A comprehensive process for Automotive Model-Based Control

A Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Sabarish Gurusubramanian, B.E.

Graduate Program in Department of Mechanical Engineering

The Ohio State University

2013

Master’s Examination Committee:

Dr. Shawn Midlam-Mohler, Advisor
  Dr. Giorgio Rizzoni
  Dr. Marcello Canova
  Dr. Fabio Chiara
Abstract

The global automotive industry continually strives to develop advanced automotive control systems that can reduce fuel consumption and emissions towards a greener future, while improving efficiencies and enhancing safety and comfort. This effort is collaborative, with not just OEMs and their suppliers working amidst stiff competition and strict regulations, but various research organizations, both independent as well as academic collaborating with these OEMs. As a result of this collaborative effort, there is a huge potential for synergy, but also the possibility of huge disconnects in the process itself.

Model-Based System Development methodologies that use simulation models representing the controlled and controlling systems have a very important role to play in today’s scenario when developing control systems. For a research organization with strong academic background such as the Center for Automotive Research at The Ohio State University, a comprehensive process for automotive model-based control that encapsulates various practice and standards already in place in the industry can help develop better solutions when collaborating with industry. While systematic and detailed approaches exist already, there are sufficient variations amongst internal approaches and methodologies which calls for a more unified approach that is industry-inspired. This thesis presents a comprehensive process that helps a developer gain an overall perspective to the bigger problem. The proposed methodology
starts right at the fundamental opportunity identification phase, and is driven in the
early stages by product development methodologies.

Systematic approaches towards identifying opportunities, generating concepts and
selecting concepts, with a case-study based on the usage of model-based simulations
tools to select concepts are presented. With constantly changing requirements in the
automotive industry, the need for traceability to the initial requirements has been
highlighted. To serve as an example throughout this thesis, a specific engineering
problem aimed at optimizing the electrical system is chosen from an ongoing project
on fuel economy improvement with Chrysler LLC. The objective of improving fuel
economy by optimizing the usage of the alternator, and making better use of the
existing vehicle battery to meet electrical load demands in the vehicle is achieved
by means of an Adaptive Equivalent Consumption Minimization Strategy (A-ECMS)
based controller. The A-ECMS controller thus designed is implemented on a vehicle
test-setup using Rapid Control Prototyping (RCP) tools. Verification and Validation
techniques that ensure that the system is built to specifications and meets require-
ments identified are presented in detail with the same example. Finally, future rec-
ommendations are made on implementation and testing as well as other model-based
tools and approaches that can be considered.

An overall process that gives a developer the bigger picture that encompasses the
control system development process has been attempted in this thesis, drawing inspi-
ration from various product development methodologies as well as industry-standard
practices. Staying updated with relevant standards and methodologies can promote
better industry collaborations and provide greater learning, and this thesis aims to
support that with the proposed methodology for automotive model-based control.
This thesis is dedicated to my parents, friends and the Center for Automotive Research, The Ohio State University
Acknowledgments

The Center for Automotive Research has truly been home for me here in the United States, and it has been one amazing experience to work here towards my Masters Degree. First of all, I would like to thank my advisor, Dr. Shawn Midlam-Mohler for everything that has helped me get where I am today. Introducing me to CAR and its research, letting me work with the EcoCAR2 team at different levels and more importantly, helping me organize my thoughts and shape them towards this thesis and get it on paper - I owe it all to him.

I also take this opportunity to thank Dr. Fabio Chiara, who has been more of a brother than just a mentor to me. But for his support and motivation, both academically as well as personally, I would have never made it this far. It has been a tremendous learning experience working with him and his unparalleled commitment and passion towards work continue to inspire me.

I would like to thank Dr. Giorgio Rizzoni for believing in me and giving me an opportunity to work at CAR on the Natural Gas Consortium Project. The learning and experience from that project is just one amongst so many other things I’ve gained from my experience at CAR.

Majority of my work in this thesis revolves around my experience with the Chrysler Project and I owe it to Dr. Marcello Canova for the opportunity, his constant motivation and support. Dr. Lisa Fiorentini has been an amazing person to work with,
and has always been there to guide and support me throughout the implementation phase of this project. I would like to thank her for all the time she made for me even when she was over-loaded. The support of Jim Shivley, Electronics Specialist at CAR has been great and all his help with instrumenting the vehicle and the hardware development for implementation deserves special mention. I would also like to thank Chrysler LLC for the opportunity and the people there who have helped me throughout this project.

I would also like to thank Dr. James Durand for sharing his expertise and knowledge during times when we were working on the Natural Gas Project. Finally, I would like to thank all other colleagues and staff at CAR who have made my experience here unforgettable.

The EcoCAR2 team has been a great source of inspiration and I consider it a privilege to have been a part of it. I would like to thank the entire EcoCAR2 team, especially Matt Yard, Katherine Bovee and Andrew Garcia for the various interactions we’ve had the past two years.

I take this opportunity to thank Nithin Venugopal Baradwaj for being there all-throughout - we both have come a long way together here at CAR. Finally, there are so many other colleagues and friends here at CAR who have made the experience of almost two years here completely worth it - this includes Kingsly Jebakumar, Neeraj Agarwal, Balaji Sampathnarayan, Kyle Merical, Jeremy Couch, Quansheng Zhang, Shreyas Shivaprasad, Luke Debruin and Nathaniel Huntsberger.
Vita

November 11, 1987  ................. Born - Salem, India

2005  ...................... S.B.O.A Mat Hr Sec School

2009  ...................... B.E. Production Engineering,
              PSG College of Technology, Coimbatore, India

2009-2011  ...................... Dy.Manager-International Operations,
              Ashok Leyland Ltd. Chennai, India.

2011-present  ...................... Graduate Research Associate,
              Center for Automotive Research,
              The Ohio State University.

Publications

Research Publications

S. Amjad, R. Rudramoorthy, S. Neelakrishnan, S. Gurusubramanian et.al


S. Midlam-Mohler, G. Rizzoni, K. Bovee, S.Gurusubramanian, et.al


Fields of Study

Major Field: Mechanical Engineering
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Research Objective</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Scope</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Document Layout</td>
<td>3</td>
</tr>
<tr>
<td>2. State of the Art</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Product Development Processes</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Model-Based Control</td>
<td>7</td>
</tr>
<tr>
<td>2.3.1 System Development</td>
<td>7</td>
</tr>
<tr>
<td>2.3.2 Verification and Validation</td>
<td>8</td>
</tr>
<tr>
<td>2.4 Literature Review of Model-Based Development Techniques</td>
<td>15</td>
</tr>
<tr>
<td>3. Model-Based Control: Conceptualization Phase</td>
<td>32</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>32</td>
</tr>
<tr>
<td>3.2 Opportunity Identification Process</td>
<td>33</td>
</tr>
<tr>
<td>3.2.1 Opportunity listing</td>
<td>35</td>
</tr>
<tr>
<td>3.2.2 Opportunity Screening and Evaluation</td>
<td>38</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.5.1 Virtual Prototyping</td>
<td>88</td>
</tr>
<tr>
<td>5.5.2 Controller Calibration</td>
<td>90</td>
</tr>
<tr>
<td>5.5.3 Rapid Control Prototyping</td>
<td>91</td>
</tr>
<tr>
<td>5.5.4 Rapid Prototyping - Results</td>
<td>94</td>
</tr>
<tr>
<td>5.6 Hardware-in-the-Loop</td>
<td>100</td>
</tr>
<tr>
<td>5.7 Final System Integration and Release</td>
<td>102</td>
</tr>
<tr>
<td>6. Case Study - Rapid Concept Vehicle Architecture Selection using Model-Based Simulation Tools</td>
<td>103</td>
</tr>
<tr>
<td>6.1 Introduction</td>
<td>103</td>
</tr>
<tr>
<td>6.1.1 The EcoCAR2 Competition</td>
<td>103</td>
</tr>
<tr>
<td>6.2 Concept vehicle architecture selection process</td>
<td>105</td>
</tr>
<tr>
<td>6.2.1 Architecture Selection</td>
<td>105</td>
</tr>
<tr>
<td>6.2.2 Autonomic</td>
<td>107</td>
</tr>
<tr>
<td>6.2.3 Modeling Assumptions</td>
<td>109</td>
</tr>
<tr>
<td>6.2.4 Baseline vehicle architecture</td>
<td>109</td>
</tr>
<tr>
<td>6.2.5 Candidate concept vehicle architectures</td>
<td>111</td>
</tr>
<tr>
<td>6.2.6 Powertrain Simulation Approach</td>
<td>115</td>
</tr>
<tr>
<td>6.2.7 Top-Level, Energy-Based Performance Calculations</td>
<td>117</td>
</tr>
<tr>
<td>6.2.8 Candidate architectures modeling using Autonomic</td>
<td>118</td>
</tr>
<tr>
<td>6.2.9 Simulated Energy and Performance Results</td>
<td>119</td>
</tr>
<tr>
<td>6.2.10 Performance of the selected architecture</td>
<td>121</td>
</tr>
<tr>
<td>6.2.11 Impact of Component Sizing</td>
<td>121</td>
</tr>
<tr>
<td>6.2.12 Limitations of using Autonomic</td>
<td>124</td>
</tr>
<tr>
<td>6.2.13 Conclusions</td>
<td>127</td>
</tr>
<tr>
<td>7. Conclusions and Future Work</td>
<td>128</td>
</tr>
<tr>
<td>7.1 Conclusions</td>
<td>128</td>
</tr>
<tr>
<td>7.2 Future Work</td>
<td>129</td>
</tr>
<tr>
<td>7.2.1 Model-based approaches</td>
<td>129</td>
</tr>
<tr>
<td>7.2.2 Electrical system optimization problem</td>
<td>132</td>
</tr>
</tbody>
</table>

Appendices

| A. Autonomic file tables                                               | 135  |
| B. Rapid Control Prototyping using INTECRIO                           | 141  |
| B.1 Simulink Model Configuration                                       | 141  |
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The generic product development process</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Process flow diagrams for different product development processes</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Requirements and Specifications - Validation and Verification</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Model-in-the-Loop Simulation</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Software-in-the-Loop Simulation</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>Processor-in-the-Loop Simulation</td>
<td>12</td>
</tr>
<tr>
<td>2.7</td>
<td>Offline Simulation, Real-time Simulation and Simulation Overrun</td>
<td>13</td>
</tr>
<tr>
<td>2.8</td>
<td>Hardware-in-the-Loop Simulation</td>
<td>14</td>
</tr>
<tr>
<td>2.9</td>
<td>Hardware-in-the-Loop Simulator capabilities</td>
<td>15</td>
</tr>
<tr>
<td>2.10</td>
<td>Traditional vs Model-Based Development for ECUs</td>
<td>17</td>
</tr>
<tr>
<td>2.11</td>
<td>Model-Based Software Development process for ECUs</td>
<td>17</td>
</tr>
<tr>
<td>2.12</td>
<td>Feature Model Development - Functional specification</td>
<td>18</td>
</tr>
<tr>
<td>2.13</td>
<td>Seamless Knowledge Oriented Model-Based Product Development for Embedded Systems</td>
<td>19</td>
</tr>
<tr>
<td>2.14</td>
<td>Dimensions of reuse - Model-Based Product Development for Embedded Systems</td>
<td>20</td>
</tr>
</tbody>
</table>
4.1 Model-Based Control System Design approach .......................... 59
4.2 System Identification Process ................................................. 63
4.3 Vehicle Electrical System with existing control .......................... 66
4.4 Alternator Modeling - Black Box model ................................. 68
4.5 Battery Modeling - Grey Box model .......................... 69
4.6 Parameters influencing the Control Logic Selection Process ............ 72
4.7 Conventional Vehicle Electrical System and Parallel HEV Architectures 74
4.8 A-ECMS Control System Design methodology ......................... 76
4.9 A-ECMS Control System Simulink Model - overall view .......... 77
4.10 A-ECMS Control System Simulink Model - subsystem view .... 77
4.11 Requirements Traceability - Spreadsheet and Simulink Model ...... 81
5.1 A-ECMS MiL results: 5 consecutive FTP cycles with medium electrical loads ......................................................... 84
5.2 A-ECMS - Sample model coverage report .................................. 86
5.3 Software Code Testing Approaches ......................................... 87
5.4 SiL vs MiL - ECMS Results comparison ................................... 88
5.5 ETK in parallel with ECU - Measurement and Calibration ........... 91
5.6 OSU Rapid Control Prototyping Setup for Alternator Control ...... 92
5.7 A-ECMS Standalone controller for Rapid Prototyping ................ 92
5.8 A-ECMS Standalone controller setup procedure ..................... 93
5.9 INTECRIO Integration Platform for the A-ECMS Implementation ... 94
5.10 INTECRIO Experiment Environment for the A-ECMS Implementation | 95
5.11 Implemented A-ECMS controller in comparison to MiL | 97
5.12 A-ECMS Testing: Charge sustaining performance over 1 FTP Cycle | 98
5.13 A-ECMS Testing: Test plan | 100
5.14 A-ECMS Testing: Electrical System Loading Schedule | 101
6.1 OSU EcoCAR2 vehicle architecture selection process | 105
6.2 Autonomie - simulation process | 108
6.3 Flow of Power through the Autonomie Chevy Malibu Model | 110
6.4 Series E-REV | 112
6.5 Pre Transmission Parallel E-REV | 113
6.6 Post Transmission Parallel E-REV | 114
6.7 Pre+Post Transmission Parallel E-REV | 115
6.8 Candidate architectures - simulation process approach | 116
6.9 Fuel Economy and Performance Simulation Results of candidate architectures | 120
6.10 Simulation performance results of Pre+Post Transmission parallel E-REV architecture | 122
6.11 Impact of Electric Motor Size on Pre-Transmission Hybrid Vehicle Fuel Economy | 124
6.12 Impact of Electric Motor Size on Pre-Transmission Hybrid Vehicle 0-60mph Acceleration | 125
6.13 Comparison of Pre-Transmission Hybrid Vehicles Acceleration and Fuel Economy with Different Electric Motor Sizes | 125
Chapter 1: Introduction

1.1 Research Objective

The primary research objective of this thesis is to propose a comprehensive methodology for the model-based development of automotive systems, with focus on control systems. However, the proposed framework holds well for the development of any system in a broad sense. The goal is to propose generic methodologies with a specific, detailed example that highlights the entire development methodology, drawing inspiration from existing industry practices and standards that would help CAR stay competitive in the collaborative system development scenario.

1.2 Scope

The Ohio State University's Center for Automotive Research (CAR) is working with Chrysler LLC on a large research project that is jointly funded by Chrysler LLC and the U.S. Department of Energy. The project’s main focus is on advanced technology powertrains for light-duty vehicles with an overall goal of improving fuel economy for a Chrysler Town & Country minivan by 25% over the Federal Test Procedure drive cycle using only non-hybrid powertrain enhancements. As a research organization, CAR’s responsibilities include the development of a Vehicle Energy Simulator (VES)
with detailed models and tools to capture the actual physical system’s energy usage, as well as a supervisory Vehicle Energy Manager (VEM)). The role of the VEM is to improve fuel economy by 4-8% through two approaches


2. Vehicle Ancillary Load Reduction (VALR).

On the TMS side, the aim is to reduce fluid warm-up time during cold start and maintain stable temperatures thereafter. Extensive work has been done on the TMS modeling and analysis as well as control development, with the aim of reducing fluid warm-up time during cold start and maintain temperatures in the warmed-up state [1] [2]. On the VALR, detailed electrical system model development followed by the development of a control strategy for optimizing the electrical system’s usage have been done in [3].

This thesis is more related to the VALR approach, and specifically draws examples from the development of a control strategy to optimize the electrical system’s usage. The electrical system of the Town & Country Van was modeled and a control strategy for the alternator was developed as a part of the work done by [3]. The control strategy developed was based on the Equivalent Consumption Minimization Strategy [4] and was adaptive, and henceforth shall be referred to as the A-ECMS. This was taken over from the modeling environment, adapted for implementation followed by verification and validation. The controller was implemented in the Town & Country Van by means of Rapid Controller Prototyping and is experimentally being validated, as on date.
Through each stage of this thesis, parallels have been drawn to the development of this A-ECMS controller, the verification and validation done before implementation and the rapid control prototyping procedure that was followed.

1.3 Document Layout

Excluding the current chapter, which is an Introduction, this thesis document has been laid-out as described below:

- Chapter 2 identifies the state of the art and provides sufficient background to the comprehensive process being proposed for model based control.

- Chapter 3 explains the model based control development process, with focus on the initial phase that involves conceptualizing the system to be built.

- Chapter 4 takes over the conceptualization outcome of chapter 3 and highlights the rapid prototyping process for implementation and the various Verification and Validation methodologies involved before the final integration and release of the entire system.

- Chapter 5 takes over the model of the control system built in the modeling environment in chapter 4 and highlights the various Verification and Validation methodologies including the rapid prototyping process for experimental validation before the final release to the customer for integration into their platform.

