, type and viewing priority of the formal notations of each pattern. This gives flexibility to the requirement developers to choose different formal notations for representing different patterns. In addition, the Specifier tool supports an SPS formalization process. This means that the different conditions’ propositions are formatted and displayed in the formal notations’ formulae in condition specific formats to enhance clarity. The PASS tool utilizes a view which displays a μ-calculus formula, and Propsec’s view can show either an FIL or an LTL formula. On the other hand, PROPEL and CHARMY use FSAPPT, and Buchi Automata, respectively to represent formalized requirements’ temporal behaviors. Regarding the graphical view, the Specifier tool has the advantage again over the rest of the mentioned related tools, as there is no limit to the number of supportive methods that can be integrated into it. Currently, the Specifier tool provides the RTGIL, UCMPPS and MLR supportive methods as graphical feedback on the formalized requirements' semantics. PROPEL and CHARMY, respectively, provide FSAPPT and FSA representations, which require having a level of expertise on behalf of the users. Remenska et al. considered USD to be more comprehensible by users than the FSA, and therefore, PASS provides representations in both FSA and USD [14]. Finally, Prospec does not provide any graphical view.

With respect to the user-guidance feature, the TBATLV supportive method has been integrated into the Specifier tool to provide user-guidance in selecting the SPS scopes and patterns. TBATLV consists of a flow-diagram, and this visual approach in guiding is more user-friendly than textual-based user-guidance alternatives. PROPEL and PASS provide questionnaire trees for user-guidance in selecting scope and pattern behaviors. However, these questionnaires are textual and take longer time to complete. Prospec provides a decision tree to guide users in selecting scopes only, while it provides diagrams to aid in pattern selection. Finally, CHARMY does not provide any user-guidance.

Operational pre-requisites for the requirement formalization process is a feature that can limit on the usability of the tools. The Specifier tool requires connection to a database, and the advantage is that it enables the signal auto-completion feature which aids users in entering signal names to construct pattern conditions' propositions. The Prospec and the PROPEL tools do not have any pre-requisites either. On the other hand, both CHARMY and the PASS tools require system UML 2.0 design information as a pre-requisite, and they cannot be used with projects that do not have this information.
<table>
<thead>
<tr>
<th>SPS Pattern Support Type</th>
<th>Specifier</th>
<th>PROPEL</th>
<th>PASS</th>
<th>Prospec</th>
<th>CHARMY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formalized Requirement Views</td>
<td>Textual</td>
<td>Formal Notation</td>
<td>Graphical</td>
<td>RTGIL</td>
<td>UCMPSS MLR</td>
</tr>
<tr>
<td></td>
<td>CNL</td>
<td>DNL</td>
<td>FSAPPT</td>
<td>FIL/LTL</td>
<td>μ-calculus Formula</td>
</tr>
<tr>
<td></td>
<td>Qualitative</td>
<td>Qualitative</td>
<td>Qualitative</td>
<td>Natural Language</td>
<td>Natural Language</td>
</tr>
<tr>
<td>SPS Tool Comparison Table</td>
<td>Operational Pre-requisites</td>
<td>Requires connection to database</td>
<td>Questionnaire Tree</td>
<td>UML 2.0 Design Information</td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>Requires connection to database</td>
<td>Questionnaire Tree</td>
<td>FSA and USD</td>
<td>None</td>
<td>FSA</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

This section presented a number of features that give the Specifier tool several advantages over other related tools mentioned in this chapter. Among the main advantages of the Specifier is that it supports a requirement formalization process using both the qualitative and the real-time SPS patterns. The tool has a view dedicated for providing graphical feedback to help guarantee that the behavior of the formalized requirements’ semantics match the intended behavior of the requirement developer. The Specifier tool also provides views for displaying CNL, and formal notations representing the formalized requirement semantics. Theoretically, there is no limit on the number of formal notations and feedback supportive methods that could be integrated into the
Specifier tool. In addition, the tool is user-friendly providing a signal name auto-completion service, and user-guidance to aid in the patterning process.

