META II: FORMAL CO-VERIFICATION OF CORRECTNESS OF LARGE-SCALE CYBER-PHYSICAL SYSTEMS DURING DESIGN

Serdar Uckun

Palo Alto Research Center

SEPTEMBER 2011
Final Report

Table of Contents

Section **Page**

List of Figures ..... **ii**
List of Tables ..... **iii**
1.0 Summary **1**
2.0 Introduction **2**
2.1 Impact Statement 3
3.0 Methods, Assumptions, and Procedures **4**
3.1 The META-II Tool Chain and Design Workflow 4
3.2 Component Library 6
3.3 Design Space Exploration 6
3.4 Functional Failure Analysis 9
3.5 Functional Verification 10
3.6 Performance Verification 10
3.7 PoC and PCC Computation 13
3.8 Reliability Analysis 14
3.9 System Integration 14
4.0 Results and Discussions **16**
4.1 ADAPT Electrical Power System Example 16
4.1.1 Domain Description 16
4.1.2 Models 18
4.1.3 Findings 22
4.2 IFV Ramp Example 25
4.2.1 Domain Description 25
4.2.2 Models 26
4.2.3 Findings 28
5.0 Conclusions **32**
6.0 References **33**
Appendix A: Project Team **34**
List of Acronyms and Abbreviations **39**

List of Figures

Figure **Page**

Figure 1. The Model-Based System-Engineering Process Flow 5
Figure 2. An Example Graph Grammar Rule for an EPS 8
Figure 3. A Design Concept Generated for the EPS Domain 8
Figure 4. Results of an FFA Fault Propagation Simulation on an EPS Design 9
Figure 5. PoC Computation for an EPS Model with Respect to Two Different Requirements 12
Figure 6. The Advanced Diagnostics and Prognostics Testbed (Courtesy of NASA Ames Research Center) 17
Figure 7. The ADAPT Testbed Structure (Picture Reproduced from [10]) 17
Figure 8. The EPS Modelica Performance Components Library Structure 20
Figure 9. Modelica EPS Components 20
Figure 10: Type of Modelica models generated by the Automated Model Generator 21
Figure 11. Automatically generated Modelica performance with redundant power sources. 22
Figure 12. Automatically generated Modelica performance model with one power source and multiple loads. 22
Figure 13. Phase 1 Output Distribution from MCS. 25
Figure 14. Phase 2/3 Output Distribution from MCS. 25
Figure 15. A BAE Bradley Fighting Vehicle (Courtesy of BAE Systems) 25
Figure 16. The Modelica Model of a Rear Ramp System 26
Figure 17. The Electrical Power Subsystem 27
Figure 18. The Control Subsystem 27
Figure 19. The Mechanical Subsystem 28
Figure 20. The Crew Subsystem 28
Figure 21. Example of Speed Deviation Requirement 30
Figure 22. Hierarchical Sensitivity Analysis 31

List of Tables

Table **Page**
Table 1. Results for the ADAPT EPS: Design Phase 1. 23
Table 2. Results for the ADAPT EPS: Design Phase 2 and 3. 23
Table 3. Performance Requirements for the Rear Ramp System 26
Table 4. Model Stochastic Inputs..... 29
Table 5. Probability of Correctness for the Requirements..... 30

1.0 SUMMARY

The complexity of modern defense systems is growing constantly. New technologies create opportunities for higher levels of integration. Modern systems such as air and ground vehicles contain a larger number of components that interact with each other in non-linear and often unpredictable ways. Unintended interactions lead to unexpected behaviors and consequences, some of which have proven to be catastrophic. A key technical challenge in developing such complex systems is to ensure that catastrophic subsystem and component interactions are well understood and contained prior to full-scale development.

To address these challenges, The Defense Advanced Research Projects Agency (DARPA) is investing in novel methods for design and verification of complex systems. The META program (META not an acronym but is typically spelled using all capital letters by DARPA) is specifically aimed at compressing the product development and deployment timeline of complex defense systems through model-based design and manufacturing. Using the META design paradigm, different component model libraries can be used to instantiate, analyze, and verify a system design independent of its physical manifestation. The goal is to establish a "correct-by-construction" design prior to detailed design and prototyping.

Under the META program, a team led by the PARC (Palo Alto Research Center) team is developing a model-based system-engineering framework that enables architectural analysis of complex systems during the conceptual design phase. Using this framework, design teams can systematically explore architectural design decisions during the early stage of system development prior to the selection of specific components. The analysis performed at this earliest stage of design facilitates the development of more robust and reliable system architectures. This report provides a summary of the work conducted by the PARC team during the course of the one-year project.

2.0 INTRODUCTION

The complexity of modern defense systems is growing constantly. New technologies create opportunities for higher levels of integration. Modern systems such as air and ground vehicles contain a larger number of components that interact with each other in non-linear and often unpredictable ways. Unintended interactions lead to unexpected behaviors and consequences, some of which have proven to be catastrophic. A key technical challenge in developing such complex systems is to ensure that catastrophic subsystem and component interactions are well understood and contained prior to full-scale development.

To address these challenges, DARPA is investing in novel methods for design and verification of complex systems. The META program is specifically aimed at compressing the product development and deployment timeline of complex defense systems through model-based design and manufacturing. Using the META design paradigm, different component model libraries can be used to instantiate, analyze, and verify a system design independent of its physical manifestation. The goal is to establish a "correct-by-construction" design prior to detailed design and prototyping.

For mission-critical design applications, a key consideration is the ability of the designed system to meet specified requirements. Ideally, designers aim to converge on an architecture with a high probability of meeting design requirements as early as possible during the design phase. Availability of high-fidelity models, simulation frameworks, and other analytical tools greatly simplifies this process. Arguably, design automation is well advanced in certain other fields such as integrated circuit (IC) design. Whether concepts and methods from IC design could be adapted to the design of electromechanical systems is a subject of much debate in the design community [1, 2]. A recent approach named Platform-Based Design aims to increase the level of automation in the design of complex electromechanical systems by introducing multiple levels of abstraction [2]. Nevertheless, design complexity remains a significant challenge in the design of electromechanical systems, often resulting in substantial delays and cost overruns in the design of military and aerospace systems.

Under the META program, a team led by PARC is developing a model-based system-engineering framework that enables architectural analysis of complex systems during the conceptual design phase. Using this framework, design teams can systematically explore architectural design decisions during the early stage of system development prior to the selection of specific components. The analysis performed at this earliest stage of design facilitates the development of more robust and reliable system architectures. This report provides a summary of the work conducted by the PARC team during the course of the one-year project.

One quantitative goal of the META program is to compress the system design, development, test, and evaluation (DDTE) cycle by a factor of 5x or more. The PARC-led META-II project supports this goal by: 1) identifying design defects early in the design process using design-stage models and abstractions; and 2) reducing the time and effort required for

system verification through hardware-in-the-loop testing using co-verification of hardware and software at the design stage. Design-stage analyses help identify system-level and component interaction problems early in the DDTE cycle and thus prevent costly redesigns during later design or deployment phases.

2.1 Impact Statement

We believe that a systematic approach that identifies design defects, uncertainty, and unforeseen fault propagation effects during the early design process will compress DDTE schedules significantly. Desired improvements of >5x are within reach if the methods developed under the META program are widely adopted in the defense community. Much of the complexity and cost of verification is mandated through complex policies such DO-254, DO-178B, and NASA's NPR 8705.2B (Human Rating Requirements). We envision that a successful conclusion to these programs would yield revolutionary new methods, tools, and processes that will help refine the rigid procedural requirements that impose substantial delays and cost overruns on every major aerospace program today.

During this project, a stated goal of the PARC team has been to generate and verify robust conceptual designs in an automated fashion in far less time than the manual system engineering approach. The impact of this achievement will be early elimination of design flaws, reduced need for hardware-in-the-loop testing, and compressed design cycle times.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 The META-II Tool Chain and Design Workflow

Design of complex systems for mission-critical applications is driven by the ability of the system to meet specified requirements. Typically, the performance of the system with respect to requirements is verified using costly and time-consuming tests of prototype hardware under realistic operational scenarios. Failure to verify performance requirements at this stage necessitates additional design/prototyping/test cycles and often results in cost and schedule overruns.

The PARC META effort is guided by three key insights about the traditional design process:

1. Traditional system design activities explore a very limited range of the overall design space, often informed by what has worked well in the past and what did not.
2. In traditional system design, detailed analysis of functional failures and fault propagation is performed late during the design process, long after the design team has – figuratively speaking – painted themselves into a corner.
3. There is ample information about component and system behavior and function available during the conceptual design process to initiate system verification.

Based on these key insights, the PARC-led team is developing an integrated design flow and verification tool chain. The PARC tool chain uses a component model library, design rules for the target domain, and system requirements as input. Using these inputs, the PARC tool chain automates the generation, evaluation, and verification of conceptual system models in a seven-step process:

1. A large number of candidate functional designs are generated using configurational requirements for the target system, a component model library, design rules, and a generative grammar. The design synthesis step ensures that the design space is adequately explored and no novel architectural solutions are left on the table early on in the process.
2. Candidate designs are evaluated with respect to robustness to known fault modes using a functional simulation. This step is the functional equivalent of a Failure Modes and Effects Analysis (FMEA) and it ensures that only the most robust designs are promoted to the final step.
3. Next, we use a set of system-level metrics (developed by other META performer teams) to select the strongest candidate designs. Applicable metrics include adaptability, complexity, and flexibility.
4. Selected candidate designs are verified using a functional verification process based on state-of-the-art model checking technology. Note that the functional

verification process may be applied to functions embodied in software as well as hardware.