- Chapter 6 is a case study of the concept selection process highlighted in chapter 2 and explains the methodology followed for rapid vehicle architecture selection as a part of the EcoCAR2 competition. Sufficient background on the competition is presented before the concept selection process.
• Chapter 7 includes conclusions followed by suggested future work from the perspective of both general model-based approaches as well as the Electrical System Optimization problem.
Chapter 2: State of the Art

2.1 Introduction

This chapter aims at providing sufficient background for the work being proposed. This work intends to propose a generic methodology that can be specifically adapted to different design tasks. An introductory section on the current overall product development process is provided to facilitate a better understanding of the bigger picture. Existing methodologies and practices prevalent in the automotive industry form a major part of the motivation behind this work. Various design methodologies and processes that exist today with respect to design of automotive control systems are reviewed with focus on model-based control system development, verification and validation involving SiL, MiL and HiL systems, automatic code-generation, system integration and final system implementation. Improved model-based systems engineering approaches relevant to the development of automotive systems and other existing practices and challenges ahead collectively provide sufficient background to the comprehensive process being proposed.
2.2 Product Development Processes

New product development can be summarized as the complete process of bringing a new product to market. This starts right from an idea, which typically is a solution that is aimed at addressing an existing problem, and ends with the realization of the finished product. A product development process is the sequence of steps or activities that an enterprise employs to conceive, design, and commercialize a product [5] A generic process that would best describe any new product development in today’s scenario as indicated by Ulrich and Eppinger is shown in Figure 2.1.

![Figure 2.1: A generic product development process](image)

The above product development process shown in six phases is very generic and can be adapted to meet specific products. The purpose of a generic process is to establish a framework that can later be modified to suit different scenarios. The authors in [5] provide significant detail into each of the above phases, while addressing both market-pull and market-push scenarios. Different product development process flows exist. Typically following a structured flow of activity and information, conventional product development processes include a review (or gate) between each phase.
to confirm the completion of the phase. Figure 2.2 shows few product development process flow variations.

![Diagram of product development processes](image)

Figure 2.2: Product development processes

The new product development process in general is discussed in detail by many authors, the above-mentioned being one example.

2.3 Model-Based Control

2.3.1 System Development

Advances in automotive control systems continue to increase fuel economy as well as improve a vehicle’s overall efficiency, performance and control. While the design of these control systems has come a long way with advances in processing/storage power and mechatronic systems, lead times involved with the development of these systems is one of the biggest market drivers today. With Original Equipment Manufacturers (OEMs) competing against each other in reducing the time-to-market, model-based control has become an established practice. Model-Based Design has been prominent
for decades in the aerospace industry primarily because of cost, safety and security concerns associated with building and testing working prototypes.

The optimization of an automotive mechatronic system with acceptable quality and time-to-market is not generally supported by traditional engineering processes. Instead, it mandates the adoption of a development approach where mechanical and controls engineering are interlocked throughout the design process, enabling upfront impact analysis and validation of different vehicle architectures and detailed designs. More specifically, a Model-Based engineering approach, using simulation models representing the controlled and controlling systems helps achieve this. Availability/development of suitable models for control model development (MIL), control software development (SIL) as well as for validation of the actual controller (HIL) are the basic assumptions and are a part of the process. The automotive manufacturing industry needs to adopt an upfront virtual design and testing approach, combining accurate simulation models of control software and the underlying physical systems, while securing a comprehensive and well-managed testing process against functional, performance and safety requirements. Testing is done virtually to the maximum extent possible, breaking the traditional build-test-redesign pattern. This approach is called Model-Based Systems Engineering or MBSE.

2.3.2 Verification and Validation

Verification and validation is perhaps the most important phase of any development process, as it helps identify how well the product being built matches up against the requirements it was intended to satisfy as well as specifications that were used to build it. While the terms verification and validation are interchangeably used by
many, a clear difference exists between them. Both are separate processes with a distinct purpose, and the distinction begins with the definition of requirements and specifications.

Requirements in short, are indicative of what tasks a system is desired to perform. Specifications, on the other hand, is a definition of how the system has to work to perform a desired task. Verification is done to ensure specifications are met, while validation is done to ensure that the requirements are met.

Figure 2.3 provides a clear overview of the relationship between requirements and specifications and how verification and validation are linked to them.

Figure 2.3: Requirements and Specifications - Validation and Verification

From the perspective of model-based automotive control, this verification and validation process begins right from the initial phase and forms an integral part of the development process. Simulations form the basis for all these model-based
verification and validation processes - the most common and well established processes are explained in the following section.

**Model-in-the-Loop**

Model-in-the-Loop (MiL) simulations help study the performance of the system and design the control algorithm(s) in a virtual environment. Major advantages of starting with MiL simulations are

- MiL simulations are inexpensive, easier to implement (but are most effective mainly in the early phases of development.)

- The control and plant models can be separate but run in the same environment and are connected directly by a virtual bus that transfers signals to and from.

![Model-in-the-Loop Simulation](image)

**Figure 2.4: Model-in-the-Loop Simulation**

At the early stages of development, it is important to build models and develop test cases keeping in mind future Verification and Validation processes right from the MiL phase. Figure [2.4] provides an outline of a typical MiL setup.
Software-in-the-Loop

Software in-the-Loop (SiL) simulations help verify that code generated from a model will function identically to the model. Control models created in the modeling environment are built and compiled to software code (automatic code generation tools are generally used - e.g. Simulink Coder). Software code runs in the modeling environment while the plant model remains the same as in the case of MiL. SiL simulations offer the following benefits:

1. Increased simulation speed by including compiled code in place of interpretive models

2. Inclusion of control algorithm functionality for which no model exists (legacy code for e.g.)

Figure 2.5 provides an outline of a typical SiL setup.

---

Figure 2.5: Software-in-the-Loop Simulation
**Processor-in-the-Loop**

Processor in-the-Loop (PiL) is an intermediate step between software in-the-Loop and hardware in-the-Loop where the control system is converted to code and flashed onto the embedded target. The embedded target interfaces directly with the plant model running on the modeling environment. A major reason to perform a PiL simulation is to double-check the system for errors associated with the code generation process (if any). Figure 2.6 provides an outline of a typical PiL setup.

![Figure 2.6: Processor-in-the-Loop Simulation](image)

**Real-time vs Offline Simulations**

Realtime simulations are necessary to be able to verify the working of the system under development under all possible scenarios. While the initial stages of Verification and Validation happen in a software environment, there is only a certain level of
behavior of the system that can be characterized in offline simulations. For an automotive control system, the timing at which a process happens is as important as the process itself. Being able to simulate these in real-time offers a significant advantage in the early stages of development by providing a better representation of the actual physical system. Towards the end of the development process, real-time simulation helps verify and test the system without having to necessarily spend a lot on an actual physical prototype, saving the latter for the final validation process. The following methodologies allow real-time simulation of systems and hence are used extensively in lieu of actual prototype hardware during developmental stages. The following set of figures 2.7 show offline simulation, real-time simulation and simulation over-run from the perspective of model simulations.

Figure 2.7: Offline Simulation, Real-time Simulation and Simulation Overrun
**Hardware-in-the-Loop**

Hardware in-the-Loop simulations facilitate validation in real-time with actual sensor and actuator signals. The hardware in the loop is typically a dedicated simulator with the plant model running in real-time. HiL enables extensive testing of multiple scenarios including potentially dangerous test cases without having to actually damage real mechanical/electrical systems. The simulator is capable of handling test drives outside the range of what real vehicles can do and hence can be used for fault-diagnosis validation at low costs. More importantly, the tests are reproducible and automatable (lights-out testing), making HiL simulation based testing one of the most important phases of Verification and Validation - this process is extensively used in the industry and has been instrumental in reducing developmental lead times and costs. Figure 2.8 provides an outline of a typical HiL setup.

![Hardware-in-the-Loop Simulation](image)

**Figure 2.8: Hardware-in-the-Loop Simulation**

HiL setups can be scaled across different simulators and provide the capability of including actual components as they get production-ready in the development process.
HiL simulators can be interfaced across multiple ECUs through standard automotive networks. Figure 2.9 shows the capabilities of a typical HiL setup.

![Figure 2.9: Hardware-in-the-Loop Simulator capabilities]

**Figure 2.9: Hardware-in-the-Loop Simulator capabilities**

### 2.4 Literature Review of Model-Based Development Techniques

The following section is based on available literature on various model-based development techniques and methodologies being followed in the industry.

[6] focuses on model-based development of electronic control units (ECUs) in the automotive domain. The use of model-based approaches solves requirements for the fast-growing integration of formerly isolated logical functions in complex distributed networks of heavily interacting ECUs. The authors propose a so-called system model
which comprises all of these constituents: the modeling language Automotive Modeling Language (AML), its mapping to the Unified Modeling Language (UML) which represents the modeling language standard for object-oriented system development as well as a system of abstraction levels which will help the AML user to achieve a well-structured development process.

Through an examination of the process and methodology used during an actual project with an automotive OEM, [7] indicates some of the lessons learned during the project implementation phase of migration to model-based engineering for the software development of electronic control units. It also highlights how the deployment of an effective model-based engineering process framework can significantly benefit an organization. Figure 2.10 shows a comparison between the traditional Systems ’V’ approach vs a model-based ”V” approach while Figure 2.11 shows a model-based software development process for ECUs as indicated in [7].

The authors also propose detailed definitions for various process steps. Figure 2.12 graphically illustrates the various process steps defined in the creation of a functional specification.

The authors in [8] propose a model-based development process along with a case study on automated testing and revalidation of the incremental model-based software development. Model-Driven Development is based on a visual language instead of human language or programming language. The target profile of a system as described in the agreed set of requirements is transformed into a visual language that allows description of behavior, how to use a feature, how a component interacts with its environment and whatever else needs to be considered. Every requirement is visualized
Figure 2.10: Traditional vs Model-Based Development for ECUs [7]

Figure 2.11: Model-Based Software Development process for ECUs [7]
in one or more model elements, and all model elements together are the model of the target system.

A visual model-based representation reduces the chances of misinterpretation and eliminates rework by removing ambiguities often associated with natural, spoken languages. Making the job easier for system developers, this forms the basis for record/playback test script based validation as well as automated generation of test cases and test vectors. The testing and validation process in its entirety can be improved by automating and reusing tests, while maintaining quality despite reduced lead-times. The authors propose the reuse of development artifacts in three dimensions as shown in Figure 2.14. One is along the process chain - modeled functional
requirements can be reused to auto-generate the test cases, translating modeling effort as payback for generating extensive test scenarios, test cases, and test vectors. The second dimension of reuse is along the time, between different versions and the third is across different vehicle lines.

[9] highlights issues associated with the creation of embedded software for automotive ECUs and explores the option of using model-based methods to design, test and implement systems. Previous surveys indicate at least five reasons for embedded system development projects in general being late - changes in specifications, application complexity, inadequate specifications, lack of developers and finally, lack of testing personnel. While this is not specifically targeted at the auto industry, the results hold well even in this application, considering the size and complexity of the automotive embedded systems industry.
Figure 2.14: Dimensions of reuse - Model-Based Product Development for Embedded Systems
Model-based design efforts involve abstraction of product and control system model from a textual level to a visual level. Hierarchical design partitioning is a natural extension of the visual paradigm, and allows even the largest systems to be represented in a compact and meaningful way, with the benefit of re-usability in subsequent design efforts [9].

Automotive manufacturers and suppliers have largely adopted model-based design methods, but to different degrees. Each design situation is unique, with varying levels of legacy and opportunity for change that drive the degree to which a model-based design approach can be adopted. The author highlights a few actual examples from the automotive industry, indicating the effectiveness of model-based design methods in a variety of automotive applications. A few of those examples are presented here as well to emphasize the need for model-based design [9].

A model-based design approach allowed Eaton to design a hybrid-electric powetrain system for trucks before prototype hardware was available. Real components were integrated eventually with the embedded control system using a real-time Hardware-in-the-Loop approach. Component models provided the specification for real-time software that could accurately simulate the behavior of the components that were not yet integrated [9].

To meet demands for increasingly complex new vehicles while continuing to reduce costs, where possible, Jaguar develops and tests new functionality using existing production vehicles instead of building expensive prototypes, following a model-based approach and using general purpose ECUs [9].

A model-based design approach was selected by Toyota as a way to bridge the gaps in their traditional automotive electronics development process and was used
to create executable specifications to consolidate the work of specification writers, control designers, and programmers. Shortened design cycles, reduced amount of hardware prototypes required, and innovative new products being developed are some benefits realized by Toyota as a result of adopting a model-based design approach [9].

Model-based control system development is not a complete solution that ensures coverage of all potential failure modes or unexpected behavior scenarios. When a new/improved control algorithm is developed, the same can be tested for functionality in a short duration with the help of rapid control prototyping. [10] describes a model-based development process for embedded control systems with emphasis on use of rapid prototype controllers and Hardware-in-the Loop (HiL) testing.

Various model-based embedded control system development process frameworks are highlighted by the authors - Figure 2.15 explains the development process using offline simulation. Mathematical models are developed to represent the physical behavior of the system, which in turn is used to develop the control algorithm and eventually refine the control based on feedback from offline simulations. These models are intended to capture the input/output behavior of the real plant and assist the control development process, and hence are often known as control-oriented models.

Figure 2.16 shows a rapid control prototyping based process. RCP tools assist the verification and validation process by allowing developers to make changes to the algorithm based on the testing results. Rapid Control Prototyping (RCP) tools enable direct generation of code from graphical controller models, which in turn can be used downloaded on a high-end microprocessor based simulation system. Live calibration of control parameters is another major advantage of RCP, and this significantly helps reduce the post-development lead-time.
Figure 2.15: Control Development process using offline simulation [10]

Figure 2.16: Control Development process using RCP [10]
RCP with hardware-in-the-Loop (HiL) testing based development process is indicated in figure 2.17. The advantage of using a HiL system is being able to test the safety and diagnostic features of the controller much earlier in the development phase. Also, operating conditions that are way beyond the usual operating conditions and are potentially hazardous can be easily tested very safely within a lab-environment by simulating the plant model and using a HiL system. The HiL plant has to be highly representative of the actual system and also has to be run real-time to be able to test the controller being developed.

Figure 2.17: Control Development process using RCP and HiL [10]

Auto code generation based process is also indicated in the same paper, shown in figure 2.18. Auto code generation in detail is discussed in a later section.
A flexible engine control architecture for model-based software development is proposed in [11]. The authors propose the same through four key features - separation of target-dependent and target-independent algorithms, partitioning of the control system at the system-level, use of hardware I/O blocksets to facilitate system-level automatic coding, and the use of a data pooling concept to simplify signal propagation throughout the architecture layers.

A Simulink-based architecture design with separated functional entities is proposed by the authors. The target-independent part of the architecture includes all functional features independent of the implementation environment, while the target dependent portion involves functional features that are strictly environment-dependent. To enable exchange of information between these, a global data pooling
Figure 2.19: Flexible Engine Control Architecture

A method is proposed, where signals are organized and stored in a global data pool accessible to any sub-system within the architecture. Detailed architectures are explained for both target-dependent as well as target-independent sub-systems. The entire architecture proposed is directed towards seamless transition across different implementation scenarios as well as integration or upgradation or modifications involving legacy code resulting in a faster development process.

[12] describes a product development process that promotes widespread usage of modeling techniques (such as physical system modeling and simulation, model-based control system design, rapid prototyping, etc.) over all technical domains involved in the product (mechanics, hydraulics, electrics, electronics, software, etc.) as well as over several stages of the product development process. Rather than having standalone systems for different tasks/domains, one commonly used tool chain is applied to the whole process, allowing for the sharing of one commonly used holistic
model, which elaborates throughout the development process. The holistic model is expected to become the overall technical specification of the product, and also promote a high level of reuse between departments, domains, customer and supplier, and even different development projects. A case study about the development of an electric powertrain of a recreational vehicle following the holistic process is presented in detail.

Figure 2.20 shows their proposed holistic model as an outline. The process has been described in detail as a four-step process that involves physical design, model-based control system design, porting to ECU and finally the replacement of the physical model by the actual system.

---

Figure 2.20: The holistic model including all domains of a product under development [12]
Figure 2.21 describes the proposed process, with progress in time along the horizontal axis and the degree of model-based vs prototype-based development processes along the vertical axis.

Figure 2.21: Visualization of the development process [12]

[13] proposes an integrated approach aimed at overcoming the tool and process inefficiencies associated with model-based development of automotive embedded control software. The 'integrated' approach attempts using a homogeneous tool to integrate
the software design, analysis, implementation, testing, validation and calibration activities, reducing various non-value added steps such as data and test case translation across the phases.

Increasing model complexity and domain-specific modeling environments built upon proprietary platforms have made integration of such models with controls development tools difficult. This leads towards control developers either losing the advantage of their own model-based environment or ending up with limited access to the plant model parameters when they settle for scaled down domain-specific models.

Multiple model versions, interface, formats and legacy code existence are all challenges from the perspective of function-oriented control development when it comes to a large organization such as an automotive OEM.

Moving across system and sub-system testing in the SiL, MiL and HiL phases calls for a consistent set of tools, datasets and methods, to ensure complete verification and validation.

Most model-based tools do not provide for controlling the model execution speed, which might be necessary in some cases. This is important in cases that may require slower execution to enable observation of the impact of parameter modifications.

Finally, there exists a significant amount of legacy C-code that has been extensively tested and proven. When developing new systems or upgrading from existing systems, being able to use this C-code in a meaningful way right from the SiL phase for complete system Verification and Validation is an existing challenge.

The author in [13] builds upon various challenges that are faced while using Software, Model and Hardware-in-the-Loop development phases to propose the integrated
approach using ETAS software. ETAS INTECRIO is shown as a solution for both homogeneous as well as heterogeneous SiL and MiL platform in figure 2.22.