**Summary**

This chapter presented a number of related tools available in stat-of-the-art literature that provided tool support for the requirement formalization process based on Dwyer et al. SPS only. The final section of this chapter highlighted the advantages of the Specifier tool by comparing its features to that of the related tools.
8. Conclusions and Future Work

8.1. Conclusions

The ISO 26262 safety standard has been introduced to address safety concerns regarding the automotive industry E/E systems, which are increasingly becoming software dependent. Scania is an industrial partner in the VeriSpec project which studies the obstacles hindering the adoption of ISO 26262 in the automotive industry, and proposes academic and industrial relevant solutions. This thesis main objective addressed one of the goals of the VeriSpec project, which is facilitating and supporting a requirement formalization process based on a patterning method.

The main objective of this thesis has been broken down into three sub-objectives. To achieve the first sub-objective, a qualitative case study with a literature survey was performed to respectively, find and evaluate state-of-the-art supportive methods. In the literature survey, state-of-the-art supportive methods were found, and criteria were applied to filter the results. None of these state-of-the-art supportive methods addressed the SPS CNL ambiguity issues discussed by Filipovikj. Additionally, only two supportive methods, the RTGIL and the UCMPPS met the rest of the criteria conditions. As a result, exemptions had to be made in order to include more supportive methods in the study. The final outcome of the literature survey and the filtering process resulted in six supportive methods. One more supportive method the MTD was later made from combining two supportive methods and added based upon preliminary interviews’ feedback. Some of the supportive methods had to be prepared in order to include them in the study. These preparations involved three supportive methods the MLR, MTD, and TBATLV, which were also enhanced to represent the CNL ambiguities using textual and graphical approaches. Furthermore, SPS qualitative and real-time patterns’ representations were made for these supportive methods. The purpose of including the modified supportive methods in the study was to provide means to illustrate CNL ambiguities to the study participants and to get their reactions on these attempts.

After preparing the supportive methods, a qualitative research methodology was defined. The purpose of the study was to evaluate the feedback of the supportive methods with regard to helping in guaranteeing that the requirement formalization semantics match the intention of the user. The research methodology was implemented as a case study approach to explore opinions and collect feedback regarding the research question. The case study category was thus exploratory, and its type defined as a single case holistic instrumental case study. The case study consisted of several procedures, a data collection, sample selection, data analysis procedure, and data validation and evaluation procedure. The data collection procedure followed the Creswell data collection cycle and was implemented during the preliminary interviews and the interviews. This data collection procedure was open-ended and relied on observations, feedback and the interview protocol questionnaire as sources for data. The observations were taken from the point of view of an observer and were recorded as hand-written notes. A participant sample for the case study interviews was provided by the Scania management. The sample consisted of seven engineers from different departments with different roles, hence it fulfilled the maximum variation sampling approach. The data analysis procedure used the Creswell spiral process to analyze and interpret the collected data. In the last stage of the Creswell spiral process, the direct interpretation analysis and categorical aggregation techniques were applied to obtain the conclusions. The final procedure of the case study was the study’s validation and evaluation procedure, which applied the triangulation approach, and criteria proposed by Creswell, respectively.

After defining the research methodology, an interview protocol was designed and consisted of an introduction segment and a questionnaire segment. Preliminary interviews were then conducted to evaluate and improve the case study material and the interview protocol. The participants were two fellow engineering students and a Scania engineer. As a result of the preliminary interviews,
the MTD was included in the case study. Seven Scania engineers were then suggested by the Scania management to participate in the case study interviews. Data was collected during the interviews as written notes, and consisted of the interviewees’ feedback, observations, and the questionnaire’s results.

The results of the case study included the patterns that emerged from analyzing feedback and observations, the selected supportive methods, and the interviewees’ suggestions. There were three patterns that emerged from the case study, the first pattern was with regard to the role of having SPS experience on the supportive methods' selection. During the interviews, the interviewees were informed about CNL ambiguity issues, and they were shown different examples illustrating them. Table 5.3 and Table 5.4 show that all the interviewees who were non-experienced in the SPS chose supportive methods, which did not address the CNL ambiguity issues except for interviewee six. Meanwhile, Table 5.3 shows that the two experienced SPS interviewees in the preliminary interviews selected the MLR which addressed CNL ambiguities. This leads to the conclusion that the lack of experience in the SPS influences the type of the supportive methods selected, and could be responsible in not addressing the CNL ambiguity issues.