5. A performance verification process utilizes detailed models of component behavior in order to compute the probability of correctness (PoC) of each candidate design with respect to a set of performance requirements.
6. A traditional reliability analysis is performed on the most promising candidate designs.
7. Information obtained from functional verification, performance verification, FMEA, and reliability analysis tasks are composed into a Probabilistic Certificate of Correctness (PCC) for the candidate designs.

Figure 1 illustrates the overall process and the major elements of the automated tool chain for model-based system engineering.

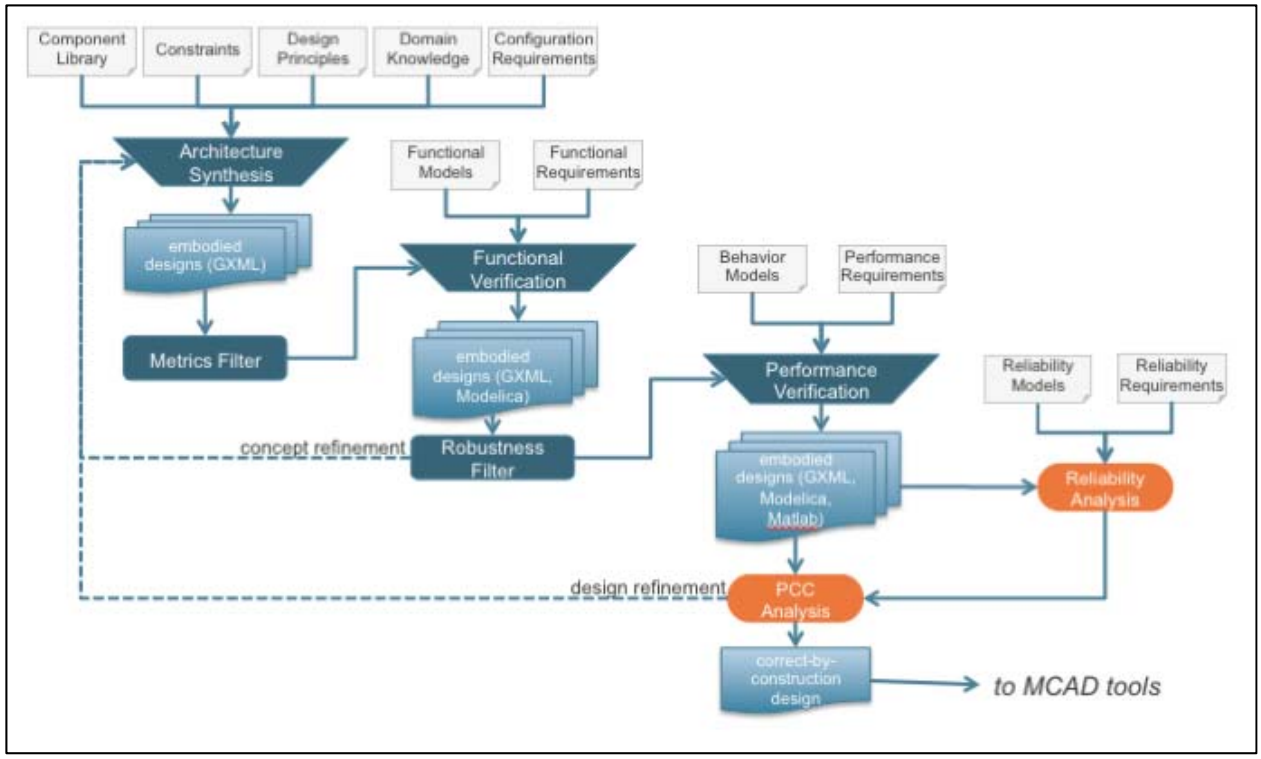


Figure 1. The Model-Based System-Engineering Process Flow

3.2 Component Library

A key ingredient of an automated tool chain for model-based system engineering is a component model library that is used to generate candidate designs for the target system. The component model library is question consists of functional and behavioral models, interfaces, and ranges of input, output, and state variables for representative components for the electrical power system (EPS) domain (of any other domain of interest). We chose the Modelica language to develop the component libraries for the META tool chain[1]. The EPS library provides Modelica models for the behavior and function of EPS components such as batteries, actuators, electrical switches, etc. For each component, the nominal behavior was modeled and augmented with the relevant failure modes. The nominal and the fault mode behavior of the EPS Library components were validated by comparing the simulated behavior of test models with measurements and sensor data from the Advanced Diagnostics and Prognostics Testbed (ADAPT) at NASA Ames Research Center.

For several components, we created models with different levels of accuracy. For example, the inverter component models range from very simple models that describe only the AC/DC power balance equation to models containing complicated electrical schematics including semiconductor components from the standard Modelica library for electrical systems. The reason for creating models of the same component with different levels of detail is to compare the performance of our tool chain during early stages of conceptual analysis versus later stages when component selection is almost final and more details are available regarding component performance and interactions. A recent article describes the EPS component model library in greater detail [3].

3.3 Design Space Exploration

The first step in the automated tool chain is to explore the space of feasible conceptual designs. As mentioned earlier, traditional system design activities explore a very limited range of the overall design space. Successful past designs often provide templates on which new designs are based. Consequently, many innovative yet unorthodox design alternatives often go unexplored. As an illustration, consider the variety and diversity of flying machine designs introduced during the first couple of decades of aviation. Over time, two templates (fixed wing with horizontal and vertical stabilizers, and rotary wing with a tail rotor) gained momentum and suppressed the exploration of alternatives for most new aircraft design projects. On occasion, we

[1] Modelica Association. (2011). *Modelica and the Modelica Association*. Available at https://http://www.modelica.org/.

see examples of innovative design appear in certain high-performance aircraft (e.g., canards, pusher props). A design space exploration process allows designers to explore the vast space of potential designs that meet configurational requirements for the system in a systematic fashion.

Recently, engineering design researchers have discovered that graph grammars provide a flexible yet ideally structured approach to the creation of complex engineering systems [4]. This interpretation of the design process makes graph grammars very suitable for computationally modeling the open-ended nature of conceptual design, where designers explore various ideas, decisions, and modifications to previous de- signs to arrive at feasible solutions.

In our approach, we use a graph-grammar-based design space exploration technique [3,4]. This generative technique takes user specified EPS loads as input and satisfies system-level configuration requirements to generate feasible EPS candidate architectures. For instance, if there were configurational requirements to provide reliable power to certain electrical loads, our approach would generate many feasible designs with alternative configurations for power generation (redundant and dissimilar sources), distribution (e.g., redundant buses), and switching.

A component model library serves as the backbone of our design space exploration approach [3]. Using the models in the model library as building blocks, this generative graph grammar based technique configures "correct by construction" EPS architectures that can be further studied by means of a simulation-based analysis. The graph-grammar-based configuration approach uses graphs as a representation scheme. This approach captures the transitions or the production rules for creating a solution, as opposed to storing the solutions themselves. The initial specification is represented as a simple graph in which the desired inputs and outputs are cast as arcs and nodes of the to-be-designed artifact. The evolution of a design from its inception to its final configuration can be viewed as a progression of graph transformations that lead to the final configuration.

Generating feasible EPS architectures using graph-grammar-based configuration is a two-step process. In the first step, we provide an EPS system design grammar to encode design rules for constructing EPS architectures. The rules are established prior to the design process. We have found that a 14-rule graph grammar is sufficient to generate feasible EPS architectures from multiple EPS requirements [3]. As an example, a prototypical EPS design requirement is to provide a battery relay for each battery circuit in order for the flight crew to isolate the battery from the rest of the EPS in case of a malfunction. Corresponding design rules govern the mapping of functional requirements to components, or the physical compatibility between EPS components. For instance, a design rule for the EPS domain would insert a battery relay for each circuit where a battery is present. Specifically, the design grammar encodes how specific system requirements can be embodied by selecting components from a full spectrum of electromechanical components represented in the component library.

In the second step, the graph transformation systems, or graph grammars, is invoked algebraically. Algebraic graph transformation methods rigorously define mathematical operations such as addition and intersection of graphs. A typical graph grammar rule is comprised of a left-hand side and a right-hand side (Figure 2). The LHS contains the preconditions that trigger the rule. Once the rule is triggered, the RHS describes the resulting graph transformation. By simply executing different combinations of grammar rules, a variety of feasible EPS architectures can easily be generated. Using rules to support component redundancy in the architectures, we are able to generate thousands of correct models using multiple starting seeds in a matter of minutes. Figure 3 illustrates a candidate design generated using a single seed and a graph grammar of 14 EPS-related rules.