INTECRIO is a software development integration platform with an open XML based interchange format to accommodate third-party tools. INTECRIO’s capability to generate an ASAM-MCD-2MC description file (*.A2L file) facilitates the usage of same tools for MiL testing as that will be used later for vehicle calibrations, eliminating non-value added steps and improving exchange of information between calibrators and control engineers. INCODIO, another software developed by SYTECS GmbH, is a C-code synthesizer that creates an interface file for INTECRIO, while preserving all real-time scheduling information and data. LABCAR AUTOMATION from
ETAS facilitates test scripts in complex programming languages with test automation functionalities, keeping the scripts independent of the test environment, the model development environment and the hardware used for HiL and thus allows MiL test cases to be re-used without adaptation in the HiL. Figure 2.23 shows the open architecture of LABCAR AUTOMATION.

Figure 2.23: Open architecture of LABCAR AUTOMATION [13]
3.1 Introduction

Automotive OEMs allocate significant resources (money as well as human resources) towards the development of new control systems. More often than not, the small teams/groups that work on the development of a smaller sub-system (which eventually is to become a key part of the final integrated part) never get to know the intended role of the system in the big picture. While all effort is made to rigidly follow the sub-system specifications, differences may arise eventually when systems are integrated, resulting in higher lead times during later stages of product development. This is just one possible scenario and multiple other scenarios might exist where the product development process takes more time/resources than intended. When a product/system is being developed, it is important to understand the background and the overall process that forms the bigger picture - the same is being attempted in this section.

The objective of this section is to outline a comprehensive process that can facilitate model-based development of automotive control systems - the process however in a generic sense will hold good for the development of any system/product and is analogous to the new product development process.
The process outlined here is applicable to a multi-disciplinary research organization similar to the Center for Automotive Research (CAR) at the Ohio State University. The example scenario used here is for the development of a control system that optimizes the electrical system of a passenger van, making better use of the vehicle’s battery.

### 3.2 Opportunity Identification Process

An opportunity is an idea for a new product/system that will help address an existing (engineering) problem. It is a perceived means of generating value that previously has not been exploited and is not currently being exploited by others. Opportunities can also emerge from a complex pattern of changing conditions juxtaposition of conditions which did not exist previously can lead to a new opportunity at some point of time. Opportunities are not always about developing completely new technologies or products. Integrating/linking the right existing technologies or making use of the same from a different perspective can still be an opportunity to pursue. From the perspective of automotive control, opportunities exist/continue to grow in multiple domains including efficiency improvement, energy utilization, environmental impact (emissions) with increasing goals pertaining to fuel economy, emissions control, passenger safety, performance and finally, cost-reduction. The above listed form a major part of what the industry looks at as an opportunity - however, many more exist in multiple other domains. While significant research continues on developing new systems, there are sufficient opportunities in improving existing systems.

Opportunity identification generally involves three steps as follows:

1. Opportunity listing
Figure 3.1: Types of Opportunities [5]
2. Opportunity screening/evaluation

3. Opportunity prioritization/selection

The opportunity listing section focuses upon opportunity generation techniques while the screening/evaluation section is about short-listing the listed opportunities. Prioritization/selection is the final part of the opportunity identification process and leads towards the next step, which is extracting the engineering problem. Figure 3.2 shows the opportunity identification process in general.

![Figure 3.2: Opportunity identification process](image)

3.2.1 Opportunity listing

Well documented opportunity generation techniques exist in the industry, as shown in most product-development processes. However, from the perspective of
developing automotive control systems, not all of them are applicable. The follow-
ing techniques, adapted from new-product development methodologies, can help in
generating opportunities:

**Opportunities from core competencies**

Core competencies and expertise pertaining to specific technologies can provide
for many opportunities. Resource availability and uniqueness of the same goes a long
way - a resource that is valuable and not easily substitutable and less common in the
industry is the starting point when looking out for opportunities. The term resource
here encompasses all of the following:

- Human resource - skilled research/engineering personnel with expertise in spe-
cific domains
- Engineering resource - Testing equipment and infrastructure

More often, core competencies of an organization go hand in hand with the reputation
it builds over a period of time - expanding these core competencies is by itself an
opportunity. Core competencies in hand can be extended and reformulated to identify
opportunities. For example, for OSU-CAR with expertise in energy management of
hybrid electric vehicles, the energy management for ancillary loads in a conventional
powertrain was an opportunity to apply its core competency.

**Opportunities outlined by customers**

Opportunities are not always generated by the developer and more often, the
research organization works as a solution provider for the industry, collaborating
with the same. Industry projects typically can involve either the industry specifically
stating what it needs or a more exploratory scenario with both the industry as well as the solution-provider working together towards the opportunity generation process. In case of the former, inputs for the industry in terms of opportunities are derived from various sources. Some of these include

1. Internally generated - opportunities generated internally based on the company’s goals/targets or opportunities generated as a result of internal research.

2. End-users (customers) - addressing what the end-customer of the product, an automobile in this case, wants.

3. Competition - the need to catch up with competition or retain existing advantage and stay competitive. Achieving greater performance than competition or reducing weakness in comparison are both possible scenarios.

The background of the opportunity generation process that happens within the automotive industry is beyond the scope of this thesis. In a scenario where the opportunity generation is in collaboration with the research organization, a systematic approach can be taken.

**Opportunities from existing entities**

Opportunities are not always created/generated. They can be identified right away from some existing problems/products. The most straight-forward scenario is where various problems of an existing system have already been identified either through a detailed analysis or based on feedback from the customer(s). Either ways, this can be a great source of opportunity when these problems are looked upon systematically and categorized. When dealing with existing products, opportunities may also exist
in the sense of making these products better in comparison with the original. Sometimes, opportunities might exist even in making an existing technology available for a scenario where it has never been used before. Better packaging or implementation of the same product can still be a project worth pursuing. From the perspective of the auto-industry, for e.g. to address any particular control problem, there may exist a solution in place that serves the purpose. However, a different implementation strategy might exist which can be advantageous from the perspective of effort or hardware requirement etc leading towards reduced manufacturing costs and so on.

3.2.2 Opportunity Screening and Evaluation

Depending on the size of the organization and the degree of involvement of the individual members of a team in the opportunity screening process, different methodologies can be followed separately or as a combination to eliminate opportunities not worth further pursuit. In cases with a high number of initial opportunities in hand, regular brainstorming does not always help as such discussions when combined with multiple screening criteria can become unfruitful or result in decisions that are highly biased by personal choices. A very effective screening process should involve minimal additional investigation, but with the right amount of focus on each opportunity to make sure no potential opportunities are lost. The opportunity identification process overlaps at times with the concept selection process (explained later in this chapter), both requiring similar decision-making processes. The following are some of the opportunity screening processes followed in the industry, adapted and relevant to this discussion on automotive systems. When multiple opportunities exist, the process of
identifying a set of opportunities worth further investigation can be considered to be a tournament, where only the best ideas prevail.

Criteria Based Screening

A simple method to screen opportunities is to identify a set of screening criteria that can be answered with a yes/no. Real-Win-Worth-it (RWW) criteria is one such strategy developed by 3M and being used within established companies to select product opportunities. The name, Real-Win-Worth-it, summarizes the three questions the organization attempts to answer when screening:

1. Is the opportunity real? - Evaluate the opportunity with criteria including market size, price points, technology availability and capability to meet future demand.

2. Can this opportunity be won? - Evaluate the opportunity with criteria including sustainability of competitive advantage, patent protection, branding, resource availability and competitive advantage.

3. Is the opportunity worth it financially? - Evaluate the opportunity with criteria including financial and developmental resource availability, return of investment etc.

From the perspective of an organization developing solutions for the automotive industry, while criteria related to direct competition might not apply, the general RWW framework of evaluating an opportunity systematically is still applicable.
Voting based screening

Multi-voting is a process where decision makers are made to vote on multiple opportunities, with the results analyzed by a coordinator and decisions/further discussions made based on the results of this analysis. While conventional multi-voting processes can be as simple as a single session with all decision makers getting together and voting for their choice, this process is effective only when the initial number of opportunities under consideration is low. In cases with over 50 opportunities under consideration, there is too much information for a decision maker to process in one session and hence, automating the voting process/facilitating the same electronically can help. Various online tools exist that allow for opportunity tournaments, one such tool widely used being the Darwinator [14]. The Darwinator is a web-based system that enables people to run tournaments. Users submit various ideas following which the same are evaluated through voting or expert assessment. The advantage of using a web-based tool is being able to get decision makers together even if they are not in the same geographic location. Moreover, when the organization is collaborating with industry, tools like this can be a better platform for collaboration in the initial phases without having to wait for matching schedules to setup appointments or opportunity screening discussions.

3.2.3 Opportunity Prioritization/Selection

The Opportunity selection process is a more precise follow-up of the screening process. Methodologies applied to concept selection (discussed later in this chapter) apply to the opportunity selection process. Some of the various opportunity selection criteria that can be considered by an organization similar to CAR-OSU include
1. Its competitive strategy - technology leadership oriented or customer oriented

2. Depth of the organization’s existing knowledge of the market/technology

3. Fit with the organization’s current research activities/products

4. Fit with the organization’s current infrastructure/capabilities

5. Fit with the academic component of research capabilities

6. Potentials for patents, IP rights

7. Project timing and fit with existing projects(resource allocation)

In the scope of a research organization similar to CAR-OSU, greater focus is needed on the concept selection process as the candidates in the opportunity screening process are typically low enough to facilitate an easy opportunity selection.
3.3 Engineering Problem Definition

The opportunity identified and selected as a consequence of the previous process is still a generic interpretation and not in a form that can help the team proceed with the development process and hence needs to be articulated into a simple problem statement from the engineering stand-point. Product development literature calls this a mission statement; however, for a research organization, the problem statement is a clear and concise description of the issues that will be addressed in the research work.

The objective of the engineering problem definition process is to focus the attention of the problem solving team and establishing a statement that would eventually define the scope of the entire project. The engineering problem forms the basis for the requirements identification process and hence the specifications definition process, as seen in the later stages of this document. While the term "Problem" is extensively used throughout this section, the meaning is not always literal and it primarily means the opportunity being addressed from an engineering perspective. The engineering problem definition addresses primarily four important questions

1. What is the opportunity? What is the ultimate deliverable?

2. Who is the customer?

3. Why is this an opportunity? Why is this problem being chosen?

4. When is the deliverable expected? What are the deadlines?

Answering the above questions in a concise manner is the problem definition process; however, it is not mandatory to answer all of those and the problem definition can vary
on a case to case basis. In some cases, the opportunity identification and hence the engineering problem definition process not necessarily is conceived in house. Projects done in collaboration with the industry can at times directly start with the customer (an OEM for *e.g.* already having his requirements at a very high-level ready. With these requirements, they can collaborate research organizations like CAR-OSU, in case of which the problem definition process and the project definition process are often the same, with the high-level requirements being defined at this stage. As described earlier in the scope of this document, the focus of the process under review is on the Ancillary load management part of the project with Chrysler/DoE, and on the control of the alternator in specific. From the same perspective, the engineering problem definition is indicated in 3.4.

![Figure 3.4: Engineering Problem Identification](image)

**Figure 3.4: Engineering Problem Identification**
3.4 Requirements identification

Identifying customer needs, often interchangeably used with customer attributes (or) requirements is the first step towards a structured development process, as these requirements are eventually what the end product is desired to satisfy. The identification/definition process of customer needs should yield a set of requirements that are independent of the concept that might be chosen at a later stage to realize the system being developed. The most important objective at this phase perhaps, is to identify what the system has to do and not how it needs to be done. Requirements form the basis for the entire validation process of a system under development. The following are imperative to ensure a complete and successful system development:

- Requirements should be captured in their entirety, without ambiguities, incorrectness/contradictions.
- Requirements continuously change - constant updation and deletion or creation of new requirements throughout the entire development process is of paramount importance.

For a control system in particular under development, requirements can be identified as instructions describing what functions the control is supposed to provide, what characteristics it is supposed to have, from the customer’s perspective.

Requirements can be classified as:

1. Functional requirements: state how the system should act.
2. Non-functional requirements: state the quality characteristics of and constraints on the system.
3.4.1 Requirements Management

Requirements Management’s main purpose is to establish a common understanding between the customer and the project, forming the basis for planning and managing the project. This encapsulates not just capturing requirements in the early stages of the project, but also monitoring changing requirements and ensuring the same is made accessible to the entire development spectrum. From a Project management standpoint, requirements management plays an important role in maintaining the scope of a project, which is one of the key drivers of a project, as shown in the project management triangle in figure 3.5.
RM (Requirements management) can be as simple as capturing them as text in simple documents/spreadsheets to dedicated RM tools. However, the former offers limited functionality in terms of being collaboration-friendly and more importantly, does not support traceability from models to requirements and vice-versa unless specified. Various tools specifically designed for collecting and managing requirements exist in the industry, the most notable ones being Rational DOORS (Dynamic Object-Oriented Requirements System) from IBM [15]. DOORS is a requirements management tool that enables users to select, create, or relate requirements in an easy to navigate user interface. Featuring multi-user access, filtering and sorting of data, DOORS allows users to link requirements with models and perform traceability and impact analysis across specifications and subsequent levels. Another perspective towards RM, perhaps at a later stage following the requirements documentation stage is the usage of model-based tools to retain information about requirements. Extensive work has been done before to capture the requirements details in models, a methodology well described under model-based systems engineering. Very similar to the software engineering process of managing requirements, techniques include using UML (Unified Modeling Language) and various adaptations of the same to capture various requirements and track the same throughout the development process. UML combines techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, right from the requirements definition stage and across different implementation technologies - and hence makes sense especially if it can be used for automotive software development. However, application of these techniques is beyond the scope of this document.
While DOORS or UML based tools were not used for requirements management in the example being highlighted in this chapter, awareness of the existence of such tools can help organizations implement better requirements management practices.

### 3.4.2 Electrical System Optimization problem - Requirements

For the example under consideration in this thesis, the requirements were identified based on the inputs from Chrysler LLC as indicated in figure 3.6

![Figure 3.6: Requirements identified for the Electrical System Optimization problem](image)

**Figure 3.6:** Requirements identified for the Electrical System Optimization problem

### 3.4.3 Other constraints

Not all constraints are often captured under the customer needs/requirements. When developing automotive control systems for example, while various objectives
and constraints on the system are identified, it is important to capture constraints such as

1. Cost of implementation - additional hardware requirements.

2. Target hardware/software platform in which the system is to be implemented.

3. Availability of resources (memory, processing capability) in the parent controller (ECU for example) and interface requirements.

Requirements of this sort are often overlooked upon and result in longer lead-times towards the end of the development cycle. While the intended functionality and objectives are met, the control system is not completely delivered until issues involving these are resolved. Hence, it is good practice to capture requirements related to implementation specifically and update them often.

3.5 Target Specifications

Specifications are the result of requirements being transformed from the customer’s language to a form that is a precise description of the same with a metric and a value. Specifications become the reference document often during development and hence, it is important that specifications when defined capture all requirements in the first place. This happens over two phases, with target specifications defined first followed by the final set of specifications, against which the system is verified against as it is being built. Target specifications are the preliminary set of specifications defined from the requirements. They are identified before the concept selection process and more than often, the concept selected results in modifications to the specifications, resulting in a final set of specifications that are referred to as the ”Final Specifications”
The most important phase of defining target specifications is identifying the needs-metrics matrix, which helps identify the relationship between various customer needs, which is often used loosely instead of requirements, and metrics that are definitive and set targets. There may exist customer needs that are not quantifiable with metrics. However, they are beyond the scope of the process under study. The following considerations are imperative when defining the metrics.

1. Metrics, as the name suggests should be measurable and tangible quantities - abstraction is not desired in this stage of the process.

2. Metrics should be representative of what the system is expected to do and not how.

3. Establishing metrics that are dependent variables will offer greater freedom when the system is being designed.

4. Metrics when defined in the initial stages should not be too rigid- specifying a range of values is good practice at this stage to offer flexibility when designing.

The needs-metrics matrix is an important part of the House of Quality, which is a graphical technique used in QFD (Quality Function Deployment) [16]. The rows of the matrix correspond to the customer needs (requirements) and the columns of the matrix correspond to the metrics. This matrix helps capture all customer needs as well as the defined metrics and serves as the foundation for the specification document. Figure ?? indicates the way forward once the needs-metrics matrix is identified.

The target specification definition process when developing control systems is best done with the involvement of the entire team brainstorming, while still following a
process similar to the one shown in figure 3.7. When specifications are more often not necessarily direct customer needs but constraints imposed on the system, there is not much flexibility in varying these target values. More often, scenarios exist where the customer directly approaches with a well defined set of final specifications that the system has to conform to.

3.5.1 Electrical System Optimization problem - Target Specifications

Figure 3.8 shows the needs-metrics matrix that was developed for the alternator control problem based on the customer needs as indicated by Chrysler and metrics that were defined by the project team. The specifications defined based on this
Figure 3.8: Needs-Metrics Matrix for the Electrical System Optimization problem
needs-metrics matrix is the target specification of the system to be built. The final specification definition happens after the concept selection process and will be explained in a later section.

3.6 Concept Selection

The first step of the concept generation and selection process is the interpretation of the problem in hand. With the same objective, describing the elements of the problem using simple models to start with can initiate the process. Figure 6.8 shows the proposed concept generation and selection process that can be followed when developing a control system. This involves a model-based approach in the sense, the problem is first decomposed into submodels that constitute the functional flow of the system and then worked upon individually to generate possible concepts. The objective of this phase is to not to define the working of the system in detail involving extensive modeling and then search for concepts, but to understand at a high-level the problem in hand and decompose the same to identify potential concepts.