The second pattern was with regard to difficulties in understanding some SPS patterns and temporal operators. The Scania interviewees were not experienced in the SPS, and some of them had difficulties in using the “Precedence” pattern and the temporal operator “Eventually”. Three Scania engineers were not comfortable using the “Precedence” pattern and experienced using it as reading backwards. The difficulty in using the temporal operator “Eventually” could be attributed to two reasons. The first reason was that the concept of the temporal operator “Eventually” was experienced as confusing to the embedded engineers who were used to working with scheduled events. The second reason was requirement ambiguity, where information such as time constraints was not mentioned in the requirement text, leading to using qualitative patterns to formalize real-time requirements. The mentioned difficulties can be addressed by improving knowledge in the SPS, and using unambiguous requirements, respectively.

The final pattern that emerged from the case study was with regard to the Scania interviewees biased selections. At the beginning of the interviews, it was noticed that some Scania interviewees who were not experienced in the SPS constantly changed their selected supportive methods. One reason for these oscillations was that the supportive methods' feedback varied in clarity depending on the patterns being represented. This explains why some Scania interviewees suggested integrating all the supportive methods into the Specifier tool. Another reason for the oscillations was that the interviewees initially favored supportive methods that seemed familiar to them, such as the FSAPPT or the RMF. To overcome these oscillations, the interview questions and the results were revisited several times to get the interviewees more familiar with the features of different supportive methods. As a result of these repetitions, the interviewees were able to provide an evaluation based upon the overall performance of the supportive methods rather than on separate incidents.

Results of the interview questionnaire show that the RTGIL and the UCMPPS were the supportive methods mostly favored by the Scania interviewees who were not experienced in the SPS. Both the RTGIL and the UCMPPS did not address the CNL ambiguity issues which were explained in detail during the interviews. However, the data from the preliminary interviews showed that the SPS experienced interviewees preferred the MLR supportive method which addressed CNL ambiguities. As a result, the methods that were selected to be integrated into Scania Specifier tool were the RTGIL, UCMPPS and the MLR as supportive methods, and the TBATLV was used for user-guidance.
8.2. Future Work

Following are some suggestions made by the author and the Scania interviewees who took part in the case study. These suggestions are generally with regard to improving the feedback of the supportive methods, and identifying issues that can be further researched.

The results of the literature survey conducted in this thesis did not yield supportive methods that addressed CNL ambiguities. One suggestion is to find or develop state-of-the-art supportive methods that can address CNL ambiguities. A second suggestion is to provide additional feedback options for the requirement developers by integrating more supportive methods into the Specifier tool. A third suggestion is to use textual examples of formalized requirements as feedback. These textual examples can either complement existing supportive methods, or be used as separate supportive methods. A fourth suggestion is to have unambiguous formalized requirements. According to this suggestion, research is needed to re-formulate ambiguous CNL so that it could correctly describe the temporal behavior of the SPS patterns.
References


Appendices
## A. Modified LR SPS Pattern Semantics Representations

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>P must always be followed by S</td>
</tr>
</tbody>
</table>

The effect S can be stimulated by other different causes and not only By P.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Response Chain 1-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>S,T responds to P: 1 stimulus produces 2 responses.</td>
</tr>
</tbody>
</table>

The effect S can be stimulated by other different causes and not only By P.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Constrained Chain Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>S,T without Z responds to P: 1 stimulus produces 2 responses while negating a third.</td>
</tr>
</tbody>
</table>

The effect S can be stimulated by different causes and not only By P.

---

Figure A.1: MLR Representations of Different Patterns with Response Semantics. These Figures show a Textual Approach to Address CNL Ambiguities.
The cause S can stimulate and precede more than one effect at the same time.

Figure A.2: MLR Representations of Different SPS Patterns with Precedence Semantics. These Figures show a Textual Approach to Address CNL Ambiguities.
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absence</td>
<td>P does not hold at any state.</td>
</tr>
<tr>
<td>Existence</td>
<td>P will become true at least once in the future.</td>
</tr>
<tr>
<td>Bounded Existence</td>
<td>P is allowed to hold at most twice in the future.</td>
</tr>
<tr>
<td>Universality</td>
<td>P holds in all states.</td>
</tr>
</tbody>
</table>

Figure A.3: Occurrence Pattern Semantics Representation in MLR.
Pattern | Real-Time Order-Bounded Invariance
---|---
Behavior | Specifies the minimum amount of time a state formula must hold once another state formula is satisfied.