Figure 2. An Example Graph Grammar Rule for an EPS

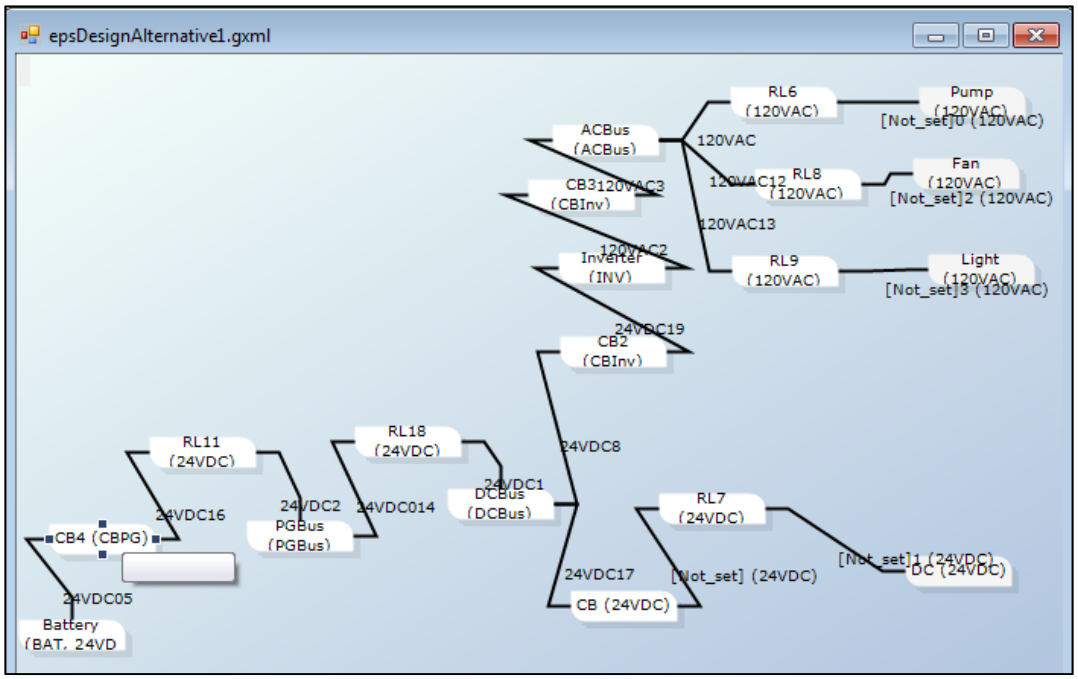


Figure 3. A Design Concept Generated for the EPS Domain

3.4 Functional Failure Analysis

Functional Failure Analysis (FFA) is a recent approach for coupling failure analysis with product design during the conceptual stage [5]. The FFA approach is based on the notion that a failure happens when a functional element in the system does not perform its intended task. For each function that is subject to failure, a functional criticality is defined depending on the role of functionality in accomplishing designed tasks. A simulation-based failure analysis tool is then used to analyze functional failures and reason about their impact on overall system functionality. Using this framework, design teams can systematically explore risks and vulnerabilities during the early (functional design) stage of system development prior to the selection of specific components. In essence, FFA provides early insight into the robustness of a design against known fault modes, much like the FMEA process that is often performed at the end of the design cycle.

Using FFA, a multitude of failure scenarios can be quickly analyzed to determine the effects of architectural design decisions on overall system functionality. In the META tool chain, each candidate generated during the first step is further analyzed using FFA. The tool chain automatically generates single, multiple, and cascading fault scenarios and measures the impact of each fault on overall system function. Figure 4 illustrates the result of a single simulation run over a candidate EPS design. Once the entire suite of functional loss scenarios is simulated, the tool chain generates an overall score for the candidate design. This score is correlated with the robustness of the candidate design with respect to known failure modes.

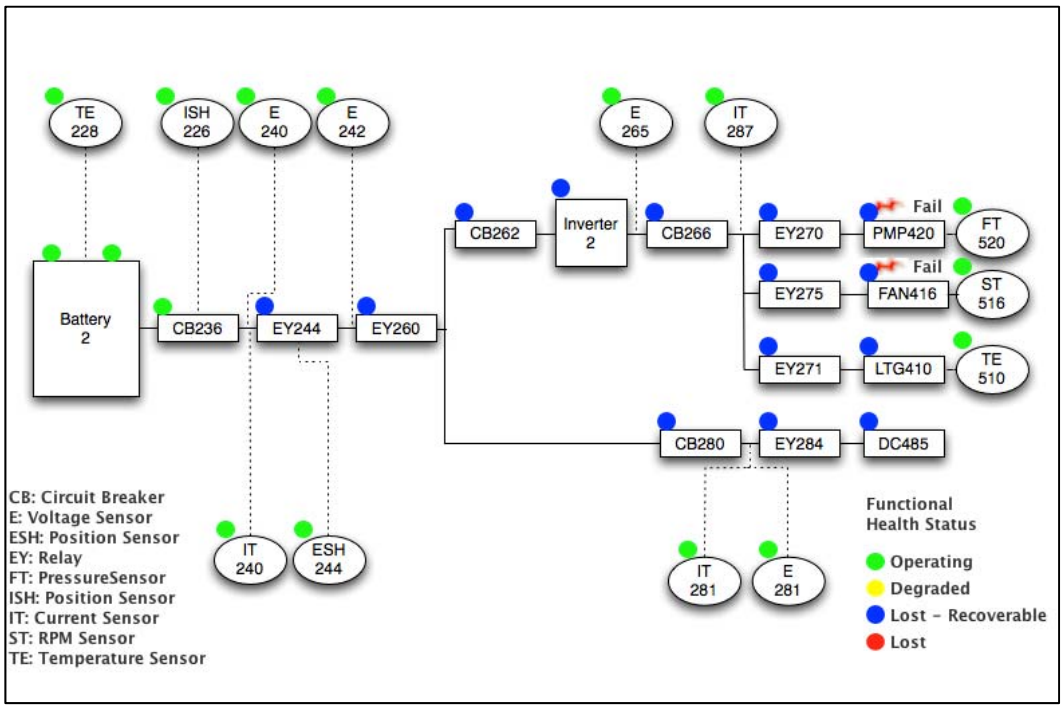


Figure 4. Results of an FFA Fault Propagation Simulation on an EPS Design

3.5 Functional Verification

The next step in the META tool chain is the functional verification of selected candidate designs. We interpret functional verification as a probabilistic evaluation of a composite functional model of the system against functional requirements imposed on the system. In order to accomplish functional verification, the first step is to compose a functional model of the candidate system by retrieving functional models for each component from the component model library and connecting those model fragments following the topology prescribed in the candidate design. Next, we use a probabilistic model checker called PRISMATIC to perform functional verification on these models. PRISMATIC is based on PRISM, an open source tool for formal modeling and analysis of systems that exhibit random or probabilistic behavior [6]. PRISMATIC supports a wide range of probabilistic models including discrete-time and continuous-time Markov chains, Markov decision processes, probabilistic automata, probabilistic timed automata, plus extensions of these models with costs and rewards. Models are described using a simple, state-based language. PRISMATIC provides support for automated analysis of a wide range of quantitative properties of these models, including:

- Internal model consistency (e.g., only one failure mode is active for any component at any time);
- Fault probability analysis (e.g., what is the probability that some fault occurs up to time T, or steady state);
- Time bounded functional assessment (e.g., what is the probability that some component is functioning normally at, before, or up to time T);
- Limited-fault analysis (e.g., if exactly X component(s) fail(s) by time T (or steady state, "long run"), what is the effect on other failure modes or functions?).

3.6 Performance Verification

Another important aspect of certification is the verification of a system (or a system design) with respect to performance requirements. In later stages of development, such performance verification is often performed through hardware-in-the-loop testing. During the early (conceptual) design stage, performance verification has to be accomplished using models of system behavior. However, two types of uncertainty confound the use of models for verification. Epistemic uncertainty comes from model fidelity and inaccuracy, and aleatory uncertainty is contributed by inherent limitations in estimating relevant system parameters and environmental variables. Thus, model-based performance verification has to take uncertainty into account. To address these needs, we employ uncertainty propagation (UP) methods used in reliability-based design [7] to quantify the PoC of a given system design.

Common UP methods can be classified in five broad categories:

- Simulation-based methods such as Monte Carlo simulation;

- Local expansion-based methods like the Taylor series method or perturbation method;
- Most probable point-based methods such as the first-order reliability method and second-order reliability methods;
- Functional expansion-based methods, such as the Neumann expansion and the polynomial chaos expansion;
- Numerical integration-based methods, where the statistical moments are first calculated by direct numerical integration, and then the probability density or the tail region probability is approximated using an empirical distribution system based on the calculated moments.

A detailed description of these methods and their application in PoC computation is beyond the scope of this paper. Further details of our approach to UP may be found in a recent publication by our team [8]. Most importantly, none of these methods is clearly superior to others. Depending on the type of models available (qualitative, quantitative, or hybrid), function types (linear, quadratic, highly non-linear), input distributions (parametric or non-parametric), and input/output distribution types (normal, uniform, beta, or exponential), we have devised an algorithm to choose the most suitable UP method from a library of six different methods.