3.6.1 Electrical System Optimization problem - Concept Selection

Figure 3.10 shows a simple functional block diagram of the existing electrical system of the vehicle.

Decomposing the above system into individual sub-systems, each of these sub-systems were looked upon as subproblems, with the problem statement being scope for optimization of the system. Roughly based on the methodology defined in the previous section, various concepts were identified and eventually rejected based on
Figure 3.9: Concept Selection Process
Figure 3.10: Electrical System Optimization problem - functional decomposition

multiple brainstorming sessions. In short however, the following steps, as shown in 3.11 were involved in the concept selection process.

3.7 Sub-system identification

From a very broad perspective, the development of an automotive system would involve not just control but mechanical and electrical sub-systems. The sub-system identification phase is to primarily define these sub-systems and hence initiate the simultaneous development of these systems. However the scope of this thesis is limited to model-based automotive control system development, as explained in the forthcoming chapter. Interactions between various identified sub-systems are beyond the scope of this thesis. However, in reality, these exist and can have a significant impact on the development process as different sub-systems undergo design modifications through the development process. Figure 3.12 offers an overview of the proposed
Figure 3.11: Electrical System Optimization problem - Concept Selection Process
methodology for model-based automotive control development, including mechanical and electrical sub-systems.
Figure 3.12: A Comprehensive Process for Automotive Model-Based Control
Chapter 4: Model-Based Control: Development Phase

4.1 Introduction

This chapter aims at providing an overview of the model-based control sub-system development process. From the outer level, this process involves four major processes, with a lot of information exchange and interactions between the models developed. The intent of a model-based approach towards the development of control systems is to minimize or completely eliminate the need for an actual physical model when the controller is being built and tested. The advantages of models from this perspective is explained in detail under Model-in-the-loop testing in Chapter 4. Figure 4.1 shows a simple model-based control system design process approach. The focus of this chapter is primarily on the plant and control model design and development processes.

As seen in figure 4.1, the definition of a plant model and control model is the most important step in understanding the problem in hand. The plant model is a representation of the actual physical system. It is to be noted that in cases of systems with existing control, the plant model should not include the baseline controller. The control model as the name suggests is what is eventually to be developed in accordance with the defined specifications to control the system and hence meet the requirements identified.
4.2 Plant Model Development

4.2.1 Introduction

Involving models throughout the development process is the underlying principle of model based development for automotive control systems. The advantage of such an approach is evident in a scenario where the actual system to be controlled is not readily available as a physical system to the developers when developing automotive control systems. There are at least four possible reasons for such a scenario to arise:

1. The actual physical system (or other systems that it is dependent upon) is also under development and hence no prototypes are readily available.

2. The actual physical system is too expensive to be used in the initial stages of the control development process.
3. Usage of the actual physical system to develop the control involves a significant amount of risk as its behavior is not completely known.

4. Identify states (or) variables of a system that cannot be directly measured (or) would be expensive and time-consuming to experimentally obtain.

Evaluating all the above and more importantly, taking into consideration the basic foundation of model-based control - which is to use models throughout the development process, it is important to develop a model that is representative of the actual physical system under study for control. This process is often referred to as the system identification process, which is explained in the following section. The level of detail and fidelity expected of this representation of the physical system is dependent on a variety of factors, and are achieved by using different types of models, as explained in the following section.

4.2.2 System Identification

No physical system can be exactly modeled and approximations form an integral part of any model. Physical phenomena and behavior of systems can be represented by mathematical descriptions - however, limited prior knowledge of the system and lack of data prevent an exact mathematical model. The system identification process can be defined as an approximate modeling of the actual physical system based on prior knowledge and experimental data. The effort required to attain a very high level of fidelity in this modeling process depends on what is expected of the model and how it is going to be used. One of the desired outcomes when modeling a physical system could be that it should not be too complex and as a consequence, become unusable in the desired application. While system identification’s outcome is a plant
model that is representative of the actual physical system, it can also encapsulate the existing control logic in some cases. This holds good especially in cases of projects where the objective is not developing something completely new but to improve or change existing control systems. Understanding of the existing system is important in this case and hence the system identification process would include modeling that as well.

From the above definition of the system identification process, it can be inferred that the process depends primarily on:

1. Prior knowledge of the system.

2. Objectives of the systems (both the plant as well as the control system whose development it is going to serve)

3. Availability of data (experimental for *e.g.*)

The above are not independent parameters. Rather, information on one can help improve the other. For example, when the prior knowledge of a system is minimal, conducting some basic experiments can improve the understanding of the system and even impact the objectives. Likewise, better information on the system beforehand can help design a good process for experimental data collection which will support the model development process. In a very broad sense, the models developed under system identification can be classified as three types as indicated below:

1. White-box models

2. Grey Box models

3. Black Box models
White box models

White box models are developed on the basis of physical laws and additional relationships that correspond to various physical parameters - mathematical equations are used to completely represent the physical processes. The need for such a model that tries capturing all nuances of the working of a system is when a detailed representation is required for simulation.

Grey box models

Not always are all parameters about a system known, and uncertainties are the only certainties in some scenarios. In cases like this, while physical parameters and mathematical equations capture the working of system, adjustable parameters that can be tuned based on experimental data are used to obtain realistic predictions. Such models are known as grey-box models.

Black box models

In control applications, linear models that are representative of the system being controlled are more often sufficient. These models can be as simple as static maps derived from a significant amount of experimental data. The advantages of these models is the speed at which they can be run. However, their fidelity depends on the quality of the data used in the first place to arrive at these models. Such models that do not necessarily refer to the underlying physical laws and relationships of the process are called Black box models.

Figure 4.2 depicts the entire system identification process.
Figure 4.2: System Identification Process
4.2.3 Model Validation

Validation from the perspective of modeling is the process of evaluating a model’s performance in representing the actual physical system being modeled. Model validation is done on a variety of parameters including but not limited to accuracy, repeatability, reproducibility, and sensitivity to both exogenous and endogenous variables. The model validation phase of system identification primarily addresses the following questions:

1. Does the model serve the purpose of describing the actual physical system?

2. Is the model representative of the experimentally observed data?

3. Is the model robust enough? How sensitive is the model to its inputs or operating parameters?

The only input that can help validate the model being developed is observed data. A set of criteria are defined to establish how accurate/representative/sufficient/robust the identified system. The result of this evaluation forms the basis for the model validation process, and this is typically iterative until the model reaches an acceptable level.

Usual methodologies include using one set of experimental data to calibrate a model and the other set of experimental data to observe the model’s performance. When validating grey-box models for example, the calibrating dataset should be representative of the system in the first place, capturing its behavior across a complete operating range desired, and the validating data set can be different operating conditions that still fall under this range. In case of white-box models, the role of experimental data in the modeling process is minimal and hence validation is typically
done on a dataset that is across the entire operating range. For black-box models, the validating data typically falls under the data used in the first place to develop the models. Extensive techniques and methodologies exist, and along with statistical tools, specific metrics can be defined to compare the outputs/performance of a model in comparison with the actual physical system. The levels of acceptability of these metrics depend completely on the system under consideration and no hard rules exist in terms of model validation.

4.2.4 Control oriented model

When the system identification procedure’s outcome is a high-order mathematical model, it is not suitable for control system design purposes. Simplification of complex models is needed in order to perform simulations within an acceptable amount of time and limited storage capacity, but with reliable outcome. Scenarios may exist where on-line predictions of the behavior with acceptable computational speed might be desired for both developmental as well as final applications. Model Order Reduction, as the name suggests helps reduce the order of the model to bring it to a level that is still acceptable in terms of being representative of the system, while offering better execution speeds. Model Order Reduction tries to quickly capture the essential features of the original with sufficient precision. Various methodologies are described in detail in [17]. Model Order Reduction techniques have been applied to the development of control-oriented models as a part of the work in [2]. Specific Model Order Reduction methodologies are not discussed in this thesis and is beyond the scope of this document.
4.2.5 Electrical System Optimization problem - Plant Modeling

The focus of this thesis is not on the details of the development but the development process itself. The following section offers an overview of the electrical system development process based on [3]. For the electrical system optimization problem, the overall system consists of four main components:

1. Alternator
2. Battery
3. Electrical Loads
4. Controller (Voltage Regulation)

Figure 4.3 shows a schematic of the existing vehicle electrical system (along with existing control).

![Vehicle Electrical System with existing control](image)

Figure 4.3: Vehicle Electrical System with existing control

The basic functioning of the existing system can be summarized as follows
1. The various electrical loads on the system include but are not limited to the various lights, entertainment system, HVAC system, radiator fans, heated seats, mirrors and all other electrically operated equipment.

2. The battery of the vehicle is used only when necessary - engine cranking and as a buffer. The battery voltage is dependent on the state of the system and the battery current.

3. The battery current is the difference between the alternator current and the current demanded by the electrical loads.

4. Current produced by the alternator is a function of the field current, engine speed and battery voltage.

5. The field current supplied to the alternator is determined by the controller, which outputs a duty cycle. The controller is conservative as it compares feedback on the battery voltage to a temperature dependent reference voltage and generates a duty cycle correction to regulate the voltage.

With the above background information on the working of the existing control and electrical loads, the alternator and battery model development process is summarized in figures 4.4 and 4.5 based on work done in 3.

4.3 Control Logic Selection and Design Process

4.3.1 Introduction

This objective of this section is to explain the control logic selection process and subsequently, the control system design process. While multiple methodologies exist, model-driven approaches alone are taken into consideration in this thesis.
4.3.2 System Analysis

During the system identification process, the identification process by itself provides the developer(s) a fair understanding of the actual physical system. However, a more detailed and systematic approach to the problem in hand is a key input to the control logic selection process. The system analysis process is aimed at understanding the working of the actual physical system (or) process desired to be controlled and draw some inferences from the same, which will serve as the starting point for the logic selection process. System analysis can be done from multiple perspectives, including their behavior in frequency and time domains, input/output relationships, energy flows in the system and so on.
Figure 4.5: Battery Modeling - Grey Box model [3]
4.3.3 Control Logic Selection Process

The control logic selection at a very fundamental level should be able to meet all specifications as defined earlier and hence, all requirements at a higher level. The control logic selection process is driven by at least three different parameters as follows:

System Analysis outcome

Analyzing the system to be controlled provides the developer a clear picture of the problem in hand. The logic selection beyond this can be either based completely on the experience of the control design engineer as an individual or the outcome of a group brainstorming. There is not specific hard and fast rule that exists when it comes to selecting a control logic for a specific application. Multiple strategies can serve the purpose at similar levels of acceptability.

The design context

The design context here refers to the level at which the control system designer starts on the process. At least three different design contexts can exist:

1. From scratch - the control for the physical system has to be developed from scratch and no control exists in the first place. The logic selection in this case is often based on previous experience with similar problems coupled with experience of the system designer.

2. Adaptation - A control system exists and the new system is adapted from this as an improvement. In cases like this, the control logic can fundamentally remain
the same with few parameters alone being changed or calibrations alone being modified.

3. Research - Similar to the starting from scratch context. However, there are no constraints on the design process and hence the logic selection can be exploratory. Extensive usage of model-based techniques can assist in testing various strategies without the need for expensive rapid prototyping hardware.

Implementation constraints

Control systems developed for automobiles eventually need to be implemented at some level or the other. While research organizations often do not see the entire development process, the final implementation especially, awareness of how a system is going to be implemented can assist the control logic selection process. If the implementation requirements are bound by memory and processing speed constraints, designing a controller that is mathematically or storage-wise intensive might not meet the final requirement. Likewise, the control logic process being chosen must be capable of being implemented using industry-standard tools and not call for more-than-necessary additional component/system development.

Figure 4.6 summarizes the parameters that influence the control logic selection process.

4.3.4 Control Design Process

A detailed methodology has been proposed in Chapter 2 that summarizes the process involved in the early stages of a project. However, from a control-system
specific standpoint, the design and development of control systems after selecting a suitable control logic involves multiple steps. This section focuses on a model-driven methodology for the design of such systems. Once the control logic is chosen, this is approached with the usage of a suitable modeling environment, Simulink being the case throughout this thesis. The development of the control model is always done with the plant model as the reference during the early stages of development, as this best offers the ability to see how the system responds and make changes right-away.

If the control model were to be built and converted to software code and then flashed to a controller and then tested with the physical system upon every design iteration, the process would become completely time-consuming and the whole model-based approach would be rendered useless. Before starting to use the modeling environment, it is always helpful to approach the control system again as sub-systems and identify the structure of the controller at the higher level, with its components as blocks.
While the outer layer of the controller has to be designed such that it takes in inputs and outputs signals as defined earlier, different control logics may result in changes to this I/O structure as the design process evolves. Capturing these changes and modifying the identified I/O structure to reflect the same is important, as this would otherwise impact the overall system later during implementation. Details pertaining to designing for implementation are discussed in detail in a later section.

4.4 Electrical System Optimization problem - Control Logic Selection and Design process

4.4.1 Introduction

For the electrical system optimization problem, system analysis clearly showed that the existing control was not utilizing the battery. The existing control strategy, henceforth referred to as the VR (Voltage Regulation) is conservative, as it does not utilize the battery except during cranking the engine. The electrical system optimization problem can be approached from an energy perspective, with the alternator and battery being the energy sources, while the electrical loads are the sinks. With this approach, significant parallels can be drawn between this system and a typical Hybrid Electric Vehicle Energy management problem. As a consequence of this perspective, energy management strategies applicable to HEV control can be considered, and in particular, the Equivalent Consumption Minimization Strategy (ECMS) is identified as a potential solution to this problem since it reduces a global minimization problem to a realizable, local problem. Figure 4.7 shows a schematic of the electrical system of a conventional vehicle in comparison to the hybrid electric vehicle system.
Figure 4.7: Conventional Vehicle Electrical System and Parallel HEV Architectures
4.4.2 ECMS

The ECMS was first proposed by Paganelli et al. as a supervisory energy management strategy for charge-sustaining HEVs. This strategy has been implemented as a real-time strategy in some applications and is extensively used in multiple domains [4, 3] provides significant background on ECMS along with other references. One important observation though to be made at this stage is that the regular ECMS strategy requires a calibration parameter better known as the equivalence factor that is drive-cycle dependent. The equivalence factor is what relates battery energy to the fuel energy and is done so to capture the efficiency penalty involved with replacing in future the energy now extracted. Since a single equivalence factor cannot serve the purpose for all driving conditions, the ECMS is suited for real-time control only in case of the cycle for which it has been tuned.

4.4.3 A-ECMS

The adaptive ECMS (A-ECMS) is the same strategy as discussed above, however with an adaptive equivalence factor, which is near-optimal and takes into consideration changing driving conditions. Several methods have been employed to make the equivalence factor adaptive including drive cycle prediction, driving pattern recognition, and SOC feedback [18]. Based on research by the authors in [18], SOC feedback has been found to offer a slightly suboptimal solution in comparison with a tuned ECMS controller. In case of SOC feedback, the equivalence factor is updated at regular intervals to reflect the battery’s SOC, with penalties and constraints in place to ensure the SOC is within limits.
From the perspective of thesis, the development and implementation of the A-ECMS is used as an example for the forthcoming sections.

Figure 4.8 shows the A-ECMS control system design methodology, based on work by [3].

Figure 4.8: A-ECMS Control System Design methodology

- The current split factor is defined: \( \gamma = [0,1] \in \mathbb{R}^a \).
- The \( \gamma \) vector is multiplied by \( I_{alt,max} \) to give a set of potential solutions ranging from not using the alternator to providing the maximum alternator current.
- Only those current splits that do not violate the component limitations and system constraints are selected for further examination.
- The remaining vector of viable alternator currents are subtracted from the load current demand to give a corresponding vector of battery currents.
- Alternator fuel consumption and battery equivalent fuel consumptions are calculated.
- The alternator and battery equivalent fuel consumptions are summed.
- The alternator current corresponding to the minimum total fuel consumption is selected.
- An autoregressive moving average model for the equivalence factor is executed at regular intervals to make the controller adaptive.

Figure 4.9 shows the A-ECMS controller along with the plant model upon completion of modeling using Simulink, and figure 4.10 shows the controller model alone in detail with its sub-models.
Figure 4.9: A-ECMS Control System Simulink Model - overall view

Figure 4.10: A-ECMS Control System Simulink Model - subsystem view
4.4.4 Requirements traceability

The importance of a dedicated Requirements Management tool has been highlighted in detail in Chapter 2. While requirements are identified at early stages of the project and documented, followed by specifications upon further interpretation, information about these are retained only in documents that are eventually not accessed unless required. This presents the following disadvantages

1. The models are built just based on the designer’s interpretation of requirements/specifications. There is no accountability when this process involves multiple teams/members working on different sections of the model.

2. Requirements are constantly changing entities, especially in today’s fast paced development scenario. In cases of projects that run over few years, the case is often that the requirement in the final year is significantly different in comparison to the requirements identified in the inception stages. While the high level requirements to do not change, more specific details are the ones that cause problems when left untraced. These changing requirements are not necessarily updated in all models, especially when multiple models cater to these requirements.

3. In projects that are in collaboration with external partners, information pertaining to requirements stays only in the initial document, which is seldom shared, in comparison to models which are extensively shared and modified.

4. Since there is no traceability that exists between models and the requirements, changes that are often implemented in a model to ensure something also results
in failure to comply with a requirement, which does not present itself until
extensive validation, resulting in unnecessary rework at later stages.