Pattern | Real-Time Order-Bounded Response
---|---
Behavior | Restricts the maximum amount of time that passes after a state formula holds until another state formula becomes true.

Pattern | Duration-RT Max Duration
---|---
Behavior | Denotes the maximum amount of time a state formula has to hold once it becomes true.

Pattern | Duration-RT Min Duration
---|---
Behavior | Denotes the minimum amount of time a state formula has to hold once it becomes true.

Pattern | Periodic-Bounded Recurrence
---|---
Behavior | Denotes the amount of time a state formula has to hold periodically.

Figure A.4: Real-Time Pattern Semantics Representation in MLR.
B. Template Based Diagram SPS Pattern Semantics Representations

Figure B.1: TBATLV Diagram of the Main Branch in the Flow-Diagram.
Figure B.2: TBATLV Diagram of the Real-Time Patterns Branch.
Is it a cause-effect relation or just an effect?

Only an effect relation

Cause-Effect relation

Occurrence Patterns

Order Patterns

Should the effect proposition become true?

No never

Yes

Absence

Universality

How often does it become true?

Always

Eventually

Twice at most

Bounded Existence

Existence

Figure B.3: TBATLV Diagram of the Qualitative Patterns Branch.
Order Patterns

Is the effect stimulated by a specific cause which can stimulate one or more effects?

Does the effect or cause consist of a chain of events?

Precedence Pattern

What type of a chain?

Response Pattern

Response Chain

1 stimulus - 1 response

2-1

Constrained

Precedence Chain

2 causes - 1 effect

Response Chain

1-2

Constrained Patterns response chain, 1 stimulus and a simple proposition negator

Figure B.4: TBATLV Diagram of the Order Pattern Branch.
Figure B.5: TBATLV Diagram of the Pattern Scopes.
C. Modified Timing Diagram SPS Pattern Semantics Representations

Figure C.1: MTD Representation of the Precedence Pattern - The Cause S Precedes and Stimulates the Effects V, T and P. This Figure shows A Graphical Approach to Address CNL Ambiguities.
Figure C.2: MTD Representation of the Response Pattern with “After Q” Scope - The Pattern Always Holds After the First Occurrence of Q but Not Before it. This Figure shows A Graphical Approach to Address CNL Ambiguities.
Figure C.3: MTD Representation of the Response Pattern with the “Before R” Scope - The Pattern Always Holds Before the First Occurrence of R but Never After it. This Figure shows A Graphical Approach to Address CNL Ambiguities.
D. Case Study Questionnaire

D.1. Introduction

In the introduction segment, the interviewer will provide the interviewees with information on the SPS, temporal operators, and the different supportive methods. Material used in the introductory segment consists of both textual and graphical information and examples, however, this material is not included in this document due to its size.

D.2. Assignments

The goal of the assignments is to evaluate the feedback provided by the supportive methods to help in guaranteeing that the formalized requirements’ semantics match the intention of the user.

D.2.1. Formalizing Requirements

Instructions:
1- Selecting a pattern
   a. There are 3 pattern categories (RT, timed or quantitative, non-timed or qualitative). In order to select a pattern group, first read the requirement and understand its behavior.
   b. Check if the requirement is a timed (quantitative) or a non-timed (qualitative) requirement.
   c. Check if the requirement is a cause-effect or just an effect requirement. At this point, you can select a pattern category.
   d. Answer the questions in the flow-diagram to select a pattern.

2- Selecting a Scope:
   a. Read the requirement and use the scope flow-diagram to choose a scope that describes in which states during program execution the requirement holds.
   b. Be aware that some scopes alter the requirement behavior. Check the scope diagram for any remarks on the selected scope that can affect where in the program the requirement can apply.

3- Supportive methods’ feedback:
   a. Find a feedback diagram that represents the intended behavior of the formalized requirement and check if it matches your scope and pattern selections.

4- Give an assessment with regard to the supportive methods’ feedback.

Exercises:
For each of the following exercises, select an option from the list below.
1. “If P holds, then output S shall hold”.
2. “When S or T holds, the status of input signal P shall be set to Error”.
3. “If P holds, then, S has previously held”.
4. “P shall hold within 2 seconds from when the ECU_Start = true holds.
5. “P shall hold for at least 2 seconds from when the ECU_Start = true holds.