The first step in the performance verification process is to compute the PoC of the model (or a model fragment) with respect to a single performance requirement. For instance, we computed the PoC for an EPS subsystem containing a battery, an AC inverter, and three loads consisting of an electrical motor, a pump, and a light. The left hand graphic on Figure 5 illustrates the result of PoC computation for this system with respect to a requirement that states that the motor speed shall remain between 900 and 1000 RPM during nominal operation. The right hand side of the same figure illustrates the result of a second PoC computation for the same system with respect to a different requirement that requires the pump voltage to remain between 117 and 125 VAC. In this example, the PoC with respect to the first requirement is computed as 0.881 whereas the PoC for the second requirement is 0.899. The solid red areas indicate the total probability mass associated with meeting each requirement.

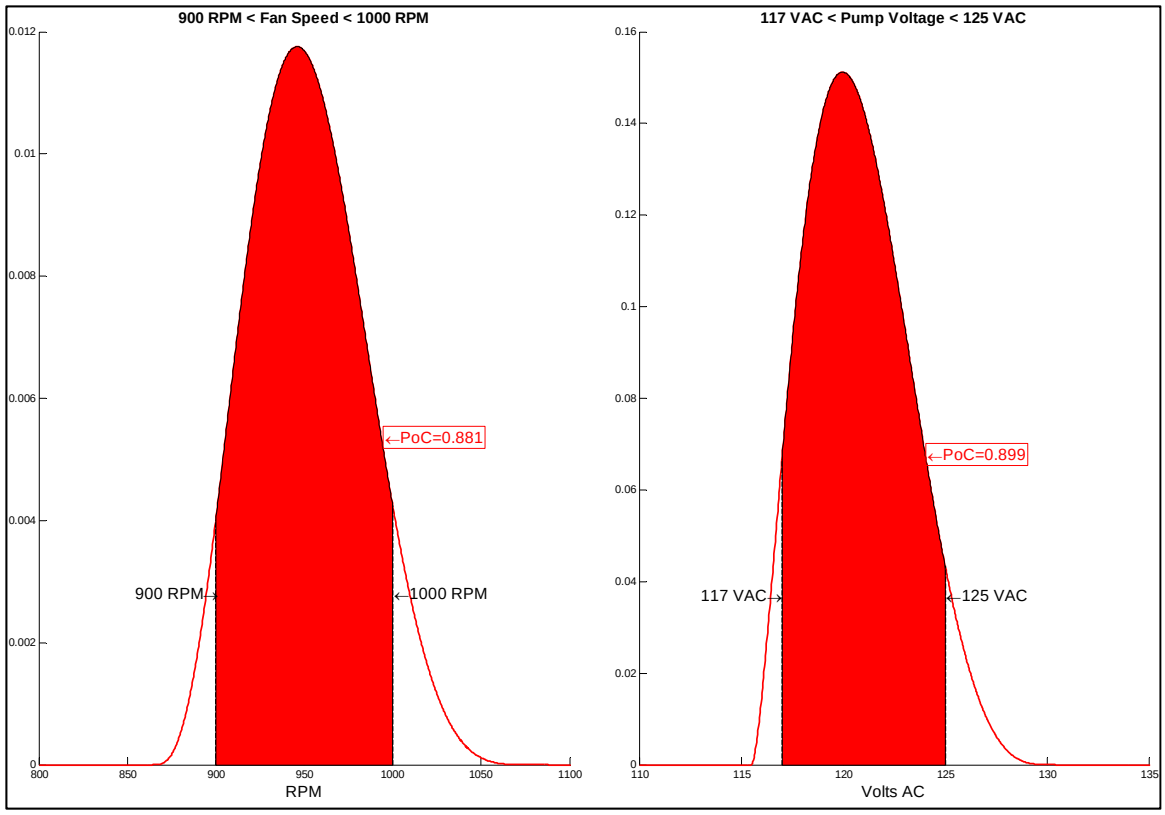


Figure 5. PoC Computation for an EPS Model with Respect to Two Different Requirements

The second step in the performance verification process is to compute the joint probability of meeting multiple requirements. Note that the joint probability cannot be obtained by simply multiplying the two marginal probabilities. Instead, we need to compute the covariance matrix for the marginal probabilities.

$$\Sigma = cov(Y_i, Y_j) = E\left[(Y_i - \bar{Y}_i)(Y_j - \bar{Y}_j)\right] \quad (1)$$

For the same example above, the covariance matrix calculation yields a joint probability of 0.838 for meeting both requirements at the same time (note that this PoC is higher than the simple multiplication of the two individual PoCs, which would have yielded 0.792).

As a final caution regarding the joint probability computation, we cannot use the covariance matrix computation if the marginal distributions are different. In that case, we use the Gaussian Copula function to approximate a true multivariate distribution. Using the Copula function, we can join different types of distributions (normal and beta, etc.).

$$C_Y(u_1, u_2, \ldots, u_M) = \Phi_{\mathcal{N}(0,\Sigma)}(\Phi^{-1}_{\mathcal{N}(0,1)}(u_1), \ldots, \Phi^{-1}_{\mathcal{N}(0,1)}(u_M))$$
$$u_M = F_{Y_M}(Y_M) \quad (2)$$

3.7 PoC and PCC Computation

We define correctness as a measurement of how well the system design meets or exceeds its requirements. Due to uncertainty arising from design abstractions as well as system interactions with the environment and its users, correctness is specified as a probability distribution over multiple dimensions. These dimensions include the design hierarchy (i.e., subsystems and components), environment, use conditions, functions, functional failures, and adherence to design rules.

We define the Probabilistic Certificate of Correctness (PCC) as a data structure that documents the correctness of a particular system design with respect to each class of pertinent system requirements. At the system level, the PCC is represented as a multidimensional probability density function where each dimension represents a distinct system requirement. The PCC is further organized in a hierarchy that provides PCCs at subsystem and component levels. Given a particular use case (represented as scalar values of performance metrics mentioned in the requirements), the PCC provides a probability of correctness for the design under those use conditions. This approach allows the design and development process to focus on areas of low probability of correctness for further refinement and elaboration.

The PoC is represented as a probability distribution (or probability density function) reflecting the probability that the given design meets a performance requirement or a set of related requirements. It is possible to combine the PoCs into a unified PCC. Since the behavior of models is an approximation of actual system performance, the PCC process takes this uncertainty into account and provides information on the resulting uncertainty as well as what components contribute to the uncertainty. The resulting PCC serves as the formal record of performance verification record for the candidate design. Most importantly, unlike traditional design flow, this performance verification is achieved using only (virtual) models of the system.

Our tool chain and framework is the basis for specifying system requirements, supporting design space exploration, and analyzing the performance associated with promising architectural design alternatives. To support a model-based design paradigm, the framework allows the designers to combine models from different domains into integrated system level models, and allow models of components and sub-systems to evolve throughout the design process. At the end, component models are composed into a system that achieves the intended functionality given specified requirements such as reliability, risk, and performance.

3.8 Reliability Analysis

The final analytical operation in the tool chain is a traditional reliability analysis. Our method is based on identification of criticality and sensitivity of system components, and a simulation model that incorporates probability and failure rates of individual components such that system-level reliability measures can be computed. Computed reliability measures include expected system lifetime, component criticality and sensitivity values, and weighted or maximum failure probability [9]. Future work may involve integration of computed criticality data with the FFA method in order to generate the equivalent of a Failure Modes, Effects, and Criticality Analysis (FMECA) during the conceptual design phase.

3.9 System Integration

The Meta-II software tool chain is a heterogeneous set of software systems developed in five programming and modeling languages by four organizations across the United States. It includes significant legacy code bases and newly developed reasoning and integration code. PARC has leveraged its location within Silicon Valley to apply state-of-the industry software engineering practices to ensure successful software development and integration.

The PARC team followed a best-practice agile software development process. Development was planned in short two-week iterations that emphasize the frequent delivery of working software that demonstrates progress allowing us to assess if the project is moving in the right scientific direction.

The team coordinated its work through the JIRA issue-tracking system available to all members through a Web based interface, which provides visibility on task state and is especially effective at identifying dependencies between team members.

Software source was controlled via a subversion repository and change notifications are emailed immediately to team members. This practice ensures the team is kept up to date and also creates peer pressure to write clean code, as it is visible to all.

PARC has taken a test-driven approach to new software development by providing unit tests and monitoring our cost test coverage levels. On each code check-in, the entire system is built and run together with unit and integration tests using a Hudson continuous build server.

PARC has used industry standard tools, techniques, and design patterns in this project. META-II is built using the ANT build system. Tests are written using the JUnit and Mockito frameworks. Code coverage is monitored using the Clover test coverage tool. Logging is controlled through JLog.

The software engineers at PARC and MCT participate in weekly code review meetings with all the engineers in the Embedded Reasoning Area of PARC team. This group with broad research and industry experience ensures that the team keep up to date on industry practice and provides a forum for peer-reviewing software architecture and design issues.

4.0 RESULTS AND DISCUSSIONS

The PARC team has used two case studies to design, develop, and test the META-II tool chain and to demonstrate the features of the proposed verification process. These are the ADAPT EPS testbed at NASA Ames and the Ramp System of an Infantry Fighting Vehicle (IFV). The two systems have been modeled with the help of the Modelica language. Because Modelica is a system modeling language as opposed to a programming language, Matlab is used to code the various UP methods. Matlab calls the black box OpenModelica model using the OpenModelica-Matlab-Interface tool[2], requiring only that the input and output variables be known from the Modelica model, but not the functional relationships coded in Modelica.