5. Finally, a holistic approach towards the development process calls for regular
validation against requirements, to keep the effort lost in rework minimal and
to ensure a successful product development. This cannot be done unless the re-
quirements are captured in the model itself or there exists a platform that allows
linking the models with the requirements they satisfy. Requirements traceabil-
ity is an approach that ensures that the models that are built as a part of the
system development process are linked to the requirements that were identified
in the initial phases. This is facilitated by many modeling environments, and
in this case, Simulink.

For a research organization such as CAR, while a dedicated RM tool might not al-
ways be necessary, requirements that are documented as a regular text document or
spreadsheet can still be linked to the model and its components. Simulink allows
linking various sub-models that are developed in its modeling environment to specific
requirements that are documented either using a dedicated RM tool or a simple doc-
ument such as a Microsoft Word document or Microsoft Excel Spreadsheet. It also
supports a variety of other formats. More specific details can be found in Mathworks’
Documentation on Requirements Traceability. Linking requirements to the model
that was built is not a one-way process. Links to Individual objects/blocks or even
signals in Simulink can be generated and captured in the requirements document. By
doing this, once the requirements document is opened, the associated model and spe-
cific sections of the model that cater to every single requirement can be accessed from
one location. Moreover, multiple links can be generated if there are sections that cap-
ture multiple requirements and vice-versa. Requirements traceability offers multiple
benefits, primarily addressing the various problems listed earlier. From the over-
all perspective, requirements traceability results in better accountability and a more
holistic approach towards building models while simultaneously ensuring throughout
the development process that the requirements are completely captured in the model.

Figure 4.11 shows the requirements document as an Excel Spreadsheet for the
electrical system optimization problem with links to specific sections in the model
that meet them. Also highlighted in the model is one example block that meets the
highlighted requirement in the spreadsheet.
Figure 4.11: Requirements Traceability - Spreadsheet and Simulink Model

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Link to Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Achieve improvement in baseline Fuel Economy (EPA FTP Cycle)</td>
<td>Simulink model</td>
</tr>
<tr>
<td>2. Optimize Electrical System energy usage</td>
<td>Simulink model</td>
</tr>
<tr>
<td>3. Meet all electrical load requirements</td>
<td>Simulink model</td>
</tr>
<tr>
<td>4. Cause no harm &amp; protect the customer even during a failure</td>
<td>Simulink model</td>
</tr>
<tr>
<td>5. Maintain battery voltage within safe limits</td>
<td>Simulink model</td>
</tr>
<tr>
<td>6. Maintain battery temperature within safe limits</td>
<td>Simulink model</td>
</tr>
<tr>
<td>7. Do not completely drain the battery/prevent the car from starting</td>
<td>Simulink model</td>
</tr>
<tr>
<td>8. <strong>Do not discharge/charge the battery at a very rapid rate</strong></td>
<td>Simulink model</td>
</tr>
<tr>
<td>9. Not affect battery life/quality over its useful lifetime</td>
<td>Simulink model</td>
</tr>
</tbody>
</table>
Chapter 5: Model-Based Control: Verification and Validation Phase

5.1 Introduction

Since the basics of MiL, SiL, PiL and HiL have already been explained in Chapter 2, the focus of this section is more on the actual implementation of the designed Electrical System controller and the various Verification and Validation processes that were done before implementing the same using rapid prototyping tools. MathWorks tools are extensively used in the various Verification and Validation processes described in this chapter, as the electrical system optimization problem taken as an example was modeled and eventually implemented using these tools. While the examples are application specific, a lot of these verification and validation processes are platform independent and fundamentally aim towards the same purpose of meeting specifications and hence the identified requirements.

5.2 Model-in-the-Loop

In the initial phases of the control system design and modeling process, it is desirable to be able to test the model as it evolves into a complete system. When modeling of the control is done in the same environment as the plant model, it becomes
easier to ensure that the controller model design process is on-track. Since simulations in the modeling environment are not real-time and run offline, depending on the processing capability of the simulating system, processes can be run much faster and results observed immediately. This is important in the early stages of design, where more than accuracy, being able to develop better models by means of re-iterative design that depends on observing simulation results is more important. Figure 5.1 shows the results of MiL simulations of the A-ECMS over 5 consecutive FTP cycles at a medium load condition.

5.2.1 Model Coverage

Testing for model coverage is a very useful process that can be done right at the MiL stage, to establish test cases and their effectiveness. Doing this at this phase will help define test cases that can be used all the way downstream the verification and validation process. Simulink’s Model Coverage tool is taken as an example to illustrate the model coverage test. The following analysis metrics are available as a part of the model coverage tool offered by Simulink.


2. Decision coverage: analysis of items such switches, state blocks (Stateflow) that represent decision points in a model.

3. Condition coverage: analysis of blocks such as the logic block and Stateflow transitions where the output is a logical combination of the inputs.
Figure 5.1: A-ECMS MiL results: 5 consecutive FTP cycles with medium electrical loads
4. Modified condition/decision coverage (MC/DC): analysis of safety-critical software and determine whether the logical inputs have independently changed the output.

5. Lookup table coverage (LUT): measure of the frequency of usage for each interpolation interval. (Execution of each interpolation and extrapolation interval at least once corresponds to full coverage under a test case.)

6. Signal range coverage: measure of the maximum and minimum values generated during simulation by each block output and for all Stateflow data objects.

7. Signal size coverage: measure of the minimum, maximum, and allocated size for all variable-size signals in a model. Only blocks with variable-size output signals are included in the report.

8. Simulink Design Verifier coverage: record of model coverage data for the Simulink Design Verifier blocks and functions.

Extensive model coverage testing can help identify blocks/sections of the model that are not being used and eliminate redundant sections or unused sections. Moreover, results of model coverage can help identify potential errors in the design process at a very high-level, when a section that is intended to perform at a particular condition fails to do so. For the example under consideration, model coverage was run to check primarily for decision coverage, and results presented here are illustrative of the model run over a single test case. To achieve complete model coverage, extensive test cases need to be generated. Figure 5.2 shows results from a sample model coverage check done on the A-ECMS model over a single FTP cycle.
5.3 Automatic Code Generation

The next major transformation that a control system under design in the modeling environment undergoes is to software code. Automatic code generation facilitates the transformation of models built in the modeling environment into equivalent C code for deployment in an embedded application. This approach is convenient and because it is automatic, faster and less error-prone than hand-written code. However, extensive verification and validation needs to be done on the code generated automatically to ensure that no information was lost in the code generation process that might lead to a risk or undesired performance during operation. The following section highlights some software code testing approaches.
5.3.1 Software code testing approaches

Figure 5.3 illustrates various software testing approaches, that draw inspiration from the software development industry and can still be used to test automotive control system software.

![Software Code Testing Approaches Diagram]

---

Figure 5.3: Software Code Testing Approaches

5.4 Software-in-the-Loop

Software-in-the-Loop simulation helps verify the automatically generated code, with the control system running as compiled code, while the plant is retained as a model. SiL basics have already been explained in Chapter 2. From the perspective
of the electrical system optimization problem, SiL testing was done to see how the generated code performs in comparison to the MiL results. Figure 5.4 shows SiL vs MiL results for battery SoC and current for a particular test case done using the ECMS controller (developed before making it adaptive).

![FTP Cycle – High loads: SiL vs MiL](image)

Figure 5.4: SiL vs MiL - ECMS Results comparison

5.5 Virtual and Rapid Prototyping

5.5.1 Virtual Prototyping

Virtual prototyping is an extension of the previous methodologies including MiL and SiL. However, virtual prototyping, as the name suggests is the process of developing the system more closer to reality, and includes the following

1. Embedded control software platform software (I/O drivers, operating system, etc.) and application software (functions for control and monitoring).

2. Plant model driver, vehicle and environment.
Virtual prototyping is done much earlier in the Verification and Validation process and happens in parallel with the design process itself. In case of large teams working on separate functions of the development process, this enables collaboration between the developers and simulation teams at a very early stage. While at the outset the virtual prototyping process appears no different from regular modeling environment simulations, the following advantages make virtual prototyping worth consideration depending on the application and availability of information.

1. Virtual prototyping allows pre-calibration, depending on available plant model information.

2. Since virtual prototyping does not happen in real-time, detailed and elaborate simulations can be run while still retaining the entire framework of platform and application layers, which is not the case in regular MiL.

3. Virtual prototyping allows running simulations at any desired rate, constrained only by processing capabilities. Both slow-motion as well as fast-forward simulation speeds can be achieved, in addition to real-time equivalent execution capability.

4. Finally, virtual prototyping at early stages helps build models and define systems well suited for rapid prototyping and eventually final implementation. This saves a lot of time and effort that otherwise goes towards the end of control design in translating from the modeling environment to the prototyping environment.
Virtual Prototyping was not done in case of the electrical system optimization problem. The A-ECMS controller was extensively tested in the modeling environment and using SiL techniques followed by rapid prototyping due to time constraints.

5.5.2 Controller Calibration

Calibration is the process of tuning and optimization (following implementation) of complex functions and control algorithms for various automotive operating scenarios and platforms in microprocessor-based control units. The objective of the tuning procedure is the vehicle-specific adaptation and optimization of the control functions of an electronic control unit. The control function algorithms are permanently stored in the control unit program; only the parameter values (maps, curves, and characteristic values) can be modified.

Calibration is an important process in the control system development cycle, as it helps in adapting the control system being built to suit the environment and platform. An emulator test probe (ETK) is installed in the control unit for the parallel application. The ETK can replace either the control unit program and the calibration data or the calibration data only, depending on the design. Figure 5.5 indicates in brief the working of ETK in tandem with an ECU and how measurement and calibration can be done using INCA, a tool from ETAS.

In case of the A-ECMS controller, there was no significant need for extensive online calibration. However, the option of being able to calibrate the control system real-time helped defining different starting conditions as well as equivalence factors on-the-fly and evaluate the performance of the controller.
5.5.3 Rapid Control Prototyping

OSU Setup

Figure 5.6 shows the rapid prototyping setup at OSU, along with the Town and Country mini-van. ETAS hardware and software are used in this case to run the developed control system in real time with actual inputs from the vehicle’s electrical system and the output of the system controlling the alternator. A detailed step-by-step procedure to rapid prototype a controller from a Simulink model is explained in Appendix B.

Figure 5.7 shows the A-ECMS controller model prepared for rapid prototyping.

The process of setting up the control system for code generation and hence rapid prototyping is shown in figure 5.8.

Following the above process in Simulink, the whole rapid prototyping process is done with the help of INTECRIO - IP (Integrated Platform), a tool from ETAS that facilitates Rapid Control Prototyping. The steps involved in the prototyping process are not explained in this section. Figure 5.9 shows the INTECRIO Rapid Prototyping
Figure 5.6: OSU Rapid Control Prototyping Setup for Alternator Control

Figure 5.7: A-ECMS Standalone controller for Rapid Prototyping
Figure 5.8: A-ECMS Standalone controller setup procedure

1. Isolate the controller from the plant model and create a separate initialization file with all control parameters declared as parameters in the MATLAB workspace.

2. The controller has to run in discrete time steps when run in the rapid prototyping environment. All continuous blocks are accordingly modified wherever applicable.

3. The controller is setup as a single block at the highest level with all inputs and outputs at the same level - there can be multiple sub-systems.

4. All I/O signals are named and corresponding Simulink objects are created and saved as a separate workspace file that is loaded every time the model is opened.

5. For all the I/O signals, the signal ranges are assigned and the storage class for these signals is declared as exported global. Calibration parameters are defined as global parameters.

6. The required embedded target is chosen under Simulink Coder configuration options.

7. The system is built and the required '*.SIX' and '*.A2L' files are generated.
Environment and Figure 5.10 illustrates the Experiment Environment, which is the interface used when testing the controller in the vehicle.

![INTECRIO Integration Platform for the A-ECMS Implementation](image)

Figure 5.9: INTECRIO Integration Platform for the A-ECMS Implementation

### 5.5.4 Rapid Prototyping - Results

There are two stages once the controller is implemented in the Rapid Control Prototyping system.

**Verification of the Implementation process**

The objective of this stage is to identify and verify that the code running on the RCP system matches against MiL and SiL results in open loop. To facilitate this, upon implementation of the system on the vehicle, a dry run of the controller
Figure 5.10: INTECRIO Experiment Environment for the A-ECMS Implementation
was done without closing the loop. i.e. with the alternator disconnected from the ECMS controller. In this scenario, the system was still reading inputs from the vehicle and processing the same while providing an output based on those. The subject of interest in this case is not what the output is, but if this output matches against the controller performing offline. To check this, all inputs and outputs of the controller were recorded during this dry run. This recorded set of inputs were next fed to the MiL model and the outputs observed and compared against the real-time results. Following this, the loop was closed and the performance of the controller in the simulation environment as well as the experiment environment were compared.

Figure 5.11 show the performance of the implemented controller in comparison to the MiL for a the same inputs (with inputs for MiL drawn from experimental data).

It is important to remember the fact that the output of MiL is dependent on the quality of the plant model and hence differences in performance may exist between the above two cases. However, significant anomalies were none and hence, it is safe to assume that the controller was running on the Rapid Prototyping system successfully. Figure 5.12 show the working of the A-ECMS controller on the van over an FTP cycle, starting at different initial conditions.

**Validation of the the controller**

The motivation behind implementing the controller in a test vehicle is to be able to observe the controller’s performance in a real-life scenario and compare it against the existing control strategy. However, before proceeding to this step, the baseline case needs to be established. Upon initial testing of the A-ECMS controller on the van in the chassis dynamometer, significant variations in the fuel economy figures
Figure 5.11: Implemented A-ECMS controller in comparison to MiL
Figure 5.12: A-ECMS Testing: Charge sustaining performance over 1 FTP Cycle
were observed. The following observations were made and accordingly steps were taken to address few issues.

1. The external conditions during testing in the chassis dynamometer were monitored and it was observed that the battery was operating at a temperature much higher than what the maps were calibrated for.

2. There were fluctuations observed in the load current though no external loads were manually turned on/off - this was found to be because of the engagement/disengagement of the radiator fan. To address this, the radiator fan was controlled and set to run at a fixed speed throughout the driving cycle to ensure that the engine was always within operating temperatures.

3. Variations in fuel economy figures with deviations from the actual drive cycle were observed. A detailed analysis of this is still underway and a statistical analysis of the baseline controller’s performance has to be first established. Since the expectations from the A-ECMS strategy in terms of fuel economy improvement lie in the 1-3 % range, it is important to establish the fuel economy variation within the baseline case first on changes in driving patterns. If this exceeds the possible ECMS benefit in fuel economy, more extensive and controlled test procedures would be required to validate the controller.

Subject to the baseline fuel economy variations lying within acceptable limits, a detailed test plan was worked out to test the A-ECMS controller against the baseline VR. Figure 5.13 illustrates the testing plan in place at OSU to validate the controller. A realistic load variation schedule is also under design, with different electrical loads in the vehicle to be turned on at different intervals when testing both the baseline
as well as the A-ECMS control strategies. Figure 5.14 shows the framework for this electrical load schedule.

Figure 5.13: A-ECMS Testing: Test Plan

Rapid prototyping is not the end process of implementation of control. While realization of model-based systems in a physical plant is the first step towards actual experimental validation, the code generated for rapid prototyping is not the final release. The following sections address the various steps that follow in brief.

5.6 Hardware-in-the-Loop

Hardware-in-the-Loop techniques for verification and validation have been explained in detail in Chapter 1. From the perspective of the example of electrical system optimization problem, the developed ECMS controller might be tested at
### ELECTRICAL SYSTEM LOADING SCHEDULE

<table>
<thead>
<tr>
<th>Lights</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AC/blower</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rear defroster</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heated Seats</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(further split if</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>required)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entertainment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC/aux power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>outlet</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.14: A-ECMS Testing: Electrical System Loading Schedule
later stages before integration into the complete vehicle ECU or post-integration to ensure that all safety constraints are met. HiL was not done at OSU for this particular controller and the steps involved towards preparing the controller for HiL are similar to the Rapid Prototyping process in many aspects. However, the most important advantage of using HiL for extensive testing is the capability of running tests over extended durations and over wider battery operating scenarios. While the rapid prototyping environment offers more close-to-reality results, HiL can be used to evaluate test cases that would not be feasible in the real environment, while still running the model in real-time.

5.7 Final System Integration and Release

From the perspective of automotive controls development within the scope of an organization such as OSU-CAR, the process typically ends with handing over of control code and associated sub-systems (if any). While this process appears simple, the integration can be the most difficult part depending on the complexity of the system. For a complete mechatronic system design, the design of the mechanical and electrical systems are as important as the control - harmonious integration of all these sub-systems is what constitutes a system that would meet all requirements. Post-integration effort might include extensive recalibration of the control system to suit the platform it is being integrated with and so on. The integration of the developed electrical system control with other sub-systems and the parent controller environment will be done at Chrysler and is beyond the scope of this document.
6.1 Introduction

This case study is aimed at explaining the concept selection process that led towards the final vehicle architecture to be used in the EcoCAR2 competition. The focus is on how a model-based simulation tool can be used to make high-level decisions early up in the stream within short time-frames while ensuring a systematic approach towards concept selection. The methodology described here forms a part of a larger report that was generated by the team for the competition [19].

6.1.1 The EcoCAR2 Competition

The Ohio State University is one of fifteen North American universities participating in EcoCAR 2: Plugging into the Future, an Advanced Vehicle Technology Competition (AVTC) headline-sponsored by the United States Department of Energy (DOE) and General Motors (GM). Spanning over three years, the primary technical goals of the EcoCAR2 competition are to construct and demonstrate vehicles and powertrains that (in comparison to production gasoline vehicles):

- Reduce fuel consumption
• Reduce well-to-wheel greenhouse gas emissions

• Reduce criteria tailpipe emissions

• Maintain consumer acceptability in the areas of performance, utility, and safety.