Options:
(a) Globally-Response
(b) Globally-Response
(c) Globally-Precedence
D.2.2. Select the Pattern that Conveys the Correct Requirement Behavior

Instructions:
1.- Read the requirement text in the exercise before formalization and then read the formalization examples.

2.- Analyze the pattern of the formalized requirement:
   a. Check if it is a qualitative or a real-timed.
   b. Check if it is a cause-effect or just an effect pattern.
   c. Use the of TBATLV flow-diagram to help in selecting a pattern. Check for any comments on the pattern temporal behavior.

3.- Analyze the scope of formalized requirement:
   a. From the formalized text, find out what scope is used.
   b. Check the scope on the TBATLV scope diagram for any remarks that can affect the behavior of the formalized requirement.

4.- Supportive method's feedback:
   a. Find a feedback diagram that represents the intended behavior of the formalized requirement and check if it matches your scope and pattern selections.

5.- Select
6.- Give an assessment with regard to the supportive methods' feedback.

Exercise:
Requirement: “If P=’Error’ or P=’NotActive’ then set P=’False’”.

Formalization options:
   a. “After ((P = ‘Error’) or (P = ‘NotActive’)), it is always the case that (P = ‘False’) holds”.

   b. “Globally, it is always the case that if ((P = ‘Error’) or (P = ‘NotActive’)) holds, then (P = ‘False’) eventually holds”.

D.3. Questions
1. Is the visualization helpful for understanding the behavior of the patterns and matching them to the requirements?
2. Which supportive method do you find most helpful?
3. Do you have any suggestions to improve the feedback process?
E. Demo-Day Scenario

During Scania Demo-day, interested visitors in the event have been shown demonstrations in requirement formalization using the Specifier tool. The purpose of this scenario is to demonstrate the modifications introduced into the Specifier tool so as to provide support for requirement formalization using Konrad and Cheng SPS. The requirement used in the demonstration is not included in this scenario as they are Scania requirements.

Scenario:
1. Click on the Pattern Admin button in the main window of the application. The “Pattern Information and Views” window is displayed.
2. Create a new pattern and check that it is added to the pattern list.
3. Exit the “Pattern Information and Views” window.
4. Insert new requirement in the document, the “Edit/Add Requirement” window appears.
5. Enter the Id/ASIL/hierarchy of the requirement.
6. From the “Select Pattern” combo-box select a pattern.
7. Relevant condition fields will appear with their related empty labels, such as ({}>) for Precondition, (>{}]) for Postcondition, and (C{}) for time constraint.
8. Fill in the condition fields. The proposition condition field such as pre and post conditions shall accept Boolean expressions using port names. For the real-time requirement enter the time constraint value.
9. The “Formalized requirement view” will show the statement of the selected pattern in CNL with its conditions labeled and filled.
10. The “Other views” view will show the pattern’s temporal behavior described in formal notations with labeled and filled conditions.
11. Save the requirement.
12. On saving the requirement, the “Edit/Add Requirement” window is closed and the requirement will be visible on the Specifier main window.
**F. Scania Requirements used for Evaluating the Specifier**

Table F.1: Scania Requirements used for Specifier Tool Evaluation. These Requirement are taken from Westman and Nyberg [65].

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR_COO</td>
<td>IF <code>actualParkingBrake</code>[Bool] is false, THEN <code>indicatedFuelVolume</code>[%], shown by the fuel gauge, is less than or equal to <code>actualFuelVolume</code>[%].</td>
</tr>
<tr>
<td>FSR_Driver</td>
<td>IF <code>actualParkingBrake</code>[Bool] is false, THEN the derivative of <code>actualFuelVolume</code>[%] is less than or equal to 0.</td>
</tr>
<tr>
<td>TSR_RCAN</td>
<td>IF CAN1 is equal to a CAN message <code>rmsg</code>, THEN <code>rmsg</code> is in FiFoBuffer within 20ms.</td>
</tr>
<tr>
<td>SSR_BUFF</td>
<td>When <code>Buff_get_B</code> transitions from true to false, <code>rmsg</code> is the oldest message in FiFoBuffer.</td>
</tr>
<tr>
<td>HWSR_HW</td>
<td>IF CAN1 is equal to a CAN message <code>rmsg</code>, THEN <code>rmsg</code> is in HWreceiveBuffer AND Rcan decodeCan is set to true.</td>
</tr>
</tbody>
</table>