The following subsections describe in details the two study systems, the associated models and the experiments performed on the models.

4.1 ADAPT Electrical Power System Example

4.1.1 Domain Description

ADAPT, pictured in Figure 6, consists of a controlled and monitored environment where faults are injected into the system in a controlled manner and the performance of the test article is carefully monitored. The hardware of the testbed represents an EPS for a hypothetical space exploration vehicle.

[2] Schaad, C., 2009. OpenModelica Matlab functions.
http://openmodelica.ida.liu.se:8080/cb/proj/doc.do?doc_id=1085.

Figure 6. The Advanced Diagnostics and Prognostics Testbed (Courtesy of NASA Ames Research Center)

The ADAPT system consists of three major modules: a power generation unit, a power storage unit and a power distribution unit. The interconnections among the different units are depicted in Figure 7.

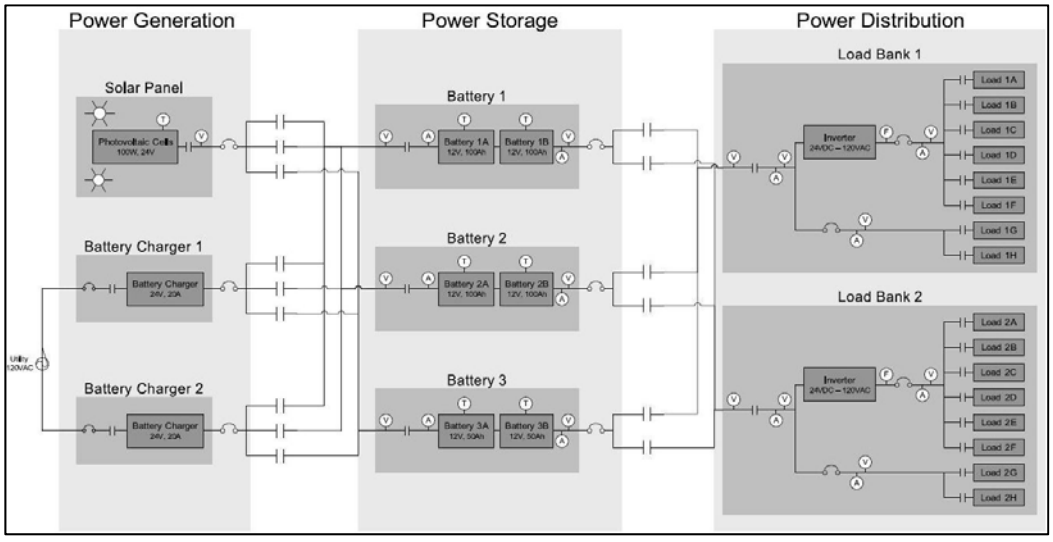


Figure 7. The ADAPT Testbed Structure (Picture Reproduced from [10])

The *Power Generation* unit can charge the batteries located in the *Power Storage Unit* with the help of two battery chargers and a photovoltaic unit (solar panel). The power generation unit is divided into six subsystems: the solar panel unit, the battery charger panel, the protection and enable panel and three battery-charge selection panels. The power storage unit contains three battery packs and several relays that control the connections between the load bank and the batteries. Circuit breakers protect the power distribution unit from dangerously high currents coming from the batteries. The *Power Storage* unit is divided into two major subsystems: the battery cabinet and the battery-load selection panel. The *Power Distribution* unit consists of two identical load banks. Each load bank is connected to the *Power Storage* unit and powers two DC loads and six AC loads.

A complete description of the ADAPT system can be found in [11]. The testbed is controlled by a number of relays and monitored by a large set of sensors. Consequently, it is possible to detect an injected fault and recover from it if the correct action is taken. To facilitate the execution of the experiments performed with the testbed, three operating roles have been defined [10]: *user*, *antagonist* and *observer*. The user simulates an actual crewmember or pilot who operates and maintains the EPS with the help of a vehicle health management application. The antagonist injects faults into the system, either manually by physically acting on the system, or remotely by spoofing sensor values through a computer connected to the system. The malicious actions of the antagonist are not known to the user who is responsible of choosing a suitable recovery action. The observer logs all data in the experiment and monitors how the user responds to the faults injected by the antagonist and therefore measures the effectiveness of the test article. The observer also acts as a safety officer of the experiment and can issue an emergency stop.

4.1.2 Models

The Automated Model Generator developed in the framework of the projects translates a feasible EPS architecture generated by the concept generator into a corresponding model expressed in the Modelica Language.

Modelica is a language for hierarchical object oriented physical modeling, which is developed through an international effort. The language unifies and generalizes previous object-oriented modeling languages and is intended to become a de facto standard for modeling and simulation of dynamical systems. The language has been designed to allow tools to generate efficient simulation code automatically with the main objective to facilitate exchange of models, model libraries and simulation specifications. It allows defining simulation models modularly and hierarchically and combining various formalisms expressible in the more general Modelica formalism. The multidomain capability of Modelica gives the user the possibility to combine electrical, mechanical, hydraulic, thermodynamic etc, model components within the same application model. Modelica is primarily a modeling language, sometimes called hardware description language, that allows the user to specify mathematical models of complex physical systems, e.g. for the purpose of computer simulation of dynamic systems where behavior evolves as a function of time. Modelica is also an object-oriented equation based programming language,

oriented towards computational applications with high complexity requiring high performance. The four most important features of Modelica, relevant for out project, are:

- Modelica is primarily based on equations instead of assignment statements. This permits acausal modeling that gives better reuse of classes since equations do not specify a certain data flow direction. Thus a Modelica class can adapt to more than one data flow context. This feature of the Modelica language was extremely useful in defining complex component failure mode behavior of the EPS components that gave us to possibility for enhanced FFA and PCC analysis.
- Modelica has multi-domain modeling capability, meaning that model components corresponding to physical objects from several different domains such as e.g. electrical, mechanical, thermodynamic, hydraulic, biological and control applications can be described and connected. The EPS consist of components from the electrical and mechanical domain.
- Modelica is an object-oriented language with a general class concept that unifies classes, generics — known as templates in C++, and general subtyping into a single language construct. This facilitates reuse of components and evolution of models.
- Modelica has a strong software component model, with constructs for creating and connecting components. Thus the language is ideally suited as an architectural description language for complex physical systems, and to some extent for software systems.

Modelica defines a standard mapping of hierarchical package structures onto file systems or other storage mechanisms such as databases, which provides the user with a simple and unambiguous way of locating a package. The Modelica library path mechanism makes it possible to make multiple packages and package hierarchies simultaneously available for lookup. All these mechanisms give the user a considerable flexibility and power in structuring a package. Modelica defines a method for locating a package by providing a standard mapping of package names to storage places, typically file or directory locations in a file system. The Modelica format gave us the possibility to easily store the model components in component library repositories that can be easily accessed and retrieved for automatically building feasible EPS architectures. Figure 8 shows the structure of the EPS Modelica performance library and Figure 9 shows the graphical representation of the EPS library model components.