Each team's task is to design, build, and optimize a new powertrain for their vehicle, resulting in a fully functioning prototype vehicle by the end of the third year. The prototype vehicles created by each team are evaluated over a wide range of metrics that include consumer acceptability, fuel economy, emissions, drivability, acceleration and braking.

During the first year of the competition, teams will use computer simulation tools to compare different types of vehicle powertrain architectures and select a design for their vehicle. They also used Computer-Aided Design (CAD) software to decide how to package their new powertrain design into the General Motors donated 2013 Chevrolet Malibu.

At the beginning of Year 2, teams will receive their General Motors-donated 2013 Chevrolet Malibu Ecos, remove the stock powertrain, and work on integrating their own powertrain components into the vehicle. This requires teams to do mechanical, electrical, and controls work to make sure their components were securely and safely mounted in the vehicle and functioning properly to enable the vehicle to drive.

By Year 3, teams will have built their vehicles and the focus turns to refinement and optimization. During this year, teams work on making the vehicles more efficient by efforts such as: weight-reduction, reducing accessory loads, refining control strategies and software, aerodynamic improvements, etc. Teams also make their vehicles
show-room ready by making improvements to the vehicle interior, exterior, and driver interface.

6.2 Concept vehicle architecture selection process

6.2.1 Architecture Selection

The Ohio State team had approximately two months to select the hybrid vehicle architecture and powertrain components they would use in their vehicle to meet the EcoCAR 2 competition deadlines. Although the team had a very short amount of time to select their vehicle architecture, it was important to follow a rigorous design process that would ensure the hybrid vehicle architecture they selected would meet the goals set by their team for acceleration, fuel economy and a variety of other parameters.

The team broke their design process down into four main steps as shown in figure 6.1:

- High level decisions
- Brainstorm candidate hybrid architectures
- Simulate candidate concepts using Autonomie
- Select the final vehicle architecture

Figure 6.1: OSU EcoCAR2 vehicle architecture selection process
The focus of this case study is the usage of model-based simulation tools to evaluate the candidate concepts, enabling the final vehicle architecture selection process.

In the beginning phase of a project/system development, information is often scarce and benchmarking multiple concepts against each other becomes difficult unless a common platform is identified. Model based design provides an efficient methodology involving the four key elements in any development process including modeling a plant (from first principles or system identification), synthesizing and analyzing a controller for the plant, simulating the plant and controller together, and programming/deploying the controller. When model based simulation tools with pre-defined datasets that represent various physical systems are used, they can help simulate complex systems and obtain an initial set of results that are representative enough of the system. When comparing different concepts, different vehicle architectures in this case, what is important is a modular platform that provides enough flexibility to modify different parameters without having to develop models from scratch. Although the relevance of such a tool is directly dependent on the availability of existing models of systems, a comprehensive package with the freedom to change parameters of the sub-systems and see how it impacts an architecture is sufficient at this level of the development process. Since a complete vehicle architecture is to be compared, it is not sufficient that the plant is modeled. A model of the controller that is consistently performing across all architectures is desired - a great control on a bad architecture could perform better than a great architecture with bad control, rendering the selection process outcome invalid. Hence, identifying and selecting the controller model for all architectures calls for equally significant attention when compared with the architecture model.
6.2.2 Autonomie

Autonomie is a platform that has been designed by Argonne National Labs to be used as a single tool throughout the different phases of Model Based Design of the Vehicle Development Process (VDP) and is based on the Powertrain System Analysis Toolkit (PSAT). Autonomie integrates all the phases of model based design and provides a common framework throughout the entire design process. Autonomie offers a physical-model based plug and play model integration environment with hierarchical layered organization of the models, making it easy to understand. Capable of simulating sub-systems, systems, a combination of systems or entire vehicles, Autonomie has an extensive database of modular models for propulsion architectures ranging from conventional powertrains to plug-in hybrid electric vehicles (PHEVs). The most important advantage of using such a flexible tool is the ability to mix-and-match models of different levels of abstraction for execution efficiency with higher fidelity models where analysis and high-detail understanding is critical, and choose not to do so when only a overview is desired. Autonomie can also be integrated with commercial off-the-shelf software applications such as GT-Power, AMESim etc when more detailed physical models are required. Figure 6.2 explains how Autonomie works.

The following capabilities of Autonomie have been put to use to enable rapid vehicle architecture selection for the EcoCAR2 competition:

1. Mix and match different sub-systems to realize the desired candidate vehicle architecture.

2. Modify existing libraries to suit the vehicle under consideration.

3. Predict and analyze fuel efficiency and performance over multiple drive cycles.
Figure 6.2: Autonomie - simulation process
4. Measure component parameter variation/robustness.

6.2.3 Modeling Assumptions

A critical step in modeling any system is to make assumptions. Due to the difficulty of constructing an exact mathematical model of a given system, empirical relationships and constant values are often utilized to simplify models, represent dynamic states, and model low-sensitivity parameter in order to expedite the simulation process. The vehicle model used to conduct simulations in Autonomie in this case makes many such assumptions. The largest working assumption is that all of the component data, models, and controls provided are accurate. Thermal effects are assumed to be negligible, which is not the case in an actual vehicle but is the case for the model used in this selection process. For example, the transmission oil, engine oil, and engine coolant temperatures in this model are assumed to remain constant throughout the given drive cycle, which would not be the case in a real-world scenario. It is well known that the temperature of components can cause significant changes in powertrain component behavior. Driveline dynamics and many component dynamics are both either ignored or greatly simplified and this is considered to have no effect on the results of the simulation. While there are many assumptions, these simplifications are essential to the speed with which simulations can be run. A detailed study on model sensitivities, assumptions and limitations has been highlighted in a report generated as a part of the competition’s requirements[19].

6.2.4 Baseline vehicle architecture

The conventional gasoline Chevrolet Malibu served as the baseline vehicle for all concepts under consideration. The baseline model was created in Autonomie
by modifying an existing conventional vehicle model. The specifications of the base vehicle were input to the model. The powertrain consisted of a 133 kW 2.4 L 4-cylinder SIDI engine and a six-speed automatic transmission with a final drive ratio of 2.89. The electrical accessory load on the vehicle was set to a constant 200 Watts and the road load was also modified to match that of the Malibu. Finally, the mass was set to match the stock vehicles curb weight at 1564 kg. Once all of the components had been added to the skeleton model and the operational parameters were set, the control strategy for each component was implemented into the model. The components and their control strategies were added to the model by specifying their file path in the initialization file callback for each component. Ideally, all sub-models and the overall vehicle model should be validated against appropriate measured data. Due to the pre-production status of the 2013 Malibu, this validation was not possible. Despite this, the model is expected to yield good results, particularly when used to evaluate sensitivities. Figure 6.3 shows the flow of Power through the Autonomie Chevy Malibu Model.

**Series E-REV**

![Figure 6.3: Flow of Power through the Autonomie Chevy Malibu Model](image-url)
6.2.5 Candidate concept vehicle architectures

Preceding the concept selection process, four different hybrid architectures were identified as candidates. The thought process behind the identification of these candidate solutions involved both internal brainstorming sessions as well as results from an external technology search. The identification of candidate solutions is a detailed process by itself and is beyond the scope of this case study. The following assumptions were made before proceeding with the concept selection process, and doing so worked in favor of a comprehensive design evaluation of the candidate architectures. These assumptions are backed by sufficient reasoning, as explained in the work done in [21].

1. Each of the four hybrid vehicle architectures were assumed to utilize the same engine (and emissions treatment system) - A Honda 1.8L high compression ratio engine that was converted to run on E85 during the EcoCAR competition.

2. Each of the four hybrid vehicle architectures were assumed to utilize the same battery pack - the largest A123 systems pack available, since the pack energy is an important factor in deciding the all-electric range which has a significant impact on projected team scores on dynamic events.

3. The amount of tailpipe emissions for each vehicle models were not simulated, since tailpipe emission models are generally inaccurate unless high quality experimental data exists to calibrate them.
Series E-REV

The series E-REV architecture (Figure 6.4) consists of an electric machine coupled directly to the 1.8L Honda engine in the front and another electric machine coupled to the rear axle via a two speed transmission. The front machine is a 75 kW machine and acts as a generator during charge sustaining mode. The rear electric machine is a 145 kW and is the primary drive for the vehicle in both charge depleting and charge sustaining modes. The rear electric machine is capable of taking power from the A123 battery to propel the vehicle and can collect energy through regenerative braking.

Pre-Transmission E-REV

The pre-transmission parallel hybrid architecture (Figure 6.5) includes an engine, an electric machine, a battery pack and an automated manual transmission. The 1.8L Honda engine is clutched directly to an 82 kW electric machine, which is then
Figure 6.5: Pre Transmission Parallel E-REV

connected to the 5 speed automated manual transmission with gear ratios: 3.45, 1.94, 1.29, 0.97 and 0.75. A final drive then connects the transmission to the front wheel axle. The automated manual transmission and the final drive used in this architectures power-train use gear ratios optimized to meet the different cycle requirements. The gear ratio of 3.63 that was used for the final drive was found through an optimization analysis. The electric machine is capable of taking power from the A123 battery to propel the vehicle and can collect energy through regenerative braking.

Post-Transmission E-REV

The post-transmission parallel hybrid architecture(Figure 6.6) includes an engine, an electric machine, a battery pack and an automated manual transmission. The 1.8L Honda engine is clutched directly to the 5 speed automated manual transmission. A final drive then connects the transmission to the front wheel axle. The 150kW electric machine is also connected to the final drive The final drive and the gear ratio was
optimized to 3 and [0.5, 8.382, 2.5, 1.38, 1.06] respectively. The electric machine and engine are made to run efficiently to obtain a good acceleration figure of 0-60mph in 7.1 sec. The electric machine is capable of taking power from the A123 battery to propel the vehicle and can collect energy through regenerative braking.

**Pre+Post-Transmission EREV**

The pre+post-transmission parallel hybrid E-REV powertrain architecture(Figure 6.7) includes an engine, two electric machines, a battery pack and an automated manual transmission. The 1.8L Honda engine is clutched directly to the 82kW front electric machine, which is then connected to the 6 speed automated manual transmission. A final drive then connects the transmission to the front wheel axle. The automated manual transmission and the final drive used in this architectures front powertrain use the same gear ratios found in the Honda Civic SI Sedan. The transmission and final drive found in this vehicle are compatible with the 1.8L Honda
engine the team has already decided to use. The rear powertrain contains a second 82kW electric machine and a single final drive gear that is then connected to the rear wheel axle. The gear ratio of 3 that was used for the final drive was found through an optimization analysis. Both electric machines are capable of taking power from the A123 battery to propel the vehicle and can collect energy through regenerative braking. The configuration also allows limited series operation with the transmission engaged in neutral.

6.2.6 Powertrain Simulation Approach

The approach the Ohio State team used to simulate each of the four hybrid vehicle architectures is shown in figure 6.8.

The team started by performing a top level energy analysis on the 505 cycle, US06 city cycle, HWFET cycle, US06 highway cycle, 0-60mph acceleration test and gradeability cycle to determine the top-level power and torque requirements for the
Figure 6.8: Candidate architectures - simulation process approach
vehicle. Once these requirements were known, the team could estimate the initial sizes for the electric machines in all four hybrid vehicle architectures, since the size of the engine and battery pack were already fixed. A rough estimate of the weight of each vehicle was also made at this point, since the general components in each vehicle were known. Once the vehicle models were assembled, the process of optimizing the vehicles started. Each of the vehicles had a different set of free parameters to optimize, such as electric motor size, gear ratios, and control parameters. Generally, the optimization process was to maximize fuel economy while maintain a 0-60 time of 7.2 +/- 0.2 seconds. The choice of 0-60 time was driven by the customer needs and wants (i.e. better performance than stock) the value was also compared against other vehicles in the Malibus class. After the models were optimized, a more refined estimate of each of the vehicle weights was made and a final set of fuel economy and acceleration drive cycles were run. In this way, the final simulation results were generated and used to evaluate each of the hybrid vehicle architectures in a decision matrix.

6.2.7 Top-Level, Energy-Based Performance Calculations

A basic preliminary analysis of vehicle propulsion requirements was conducted to assist in general component sizing and selection. This included calculating the instantaneous power and torque required at the wheels, factoring a small trailer with the Semtech emissions unit that will be pulled along in the competition during testing, electric range capability and gradeability. This analysis is explained in detail in the work done by [21] as a part of the vehicle concept selection process to determine
the approximate initial starting parameters for each of the candidate vehicle architectures. The results from the basic overall propulsion calculations helped improve component sizing and selection before moving into the simulation stage. Raw baseline specifications were formed from the same and utilized when evaluating simulation results from the candidate vehicle architectures.

6.2.8 Candidate architectures modeling using Autonomie

The models for each of the four hybrid vehicle architectures were built in Autonomie using the pre-built vehicle models whenever possible. For the pre+post transmission hybrid, the OSU simulation team modified an existing model template and controller to get the desired functionality. The initialization files which contain model parameters in Autonomie used for both the engine and battery models were modified to reflect the characteristics of the 1.8L E85 engine and the largest available A123 battery pack. These models were based on data generated in OSUs lab for the engine and data made available from A123 in the case of the batteries. For the electric motors, suitable models already available in Autonomie were used. This was motivated by the fact that these components had not been selected yet, thus, no specific manufacturers and models had been selected at this point in the design cycle. All models also used the same 200W electrical accessory load, had no mechanical accessory loads, and used weights that were determined by the weight analysis that was provided by the packaging team. The simulations for each of these architectures were run at these estimated weights, and it was determined that the pre-transmission parallel with the electric motor on the rear axle preformed the best. This further
validated the architecture selection. Each model also used the same wheel and chasis initialization files as the conventional Malibu vehicle model. All the models used many of the same controller parameters and initialization files for components like the starter, accessories, power converters, wheels and chassis to ensure that the fuel economy and acceleration numbers from each model could be compared against each other on an even playing field. Additional information on the model, initialization, pre-processing and post-processing files used in each vehicles Autonomie model can be found in Appendix [A]

6.2.9 Simulated Energy and Performance Results

Each of the four hybrid vehicle models and the conventional Malibu model were optimized over the available design space based on the process described above. The fuel economy for each vehicle was determined by running the 505, HWFET, US06 City and US06 Highway drive cycles in both charge sustaining and charge depleting mode. The fuel and electrical consumption from those drive cycles were then combined into a single weighted fuel economy number. The gradeability of each vehicle was also determined on a pass/fail basis, since that is the way it is scored at competition. Figure [6.9] shows the simulation results for different architectures.

All four hybrid vehicle architectures had significantly better fuel economy numbers than the Conventional Malibu. These fuel economy numbers ranged from 63.5mpgge for the pre-transmission E-REV to 72.3mpgge for the pre-post-transmission E-REV vehicle. All the hybrid vehicles also had all-electric ranges greater than 44 miles, which gave then utility factors of 0.654 or greater. The pre+post-transmission E-REV had the longest all-electric range of 50 miles. In addition to the good fuel
Figure 6.9: Fuel Economy and Performance Simulation Results of candidate architectures
economy and all-electric range numbers, most of the vehicles had better acceleration
times than the conventional vehicle. The pre-transmission E-REV and the post-
transmission E-REV both had 0-60mph acceleration times of 7.1 seconds, which was
less than the conventional vehicles 0-60mph acceleration time of 8.2 seconds. The
post-transmission E-REV had the best 50-70mph acceleration time of 3.2 seconds.
The series E-REV had poor acceleration times overall due to a controls strategy
problem with the Autonomie model. A 50-70mph acceleration time was not included
for the conventional vehicle since the results of Autonomie models 0-60mph acceler-
ation test did not match the Chevy Malibus actual 0-60mph acceleration time of 8.2
seconds. All vehicles were also able to pass the gradeability test.

6.2.10 Performance of the selected architecture

Figure 6.10 shows the performance of the selected architecture under Charge De-
pleting and Charge sustaining modes over a US06 Cycle.

6.2.11 Impact of Component Sizing

With engine size and battery pack energy fixed, the only major component sizing
issues remaining are the electric motor(s) in each of the architectures. Because of
the design specifications, the sizing of these electric machines was directly addressed
in the simulation process. All of the architectures studied were full-function electric
vehicles, thus, the performance of the vehicle is strongly influenced by motor selection
whereas fuel efficiency is largely fixed due to the single option for engine selection.
Because of this, the following approach was used to size components:

1. develop initial component sizes based on top-level, energy-based analysis
Figure 6.10: Simulation performance results of Pre+Post Transmission parallel E-REV architecture
2. conduct simulation and refine controls to achieve best possible vehicle operation

3. if acceleration targets are not met, revise electric motor sizes to achieve desired targets (i.e. increase motor power for better acceleration or reduce electric motor to reduce acceleration.)