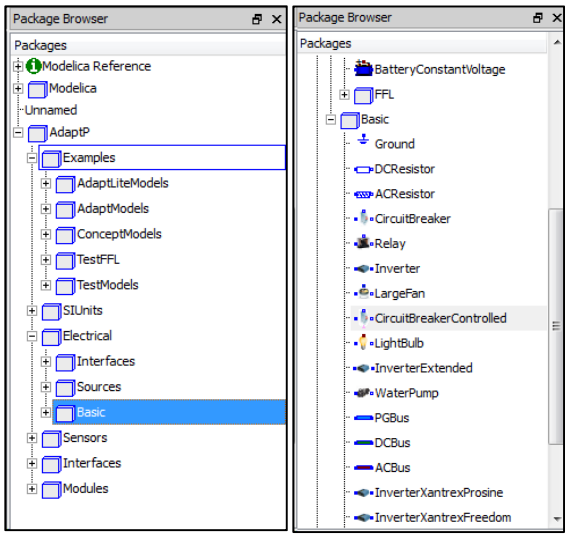


Figure 8. The EPS Modelica Performance Components Library Structure

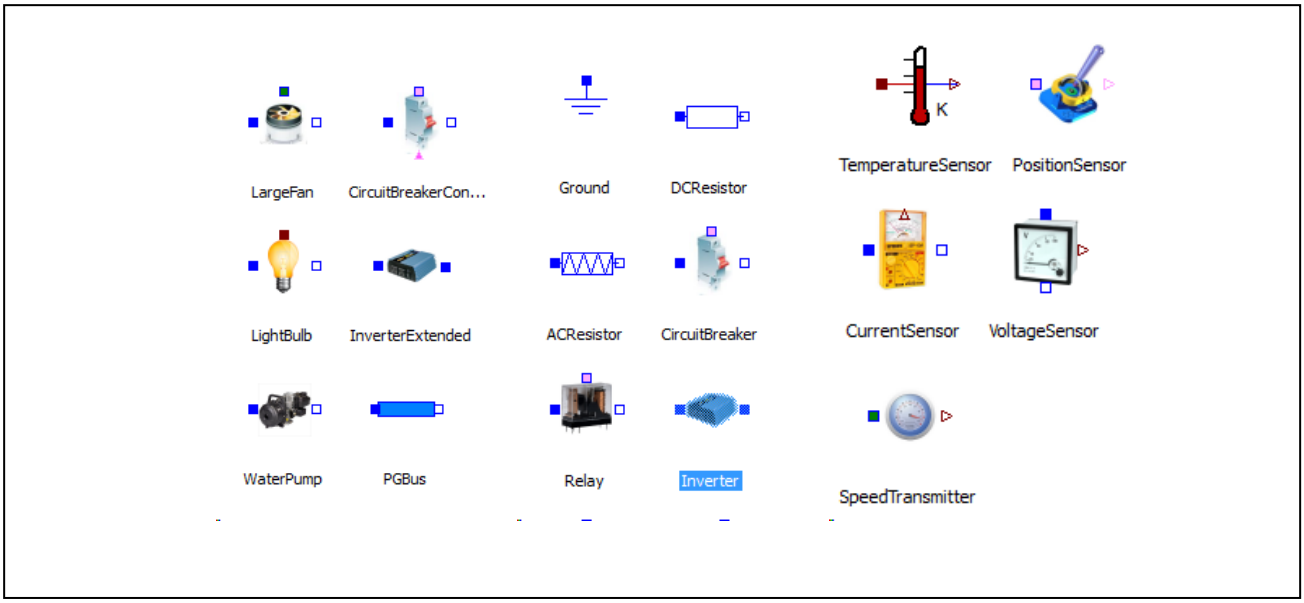


Figure 9. Modelica EPS Components

Given an input XML concept file representing the Automated Model Generator is able to build following models:

- Modelica EPS Performance models for simulation purposes, performance analysis uncertainty propagation and PCC Computation.
- Modelica EPS Reliability models for performing reliability analysis computations as described in Section 3.9.
- Modelica EPS Functional models for performing Functional Failure Analysis as described in Section 3.5.

Figure 10 illustrates the types of models generated by the Automatic Model Generator.

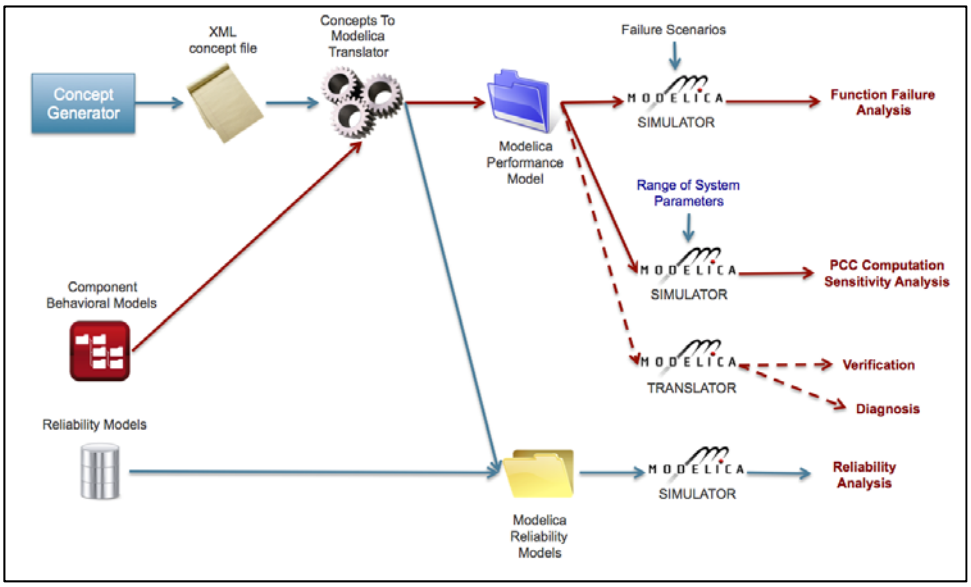


Figure 10: Type of Modelica models generated by the Automated Model Generator

Figures 11 and 12 depict two feasible EPS architectures automatically generated from the concept generator.

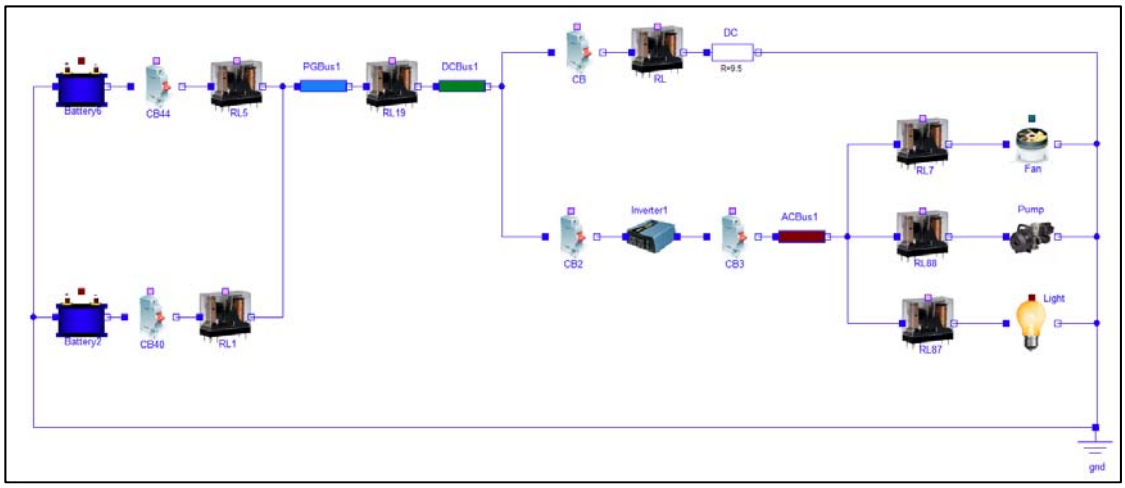


Figure 11. Automatically generated Modelica performance with redundant power sources.

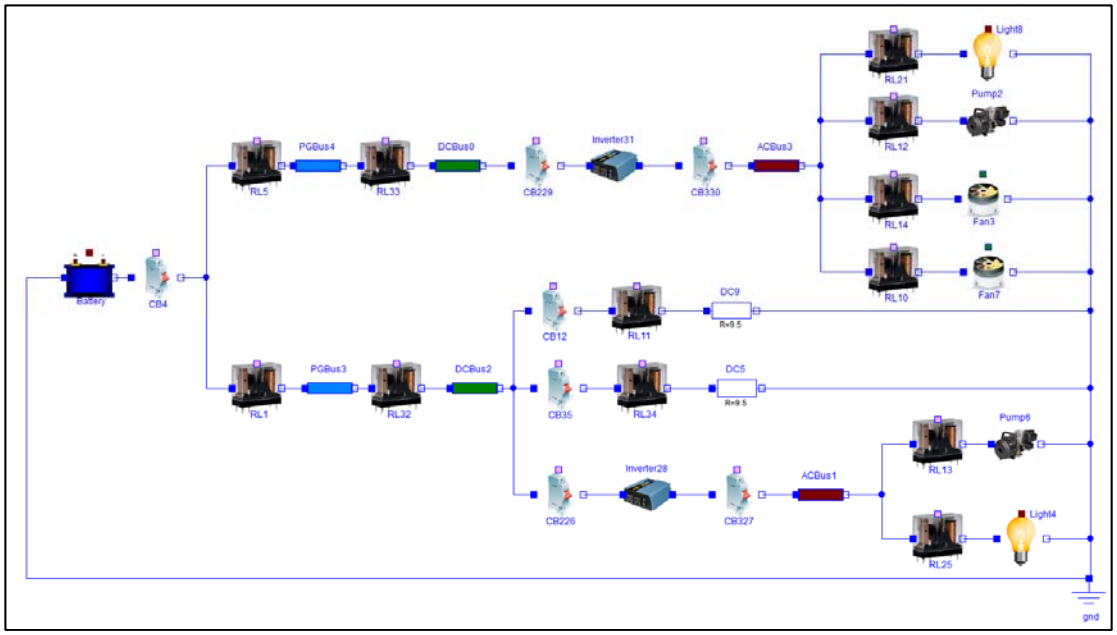


Figure 12. Automatically generated Modelica performance model with one power source and multiple loads.

4.1.3 Findings

For this case study, uncertainty is assumed in two system parameters: 1) inverter output voltage and 2) fan motor resistance. For the three design phases, concept generation, detailed

design and verification, the same Modelica model is used; however the representation of uncertainty is varied as follows:

Phase 1: In this phase, both parameter uncertainties are assumed normally distributed: Inverter output voltage ~N(120.0, 4.0) and fan motor resistance ~N(131.8, 2.2) . The use of the normal approximation of uncertainty represents early design, in which general tolerances for a given component type would be known, but not the actual performance distribution.

Phase 2: In this case, the inverter output, which is specified by the manufacturer to have output of 120VAC + 3% / −10%, is modeled using a four parameter beta distribution, Beta(4.0, 1.9, 108.0, 123.6) , to provide a distribution with + 3% /− 10% range and mode of 120. The fan resistance is modeled as a uniform distribution, U(125.2, 138.4) . This case more accurately represents actual engineering uncertainty in the parameters, which would be known in the detailed design phase.

Phase 3: In this case, the inverter output and fan resistance have the same distributions as Phase 2; however, a full MCS is conducted to verify the PoC computations of Phase 2.

These uncertainties are propagated through the ADAPT Modelica system model using the six methods described in Section 3.8. For each phase, all six uncertainty methods are utilized for illustrative purposes to compare accuracy. A single requirement is considered in the simulation: **"fan speed to be greater than or equal to 780 RPM".** The results of the study are shown in Table 1 for Phase 1 (recommended Phase 1 methods in bold) and Table 2 for Phases 2 & 3 (recommended Phase 2 & 3 methods in bold).

Table 1. Results for the ADAPT EPS: Design Phase 1.

Method	TS	MPP	UDR	FFNI	PCE	MCS
mean	**868.2405**	⋆	868.1354	868.1347	868.1347	868.0543
std dev	**38.3735**	⋆	38.2952	38.2866	38.2866	38.1584
skewness	⋆	⋆	0.0300	0.0220	0.0220	0.0180
kurtosis	⋆	⋆	3.0000	2.9970	3.0010	3.0535
dist	$\mathcal{N}$	$\mathcal{N}$	Beta	Beta	PearIV	⋆
PoC	**0.9893**	**0.9890**	0.9887	0.9889	0.9889	0.9890
# F calls	**5**	**16**	7	10	27+10	10,000

Table 2. Results for the ADAPT EPS: Design Phase 2 and 3.

Method	TS	MPP	UDR	FFNI	PCE	MCS
mean	854.9603	⋆	**855.1931**	**855.1921**	**855.1844**	**855.3800**
std dev	35.3258	⋆	**30.4884**	**30.4747**	**30.3623**	**30.5190**
skewness	⋆	⋆	**0.3130**	**0.2890**	**0.2980**	**0.2857**
kurtosis	⋆	⋆	**2.7450**	**2.7210**	**2.7460**	**2.7048**
dist	$\mathcal{N}$	$\mathcal{N}$	**Beta**	**Beta**	**Beta**	⋆
PoC	0.9831	0.9866	**0.9901**	**0.9907**	**0.9906**	**0.9914**
# F calls	5	20	**7**	**10**	**125+26**	**10,000**

The tables include the first four moments (mean, std dev, skewness, and kurtosis), the distribution of the output, the PoC, and the number of function calls for comparison. In the case of the UDR and FFNI methods, a 3-node approximation is used for both Cases 1 & 2. For the Phase 1 PCE, a second order (i.e., p = 2) polynomial approximation with 3-node quadrature for estimating the moments is used. For the Phase 2 & 3 PCE, a fourth order (i.e., p = 4) approximation with 5-node quadrature is required to accurately estimate the moments due to the non-normality of the input and output distribution. For Phase 2 & 3, the inputs are non-normal and the Rosenblatt transformation is used for the MPP and PCE requiring normal inputs. For the TS method, the mean and standard deviation of the two input distributions is used directly; however, the output is assumed to be normal. For the UDR and FFNI methods, quadrature methods are available to provide nodes and weights for the beta and uniform distributions directly without transformation to normal space. The MCS method can also handle non-normal inputs directly without transformation.

The output distribution resulting from the MCS for Phase 1 is shown in Figure 13. This output distribution is approximately normal as seen by the shape of the distribution and the fact that the skewness ≈0 and kurtosis ≈3. Despite the fact that the output is close to normal, the Pearson distribution system fits alternative distributions to the output based on the deviations of the skewness and kurtosis from the normal distribution as shown in Table 2. All methods provide good approximations of the PoC (using the MCS as the baseline) for Phase 1. This is because the inputs are normal and the system model is not highly non-linear. The assumption that inputs are normal and the model is not highly non-linear is appropriate for the concept design phase when models are of low fidelity and uncertainty distributions are approximated. The output distribution for Phase 2 & 3 resulting from the MCS is shown in Figure 14. As seen by the skewness and kurtosis in Table 2 and the shape of the output distribution, this output deviates considerably from a normal distribution, exhibiting significant skewness and lower kurtosis than the normal distribution. As shown in the table, there are greater differences in the PoC estimates in Phase 2 and 3 than in Phase 1. One reason for the deviation is that methods which assume the output is normal, i.e., TS and MPP, do not account for the skewness and kurtosis of the the actual output distribution. The TS method also suffers from the fact that only the first two moments of the input distribution are considered, and the distribution of inverter voltage is significantly skewed.

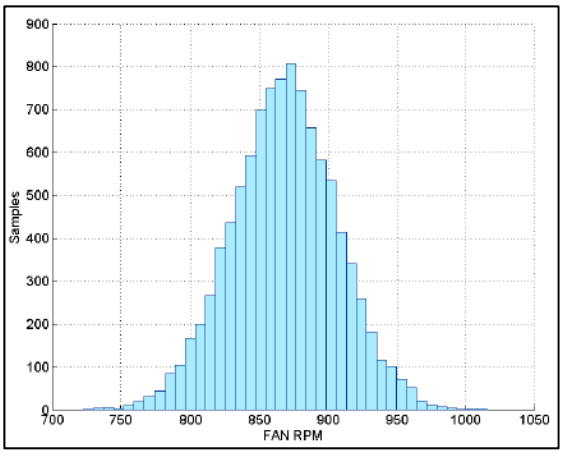


Figure 13. Phase 1 Output Distribution from MCS.

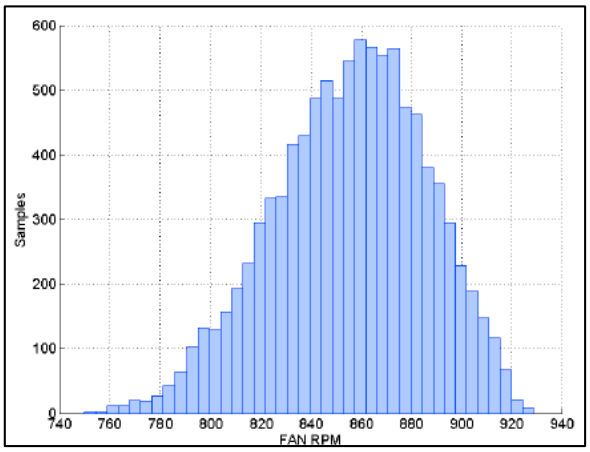


Figure 14. Phase 2/3 Output Distribution from MCS.

4.2 IFV Ramp Example

4.2.1 Domain Description

To demonstrate the features of the proposed verification process and the scalability of the methodology and the tools proposed by the META II project, a Ramp System of an Infantry Fighting Vehicle (IFV) has been modeled. A power-operated ramp at the rear of the vehicle is used for fast exit and entry of troops and the ramp is also fitted with a door (Figure 15 illustrates the rear ramp of a Bradley Fighting Vehicle).

Figure 15. A BAE Bradley Fighting Vehicle (Courtesy of BAE Systems)

Several structural, performance and safety requirements have been preliminary imposed on the design and functionality of the rear ramp system. Table 3 below illustrates some of the performance requirements for the rear ramp system.

Table 3. Performance Requirements for the Rear Ramp System

	Performance Requirements
1.	The ramp shall accommodate a static weight of 320 kg when in fully open position.
2.	The ramp shall accommodate a static weight of 320 kg when in partially open position with no more than 1 cm of play in either direction.
3.	The ramp shall operate while the CFV is inclined upslope at an angle of 60% (27 degrees).
4.	The ramp shall operate while the CFV is inclined downslope at an angle of 60% (27 degrees).
5.	The ramp shall operate while the CFV is on a level surface.
6.	With the engine running, the ramp shall operate from the fully closed position to the fully open position in 10 seconds or less.
7.	With the engine running, the ramp shall operate from the fully closed position to the fully open position in 10 seconds or less.

4.2.2 Models

A hierarchical Modelica model (see Figure 16) of the Ramp System has been built. This model consists of an electrical power subsystem (Figure 17), a control subsystem (Figure 18), a mechanical subsystem (Figure 19), and a crew subsystem (Figure 20).