As described above, the issue of component sizing was reduced drastically by pre-selecting components based on analysis conducted upstream of the simulation process, as shown in [19]. This drastically reduced the need to search a high degree of freedom design space which can be costly. This is particularly true when one considers the importance of control strategy on HEV performance. Any time a major component is changed the control also needs optimized. Because of the limited design space, the only component left to freely adjust was the electric motor sizes. As described above, this was the design parameter used largely to meet acceleration targets. The following figures show the result of a full sweep of electric motor sizes on the pre-transmission hybrid vehicles fuel economy and 0-60mph acceleration. These plots show that increasing motor size leads to shorter 0-60mph acceleration times and a decrease in utility factor weighted fuel economy. Similar motor sizing studies were also done for the other three hybrid vehicle architectures being considered by the Ohio State team. All three of those studies exhibited the same trends in fuel economy and acceleration when the electric motor size was varied.

Figure 6.11 shows the impact of Electric Motor Size on Pre-Transmission Hybrid Vehicle Fuel Economy.

Figure 6.12 shows the impact of Electric Motor Size on Pre-Transmission Hybrid Vehicle 0-60mph Acceleration.
Figure 6.13 provides a comparison of Pre-Transmission Hybrid Vehicles Acceleration and Fuel Economy with Different Electric Motor Sizes.

![Motor Sizing vs Fuel Economy Graph](Image)

Figure 6.11: Impact of Electric Motor Size on Pre-Transmission Hybrid Vehicle Fuel Economy

### 6.2.12 Limitations of using Autonomie

Autonomie has a number of limitations that are common among this class of simulation tools. It should be noted that with these limitations are generally a result of trade-offs for benefits, such as simulation speed and simplicity. The limitations discussed in this thesis include: 1) Low model fidelity; 2) Reliance on empirical data; 3) Autonomie is only a longitudinal model; and 4) Emission models have low accuracy. For the purpose of this selection process, Autonomie is predicting basic vehicle energy consumption and longitudinal performance. This allows the models to be low-bandwidth, with limited dynamics. Although the sub-models in the model are based on physical effort and flow connections, the sub-models rely on empirically determined
Figure 6.12: Impact of Electric Motor Size on Pre-Transmission Hybrid Vehicle 0-60mph Acceleration

Figure 6.13: Comparison of Pre-Transmission Hybrid Vehicles Acceleration and Fuel Economy with Different Electric Motor Sizes
component information rather than high-fidelity models. This lack of fidelity is not a flaw in the model, but an advantage that allows for rapid execution of the model at a modeling level that is consistent with the purpose of the investigation. Because of the reliance on empirical data on components, accuracy of the model is closely tied to the quality of the experimental data used to create the sub-model and how well these match the intended use of the model. Despite the accuracy limitations on the current generation of the stock vehicle model, the qualitative tradeoffs displayed above can be trusted to a high degree of accuracy. A direct result of this reliance on empirical data is that engine scaling files is only accurate when scaling an engine by +/- 20. This model is a longitudinal vehicle model and can only be used to estimate a vehicle's fuel economy or straight-line acceleration. The simulator does not have the required dynamic models of the driveline or of the suspension to simulate vehicle drivability and handling. At this stage in the vehicle development process, this is an acceptable limitation due to the complexity in implementing models that can meet these needs. The additional complexity of these models would not provide additional accuracy in a fuel economy prediction and would require significantly longer simulation times; therefore this is an acceptable trade-off. Finally, the map-based emissions models in Autonomie do not fully capture the dynamic behavior of real world emissions. Engine-out emissions are highly dependent on things like engine coolant temperature and the impact of engine transients on the engines feedback control system, neither of which are accounted for in Autonomie. Catalytic converters also require very complicated models to capture their dynamic effects, and these models generally to not fit into the simplified structure of this type of energy-based model.
6.2.13 Conclusions

The Ohio State EcoCAR 2 team had less than two months to simulate multiple hybrid vehicle architectures and select one of the architectures to design, build and test over the next three years in the EcoCAR 2 competition. Using Argonne National Labs Autonomie, four hybrid vehicle architectures and a conventional vehicle powertrain were simulated and compared. The team had to select which architecture would best meet the teams acceleration and fuel economy targets, and the results of the Autonomie simulations were used to compare the same from each of the four hybrid vehicle architectures. Following this, selection criteria like educational value and component packaging were also factored towards the end, and the team decided to design, build and test the parallel-series PHEV architecture. The usage of a model-based plug and play simulation tool helped the team compare these architectures at a high level without having to spend too much time modeling the same.
Chapter 7: Conclusions and Future Work

7.1 Conclusions

A Model-Based approach towards the design of control systems is an established industry practice and is followed at different levels by different organizations. However, for a research organization with strong academic background such as the Center for Automotive Research at The Ohio State University, a comprehensive process for automotive model-based control is needed. A process that encapsulates various practices and standards already in place in the industry can help develop better solutions when collaborating with industry as well as when working on exploratory research projects. While systematic approaches exist already, there are sufficient variations amongst internal approaches and methodologies which calls for a more unified approach that is industry-inspired. This thesis intends to capture to the best possible extent the above and present a comprehensive process that gives a developer not just a process to model and build a control system, but also to gain an overall perspective to the bigger problem. The proposed methodology starts right at the fundamental opportunity identification phase, and is driven in the early stages by product development methodologies. Systematic approaches towards identifying opportunities,
generating concepts and selecting concepts are presented in brief to set the background for the model-based development process. A separate case study on the usage of model-based simulation tools for concept selection in a scenario where minimal information is available is highlighted. Following the conceptualization phase, detailed model-based development processes are explained, with the example of an electrical system optimization problem. This includes modeling approaches towards the plant as well as the control and eventually, verification and validation techniques that ensure that the system is built to specifications and meets requirements identified. An Adaptive Equivalent Consumption Minimization Strategy (A-ECMS) used to address the electrical system optimization problem is implemented using Rapid Control Prototyping (RCP) tools and the control system is experimentally validated, and future recommendations made on implementation and testing. Observations are made on the overall process so far and recommendations are made from the perspective of standards and practices already in place in the industry, expertise in which will add significance value to an organization such as OSU-CAR.

7.2 Future Work

Future work that can be built upon the concepts and processes explained in this thesis are proposed from two different perspectives as follows:

7.2.1 Model-based approaches

Usage of Dedicated RM tools

The importance of requirements management and traceability has been explained in detail in 3 and 4. While small projects do not necessarily call for a dedicated requirements management tool and regular documentation along with traceability

129
links would suffice, a dedicated requirements management tool such as DOORS would help manage multiple projects across a variety of platforms. Moreover, industry projects in large scales use a dedicated RM tool and expertise in using them would be an added advantage when being considered as a solutions-provider.

**Modeling Standards and Guidelines Compliance**

The following standards exist in the automotive industry, and developing systems in compliance with these standards, depending on the requirements and implementation platform can work in advantage of a research organization such as CAR. Compliance with standards in the industry and systematic modeling approaches offer multiple benefits that include reduced lead times and better communication when developing in collaboration with the industry. Moreover, systems and models developed for a particular project can be reused across other platforms as well as environments with minimal changes.

**MISRA-C**

The Motor Industry Software Reliability Association (MISRA) is a collaboration between vehicle manufacturers, component suppliers and engineering consultancies which seeks to promote best practice in developing safety-related electronic systems in road vehicles and other embedded systems. MISRA-C guidelines were introduced to help assess the quality of software code (C code) used in electronic control units of automobiles and aircraft. MISRA’s aims include facilitating code safety, portability and reliability in the context of embedded systems, specifically those systems programmed in ISO C. Many tools exist for checking compliance with MISRA standards, Polyspace being one of them.
MAAB

The MathWorks Automotive Advisory Board (MAAB) was originally established to coordinate feature requests from several key customers in the automotive industry and now focuses on the usage and enhancements of MathWorks controls, simulation, and code generation products from Mathworks. Since Mathworks products are being extensively used to model and develop control systems, for an organization like CAR, conforming to industry standards pertaining to these tools can help build error-free automotive software. Extensive documentation exists with established guidelines for control algorithm Modeling using MATLAB, Simulink, and Stateflow [23].

AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) [24] is a worldwide development partnership of car manufacturers, suppliers and other companies from the electronics, semiconductor and software industry. AUTOSAR was conceived to promote innovative electronic systems that further improve performance, safety and environmental friendliness. The idea is to have a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation". AUTOSAR is a key enabling technology to manage the growing electrics/electronics complexity. Finally, such a standard allows the exchange and update of software and hardware over the service life of the vehicle. The main goal of AUTOSAR is to define standard modular software infrastructure for application and basic software, which allows exchanging parts of the systems software. This shall enable modularity, scalability, transferability and re-usability of software among projects, variants, suppliers, customers. With this background, awareness of this standard and following modeling
practices in compliance with AUTOSAR will again bridge the gap between development at a research organization setting such as CAR and the automotive industry.

7.2.2 Electrical system optimization problem

Rapid Prototyping using ETK Bypass

The current implementation of the A-ECMS in the Town and Country Van was done using a OSU-built PWM driver that was not linked to the actual ECU. The alternator was directly driven by this driver and disconnected from the vehicle ECU. While this was done considering time constraints and non-availability of direct access to the duty cycle command in the ECU, a better way of implementing this would be to bypass the ECU via software and control the duty cycle command through ETK bypass. This is better since the ECU’s original PWM driver is used and more accurate control of the alternator can be achieved.

Battery SoC Estimation using the EBS

The state of charge of the battery is one of the most important parameters that can impact the working of the electrical system control developed. The need for an accurate estimation of the Battery SoC is for two reasons

1. Ensuring the battery is operated within safe SoC limits, and is not discharged completely.

2. Facilitating better performance of the controller, as a wrong SoC prediction can result in poorer fuel economy in comparison to the baseline case.

In this regard, Chrysler has provided an EBS (Electronic Battery Sensor) module developed by Bosch [25], which can be directly attached to the negative terminal of the
battery. This module communicates via the LIN Protocol and provides information on the battery’s SoC, temperature, voltage and current. Using the EBS can help in better SoC estimation and hence a control system that is more accurate and robust.

**HiL with Actual Hardware (Battery)**

While verification and validation was done on the modeling environment and rapid prototyping environments, Hardware-in-the-Loop testing was not done for the electrical system control developed. While rapid prototyping offers realization of control in the actual vehicle, the chassis dyno test environment is not completely conducive for extended tests. Battery temperature has a significant role to play in the performance of the system and maintaining a constant battery temperature was found to be difficult in the chassis dynamometer setup at OSU. A hardware-in-the-loop setup with the actual vehicle battery in the loop and everything else modeled and running on dedicated hardware simulators can help test the system while retaining the battery at nominal temperatures. This can also help extend the spectrum of tests that can be done on the system, since the electrical loads can be more systematically controlled than in an actual vehicle. While the dynamics of these loads may not be completely captured in the HiL setup, the controller can be extensively tested under a variety of conditions.

**On-Road Testing**

Experimental validation of the A-ECMS control was done only on the chassis dynamometer and this meets the requirements in terms of performance in comparison to a baseline FTP cycle. However, testing in real conditions, such as on-road testing over extended distances can help achieve a better comparison against the baseline
VR strategy. Fuel economy figures when compared over large distances are more desirable than short distance fuel economy figures, as there are multiple uncertainties associated with driving over short distances.
Appendix A: Autonomie file tables
<table>
<thead>
<tr>
<th>Component</th>
<th>Series Hybrid File Name</th>
<th>Modified (Y/N)</th>
<th>Post-Transmission Parallel Hybrid File Name</th>
<th>Modified (Y/N)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vehicle Propulsion Controller (VPC)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Propulsion Controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>vpc_prop_ser_eng_load_following_tx_PHEV</td>
<td>N</td>
<td>vpc_prop_par_direct_par_lmot_posttx</td>
<td>N</td>
</tr>
<tr>
<td>Initialization File:</td>
<td>vpc_prop_ser_eng_load_following_tx_PHEV_init</td>
<td>Y - Battery SOC Control Limits</td>
<td>vpc_prop_par_direct_par_lmot_posttx_PHEV_init</td>
<td>Y - Battery SOC Control Limits</td>
</tr>
<tr>
<td><strong>Brake Controller</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>vpc_brake_ser_eng_p1</td>
<td>N</td>
<td>vpc_brake_par_posttx_lmot</td>
<td>N</td>
</tr>
<tr>
<td>Initialization File:</td>
<td>vpc_brake_ser_eng_p1_init</td>
<td>N</td>
<td>vpc_brake_par_posttx_lmot_init</td>
<td>N</td>
</tr>
<tr>
<td><strong>Vehicle Propulsion Architecture (VPA)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration:</td>
<td>vpa_ser_eng_2wd_p3_dm</td>
<td>N</td>
<td>vpa_par_2wd_p3_au</td>
<td>N</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>vpa_ser_eng_transmission_prepoc</td>
<td>N</td>
<td>vpa_par_posttransmission_prepoc</td>
<td>N</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>vpa_postprocess</td>
<td>N</td>
<td>vpa_lang_less_regen_postproc</td>
<td>N</td>
</tr>
<tr>
<td><strong>Chasis (chas)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plant</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>chas_plant_veh_equation_losses</td>
<td>N</td>
<td>chas_plant_veh_equation_losses</td>
<td>N</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>chas_plant_900_229_029_EcoCar2</td>
<td>N</td>
<td>chas_plant_900_229_029_EcoCar2</td>
<td>N</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>chas_postprocess</td>
<td>N</td>
<td>chas_state_postprocess</td>
<td></td>
</tr>
<tr>
<td></td>
<td>chas_state_scalar_equation_losses_postprocess</td>
<td>N</td>
<td>chas_sum_postprocess</td>
<td></td>
</tr>
<tr>
<td></td>
<td>chas_summary_postprocess</td>
<td>N</td>
<td>chas_state_scalar_equation_losses_postprocess</td>
<td>N</td>
</tr>
<tr>
<td><strong>Clutch (cpl)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>cpl_ctrl_amt_par</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>cpl_ctrl_amt_par_init</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plant</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>cpl_plant_clutch</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Electrical Accessories (acelec)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plant</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>acelec_plant_const_pwrloss_volt_in</td>
<td>N</td>
<td>acelec_plant_const_pwrloss_volt_in</td>
<td>N</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>acelec_plant_200</td>
<td>N</td>
<td>acelec_plant_200</td>
<td>N</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Energy Storage System (ees)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>ess_ctrl_generic_map</td>
<td>N</td>
<td>ess_ctrl_generic_map</td>
<td>N</td>
</tr>
<tr>
<td>Plant</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model:</td>
<td>ess_plant_generic_map</td>
<td>Y - Updated for A123 Data for 18.9kWh Pack</td>
<td>ess_plant_generic_map</td>
<td>Y - Updated for A123 Data for 18.9kWh Pack</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>ess_plant_li_6_7_5_salt</td>
<td>Y - Updated for A123 Data for 18.9kWh Pack</td>
<td>ess_plant_li_6_7_5_salt</td>
<td>Y - Updated for A123 Data for 18.9kWh Pack</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>ess_plant_preproc</td>
<td>N</td>
<td>ess_plant_preproc</td>
<td>N</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>ess_postprocess</td>
<td>N</td>
<td>ess_state_scalar_postprocess</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>ess_state_signal_postprocess</td>
<td>N</td>
<td>ess_state_signal_postprocess</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>ess_summary_postprocess</td>
<td>N</td>
<td>ess_summary_postprocess</td>
<td>N</td>
</tr>
</tbody>
</table>

Figure A.1: Series hybrid and post-transmission parallel hybrid model files(1)
<table>
<thead>
<tr>
<th>Final Drive 1 (fd)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant</td>
<td>fd_plant_map_trigloss_funTW</td>
<td>fd_plant_map_trigloss_funTW</td>
</tr>
<tr>
<td>Model:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>fd_plant_444_accord</td>
<td>Y - Optimized Gear Ratio</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>fd_summary_postprocess</td>
<td>N</td>
</tr>
<tr>
<td>fd_state_scalar_postprocess</td>
<td>fd_postprocess</td>
<td>fd_postprocess</td>
</tr>
<tr>
<td>fd_state_signal_postprocess</td>
<td>fd_state_scalar_postprocess</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Final Drive 2 (fd2)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gearbox (gb)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>ctrl_with_dmd_trs</td>
<td>gb_ctrl_dmd_accel_pedal</td>
</tr>
<tr>
<td>Model:</td>
<td>gb_ctrl_dmd_amt_motveh_spd_accel_pedal</td>
<td>gb_ctrl_dmd_n_gen_eng not included</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>gb_ctrl_dmd_no_cpl_init</td>
<td>N</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>gb_plant/au_map_trigloss_funTwin</td>
<td>N</td>
</tr>
<tr>
<td>gb_plant/au_map_trigloss_funTwin</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>gb_plant/au_map_trigloss_funTwin</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>gb_plant/au_map_trigloss_funTwin</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generator (gen)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>ctrl_with_cstr_trs_cmd</td>
<td>gb_postprocess</td>
</tr>
<tr>
<td>Model:</td>
<td>cstr/ctrl_generator_map_trq_in</td>
<td>gb_postprocess</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>gb_postprocess</td>
<td>N</td>
</tr>
<tr>
<td>gb_postprocess</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>gb_postprocess</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mechanical Accessory (acmech)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant</td>
<td>acmech_plant CONST_PWR Loss</td>
<td>acmech_plant CONST_PWR Loss</td>
</tr>
<tr>
<td>Model:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Initialization Files:</td>
<td>acmech_plant_0</td>
<td>N</td>
</tr>
<tr>
<td>Pre-Processing Files:</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Post Processing Files:</td>
<td>acmech_postprocess</td>
<td>N</td>
</tr>
</tbody>
</table>

**Figure A.2: Series hybrid and post-transmission parallel hybrid model files(2)**
<table>
<thead>
<tr>
<th>Vehicle Propulsion Controller (VPC)</th>
<th>Pre-Transmission Parallel Hybrid</th>
<th>Pre+Post Transmission Parallel Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong>: vpc_prop_par_direct_par_1mot_pretx</td>
<td>N</td>
<td>y: Propulsion Controller Stateflow to use 2nd motor</td>
</tr>
<tr>
<td><strong>Initialization File</strong>: vpc_prop_par_direct_par_1mot_pretx_PHEV_init</td>
<td>Y: Battery SOC Control Limits</td>
<td>y: Battery SOC Control Limits</td>
</tr>
<tr>
<td><strong>Brake Controller</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: vpc_brake_par_pretex_1mot</td>
<td>N</td>
<td>y: Brake Controller Stateflow</td>
</tr>
<tr>
<td><strong>Initialization File</strong>: vpc_brake_par_pretex_1mot_init</td>
<td>N</td>
<td>Y: Added Regen Braking for Second Motor</td>
</tr>
<tr>
<td><strong>Vehicle Propulsion Architecture (VPA)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Configuration</strong>: vpa_par_2wd_p2_dm</td>
<td>N</td>
<td>vpa_par_2Rwd_p2_dm_dmc</td>
</tr>
<tr>
<td><strong>Pre-Processing Files</strong>: vpa_par_posttransmission_prep求</td>
<td>N</td>
<td>vpa_par_postprocessing</td>
</tr>
<tr>
<td><strong>Post Processing Files</strong>: vpa_postprocess</td>
<td>N</td>
<td>vpa_postprocessing</td>
</tr>
<tr>
<td><strong>Chassis (chas)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Plant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: chas_plant_veh_equation_losses</td>
<td>N</td>
<td>chas_plant_veh_equation_losses</td>
</tr>
<tr>
<td><strong>Initialization Files</strong>: chas_plant_000_299_eng02_EcoCar2</td>
<td>N</td>
<td>chas_plant_000_299_eng02_EcoCar2</td>
</tr>
<tr>
<td><strong>Pre-Processing Files</strong>: N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Post Processing Files</strong>: chas_postprocess</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Clutch (clp)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Controller</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: clp_clf_amt_par</td>
<td>N</td>
<td>clp_clf_amt_par</td>
</tr>
<tr>
<td><strong>Initialization Files</strong>: clp_clf_amt_par_init</td>
<td>N</td>
<td>clp_clf_amt_par_init</td>
</tr>
<tr>
<td><strong>Plant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: clp_plant_clutch_map</td>
<td>N</td>
<td>clp_plant_clutch_map</td>
</tr>
<tr>
<td><strong>Initialization Files</strong>: clp_plant_clutch</td>
<td>N</td>
<td>clp_plant_clutch</td>
</tr>
<tr>
<td><strong>Pre-Processing Files</strong>: N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Post Processing Files</strong>: clp_summary_postprocess</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Electrical Accessories (accel)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Plant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: accelec_plant_const_pwrloss_volt_in</td>
<td>N</td>
<td>accelec_plant_const_pwrloss_volt_in</td>
</tr>
<tr>
<td><strong>Initialization Files</strong>: accelec_plant_200</td>
<td>N</td>
<td>accelec_plant_200</td>
</tr>
<tr>
<td><strong>Pre-Processing Files</strong>: N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Post Processing Files</strong>: accelec_summary_postprocess</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Energy Storage System (ees)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Controller</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: ess_clf_generic_map</td>
<td>N</td>
<td>ess_clf_generic_map</td>
</tr>
<tr>
<td><strong>Plant</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong>: ess_plant_generic_map</td>
<td>Y: Updated for A123 Data for 18.9kWh Pack</td>
<td>ess_plant_generic_map</td>
</tr>
<tr>
<td><strong>Initialization Files</strong>: ess_plant_li_6_75_salt</td>
<td>Y: Updated for A123 Data for 18.9kWh Pack</td>
<td>ess_plant_li_6_75_salt</td>
</tr>
<tr>
<td><strong>Pre-Processing Files</strong>: ess_plant_preproc</td>
<td>N</td>
<td>ess_plant_preproc</td>
</tr>
<tr>
<td><strong>Post Processing Files</strong>: ess_postprocess</td>
<td>N</td>
<td>ess_postprocess</td>
</tr>
</tbody>
</table>

Figure A.3: Pre-transmission hybrid and Pre+Post-transmission parallel hybrid model files(1)
<table>
<thead>
<tr>
<th>Final Drive 1 (fd)</th>
<th>Plant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model: fd_plant_map_trqloss_funTW</td>
<td>N</td>
</tr>
<tr>
<td>Initialization Files: fd_plant_363_cavalier Y</td>
<td>Optimized Gear Ratio fd_plant_363_cavalier Y</td>
</tr>
<tr>
<td>Pre-Processing Files: N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Post Processing Files: fd_summary_postprocess fd_state_signal_postprocess fd_state_scalar_postprocess fd_postprocess</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Final Drive 2 (fd2)</th>
<th>Plant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model: N/A</td>
<td>fd_plant_map_trqloss_funTW N</td>
</tr>
<tr>
<td>Initialization Files: N/A</td>
<td>fd_plant_363_cavalier</td>
</tr>
<tr>
<td>Pre-Processing Files: N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Post Processing Files: N/A</td>
<td>fd_summary_postprocess fd_state_scalar_postprocess fd_postprocess</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gearbox (gb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
</tr>
<tr>
<td>Model: ctrl_with_dmd_trs gb_ctrl_dmd_amt_eng_vwh_spd_accel_pedal gb_ctrl_trs_amt_par_p2</td>
</tr>
<tr>
<td>Initialization Files: gb_ctrl_dmd_amt_eng_vwh_spd_accel_pedal gb_ctrl_trs_amt_par_p2</td>
</tr>
<tr>
<td>Pre-Processing Files: N/A</td>
</tr>
<tr>
<td>Post Processing Files: gb_summary_postprocess</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generator (gen)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
</tr>
<tr>
<td>Model: N/A</td>
</tr>
<tr>
<td>Initialization Files: N/A</td>
</tr>
<tr>
<td>Plant</td>
</tr>
<tr>
<td>Model: N/A</td>
</tr>
<tr>
<td>Initialization Files: N/A</td>
</tr>
<tr>
<td>Scaling File: N/A</td>
</tr>
<tr>
<td>Pre-Processing Files: N/A</td>
</tr>
<tr>
<td>Post Processing Files: N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mechanical Accessory (acemch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant</td>
</tr>
<tr>
<td>Model: acemch_plant_const_pow_loss</td>
</tr>
<tr>
<td>Initialization Files: acemch_plant_0</td>
</tr>
<tr>
<td>Pre-Processing Files: N/A</td>
</tr>
<tr>
<td>Post Processing Files: acemch_summary_postprocess acemch_state_scalar_postprocess acemch_postprocess</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motor 1 (mot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
</tr>
<tr>
<td>Model: ctrl_with_cstr_trs_cmd met_ctrl_cstr_map_Pelec_funTW_volt_in met_ctrl_cstr_trs_nostr_slip met_ctrl_rdm_map_Pelec_funTW_volt_in</td>
</tr>
</tbody>
</table>

Figure A.4: Pre-transmission hybrid and Pre+Post-transmission parallel hybrid model files (2)
Figure A.5: Pre-transmission hybrid and Pre+Post-transmission parallel hybrid model files (3)
Appendix B: Rapid Control Prototyping using INTECRIO

This section provides a step-by-step procedure to setup Rapid Control Prototyping (RCP) using ETAS INTECRIO, and implement the same using appropriate hardware. The example used throughout this process is an ECMS-based control strategy for alternator control, developed for the Chrysler Town & Country mini-van at OSU.

B.1 Simulink Model Configuration

The Simulink model of the control system developed is taken and the controller isolated from the plant, and saved as a stand-alone model, with the necessary initialization files, as shown in Figure [B.1]. The stand-alone controller should conform to the following - discrete time (fixed step size) model with no proprietary blocks or blocks from other Simulink toolboxes - only blocks that conform with Simulink RTW (Simulink coder) are to be used. It can be helpful to have the stand-alone controller model as a single block with multiple inputs and outputs (ports) at the highest level.

The following sequence of operations need to be performed on the model to configure it for INTECRIO:

- After adding all input, output ports and calibration variables to the models, right click each signal line before an output port and after an input port and
select Signal Properties to set the name of the line to match the corresponding port name. Figure B.2 shows the Signal Properties dialog.

- Select “Signal name must resolve to Simulink signal object” and close the dialog.
- Open the Model explorer and select the Base Workspace on the left panel. Create a Simulink signal for each signal named in the previous step, following the same naming convention.
- For each signal, define the following: Minimum, maximum and Units.
- Define the Storage class as Exported Globals for each signal.
- For each calibration variable, follow the same procedure as the previous steps - however, define the initial value as well, setting it to a constant. Figure B.3
Figure B.2: Signal Properties dialog - Simulink
shows the Model Explorer dialog and the Simulink Objects definition for an example variable.

![Model Explorer dialog](image)

Figure B.3: Model Explorer - defining Simulink Objects

- The MATLAB workspace now will have multiple Simulink Objects that correspond to the various inputs and outputs and calibration variables. Save the workspace to a .MAT file. Note: Any changes done to the input, output and calibration signals will require the workspace to be saved again.

- On the Model window, click on File → Model Properties and define under Callbacks → PreloadFcn this .mat file to be loaded. Close this dialog.
- Under Simulation → Configuration Parameters → Optimization, check the box for Inline Parameters and click on the Configure button. Under this dialog, add the names of all the calibration parameters under Global parameters.

- Click on Simulation → Configuration Parameters and select Solver on the left panel. Set the value of Type to Fixed-Step and the step size to the desired value. Figure B.4 shows the Simulink Solver configuration dialog.

![Simulink Solver Configuration](image)

Figure B.4: Simulink Solver Configuration

145
Click on Tools → Code Generation → Options. Set the System target file as irt.tlc. This defines the build target as INTECRIIO Real-Time Target. Figure B.5 shows the Simulink code generation configuration dialog.

To generate code that can be used with INTECRIIO, click on the Build button. Alternate method of building is to quit this dialog and with the model open, pressing Ctrl+B.

Figure B.5: Simulink Code Generation Configuration
During the build process, several files are generated - however, INTECRIO needs only the .six and the .a2l files.

Successful code generation is necessary to proceed further in INTECRIO. Typical errors are generally because of incompatible blocks or wrong configuration of the Simulink Objects. Once all errors are eliminated and the build is successful, the next step is to open INTECRIO.

B.2 INTECRIO Configuration

INTECRIO integrates code from various behavioral modeling tools, makes it possible to configure the prototype as well as a hardware system for Rapid Prototyping, and allows the generation of executable code. Finally INTECRIO provides an experiment environment for the execution of the Rapid Prototyping experiment. INTECRIO allows the user to generate binary files to be flashed onto the ES1000 system. The ES1000 system will then run independently with or without an experiment environment like INCA. This section contains the steps to configure and use the INTECRIO environment to build a project that allows interfacing the control software with the ES1000 hardware system.

B.2.1 Adding I/O Cards to Hardware System

- Open INTECRIO. Select File → New Workspace and enter a name and location for the workspace.

- On the panel to the left, right click Hardware Systems → Add Hardware Systems. Select ES1000 System → Insert. Click OK. Right click the ES1000 system → Insert Target and select ES1135 → Insert.
Figure B.6: INTECRIIO - Hardware System Configuration
• Right click on ES1135 → Insert and select each of the required cards that have already been installed on your ES1000 system. Repeat this procedure until all cards are added to the target system. Figure [B.6] shows the hardware configuration, with all the cards added to the hardware system.

### B.2.2 Configuring the 1232 ETK Card

• Expand on the left pane under Hardware Systems ES 1232. Right click on ETK Bypass → Import A2L...

• Select the A2L file corresponding to the vehicle’s ECU with ETK from a location on disk and click on Open. Different signal groups based on the sampling rates are now populated under ETK Bypass.

• Double click on ETK Bypass to open a window on the right pane with multiple tabs. Select the Signal Selection tab.

• The Signal Selection tab window has four sections. The top-left section shows the different sampling rates based signal groups on the Raster and the bottom-left section shows the signal hierarchy. The bottom-right section has the list of available signals, and the top-right section can be used to drag-drop the required signals.

• The available signals are those that can be measured directly from the vehicle’s ECU through the ETK. Select the desired signal group on the top-left section and select the required signals from the bottom-right section. Drag-drop the selected signals to the top-right section. Figure [B.7] shows the signal selection pane for the ES1232 card.
B.2.3 Configuring the 1303 A/D Card

- Double click the ES1303 card from the left panel and select on the right pane Signal Groups tab.

- Select either -10 -10V or -60+60V for each channel. Note that a branch called in appears under the selected channel. This input is the raw analog voltage reading from the channel. To convert the raw A/D reading into a meaningful engineering unit, click on the button under Formula and define the signal conversion. For non-linear relationships or more complex conversions, you will need to create the Simulink model along with all conversions from raw-voltage inputs depending on the sensors.
Once all channels are configured, disable the unused channels by right clicking on the channel name in the left pane and un-checking the option Enabled. Figure B.8 shows the configuration dialog for the ES1303 card.

**B.2.4 Configuring the 1310 D/A Card**

- Double click the ES1310 card from the left panel and select on the right pane Signals tab.

- Name the output signals as desired and specify the units, formula for conversion/scaling and the minimum, maximum and initial values of the output voltage. Similar to the ES1303 card configuration, if the output signal from the controller is not a voltage within the range of the ES1310 card, suitable scaling might be required. Figure B.9 shows the configuration dialog for the ES1310 card.
B.3 Implementing an INTECRIIO System

B.3.1 Software System configuration

- Right click on Modules under Software in the left pane and click on Import Module.

- Browse the .six file that was generated using Simulink Coder as discussed in the previous section.

- The model now appears under Modules, and upon expansion shows names of all the inputs and outputs of that model.

- Right click on Software Systems → Create Software System and enter a name for the software system.
Once the software system shows up under Software Systems, drag and drop the imported module onto the software system that was just created.

Double click on the software system to open up a window in the right pane with a block representing all the inputs and outputs of all the modules.

Right click on the block → Map Unconnected Ports to Software System. Every I/O is assigned an input and output port which can be used to connect to the hardware system. Figure B.10 shows the software system with all the ports after configuration.

B.3.2 Overall System Build

The final stages of using INTECRIIO is the overall system build, which involve interfacing the hardware and software systems. Right click on Systems → Create System and enter a name for the overall system. Drag and drop the desired software systems into this.
• Similarly, drag and drop the ES1000 system, which is the hardware system, into the overall system.

• Double click on the overall system to open up a block with the software system inputs and outputs on the right pane.

• Expand the ES1000 system → ES1135 target under the overall system and then each hardware card to be used and then drag and drop the channels to be interfaced into the right pane.

• Each hardware channel can be connected to the software system by drawing a line between them or using the connection mapping tool by right clicking anywhere and selecting Open in Connection Window. Once in the connection window, select the Hardware signal on the left and the software signal on the right → Click on the double arrowed line to establish a connection.

• After completing all the connections, double click on OS Configuration to open a window in the right pane.

• Right click on OS Configuration→ Auto Mapping. This has to be done everytime any changes are done to the hardware (or) software systems. Figure B.11 shows the overall system connected and ready for the final build.

• Once the above steps are properly done, the final step is the actual build process. Right click on the overall system in the left pane → Build.

• If no errors were found during the build process, an .a2l file and .cod file shall be created in the Results folder or the INTECRIIO project.
B.4 Configuring the Experiment Environment

- In INTECRIO, click on Experiment→Open Experiment. This launches the Experiment Environment, an interface that can be used to download the generated code to the simulation target and run experiments.

- The Left pane automatically loads the workspace corresponding to this INTECRIO project. The right pane has three tabs - Instrumentation, Datalogger and Signal Generator. Under the Instrumentation tab, different instruments can be configured and used to display/monitor the variables while the controller is running.

- In order to assign a signal/variable to an instrument, drag-drop the desired signal from the left pane to the right pane under Instrumentation. A dialog appears...
with options to choose the instrument type as well as the group (if they need to be grouped).

Figure B.12: INTECRIO - Instrumentation pane of Experiment Environment

- Following the same procedure, the desired interface for signal monitoring can be created in the Experiment Environment as shown in Figure B.12.

- Calibration variables can be changed in real-time and need to be assigned to suitable edit-boxes or sliders.

- Once this is configured, the laptop with the Experiment Environment can be taken to the vehicle test setup. Connect this to the ES1000 system via Ethernet.

- On the top pane below the menu, select the “Download” button to download the generated code to the ES1135 simulation target.
• When the test-setup is ready, click on the "Start Simulation" button to start execution of code in the Simulation system. Click on the "Start Measurement" button to start measuring the variables and doing this starts displaying values on the instruments in real-time.

• The Datalogger pane can be used to configure the data recording when a real-time simulation is performed. Triggers can be setup to automatically start or stop recording. Figure B.13 shows the Datalogger pane of the Experiment Environment.

Figure B.13: INTECRIIO - Data Logging in Experiment Environment

• Upon completion, press the "Stop Measurement" button, followed by the "Stop Simulation" button and finally, "Disconnect" to disconnect the computer from the ES1000 system.
Bibliography


[19] “EcoCAR2 - Y1 Report”. OSU EcoCAR2 team, Center for Automotive Research, The Ohio State University, Columbus, Ohio.