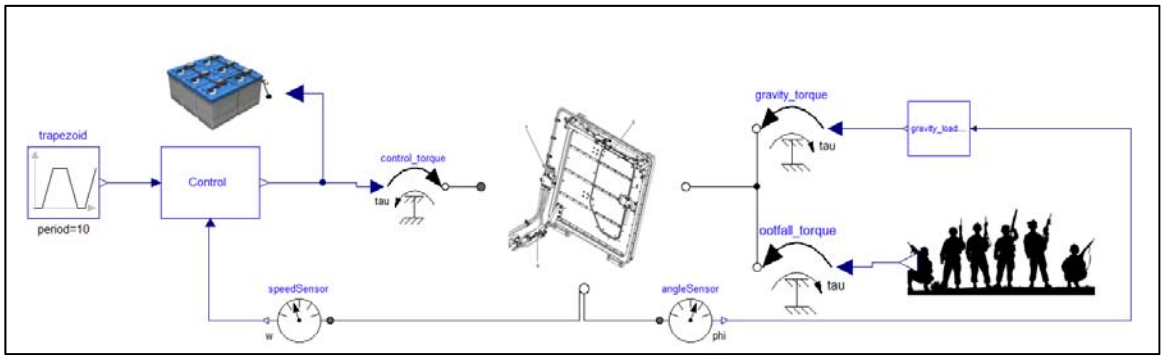


Figure 16. The Modelica Model of a Rear Ramp System

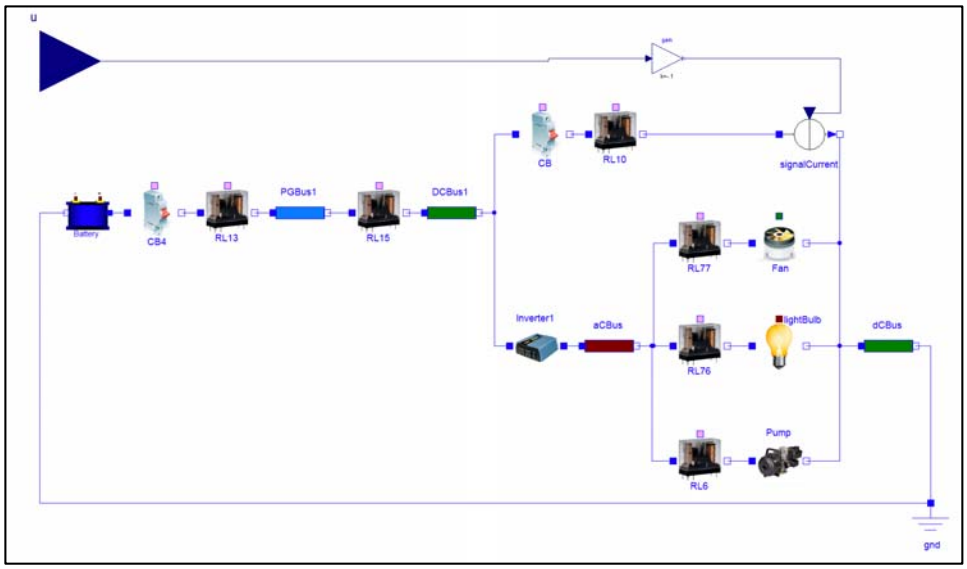


Figure17. The Electrical Power Subsystem

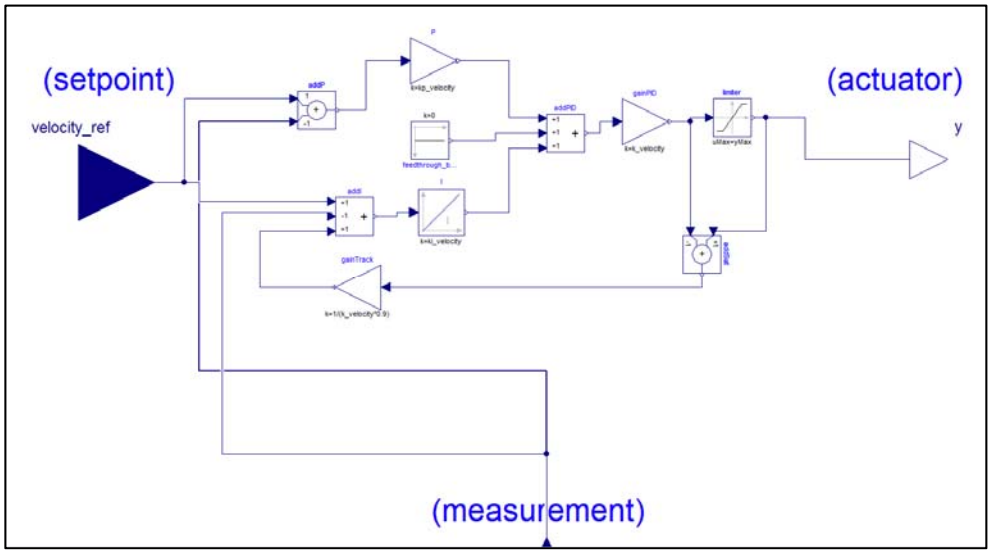


Figure 18. The Control Subsystem

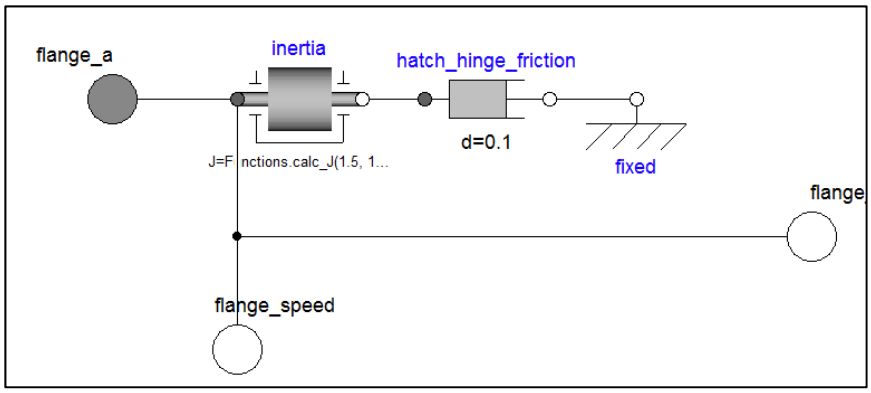


Figure 19. The Mechanical Subsystem

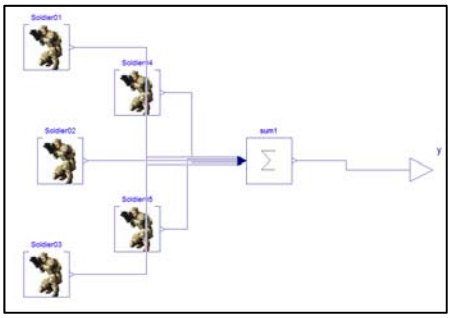


Figure 20. The Crew Subsystem

4.2.3 Findings

Uncertainty was induced in 15 components as shown in Table 4. The tolerance limits were used to model uncertainty in the inputs using 4 parameter beta distributions. The uncertainty in the components creates uncertainty in meeting the system requirements.

Table 4. Model Stochastic Inputs

	Stochastic Inputs	*Uncertainty*
	EPS Subsystem	
1	Fan.rNom	141ohms +/-5%
2	Pump.R	1ohm +/-5%
3	Battery.v	24V +/-7%
4	Inverter1.outputvoltage	120V +4% -10%
	Combat Crew Subsystem	
5	Soldier01.duration	2 +/-10%
6	Soldier02.duration	1.5 +/-10%
7	Soldier03.duration	3 +/-10%
8	Soldier04.duration	2 +/-10%
9	Soldier05.duration	1 +/-10%
10	Soldier01.height	1000 +/-20%
11	Soldier02.height	1000 +/-20%
12	Soldier03.height	1000 +/-20%
13	Soldier04.height	1000 +/-20%
14	Soldier05.height	1000 +/-20%
	Gravity load Subsystem	
15	vehicle_angle.k	0 +/- 0.54

As shown in Table 5, nine representative system outputs corresponding to system requirements were monitored, and the probability of meeting each of these requirements was computed (note that some requirements are listed as meeting the requirement with Pr=1.0 when in actuality the probability approaches but never actually reaches 1.0). In addition to computing the probability of meeting each of the nine requirements, the joint probability of meeting all the requirements is also computed, which is Pr=0.7229. This joint probability is enabled by computation of the variance-covariance matrix for the set of requirements.

Table 5. Probability of Correctness for the Requirements

Requirement	***PoC***
Fan Flow Rate	0.8603
Input Voltage	0.9798
Light Temperature	1.0000
Speed Deviation	0.9970
Door Angle with Load	0.8614
Controller torque (internal)	1.0000
Gravity torque (internal)	1.0000
Solder torque (internal)	1.0000
Opening Time @ -60Deg	1.0000
Joint PoC	**0.7229**

An example an output distribution is shown in Figure 21, which is the distribution of the deviation of the actual ramp speed from the reference signal. As seen in this distribution, the requirement is that the speed deviation be less than 0.015 and due to the variation in speed caused by component uncertainties, there is some area in the upper tail region which is beyond the requirement limit, resulting in a probability of meeting the requirement of 0.997.

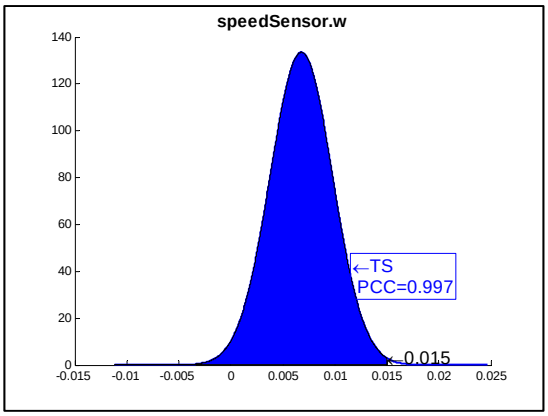
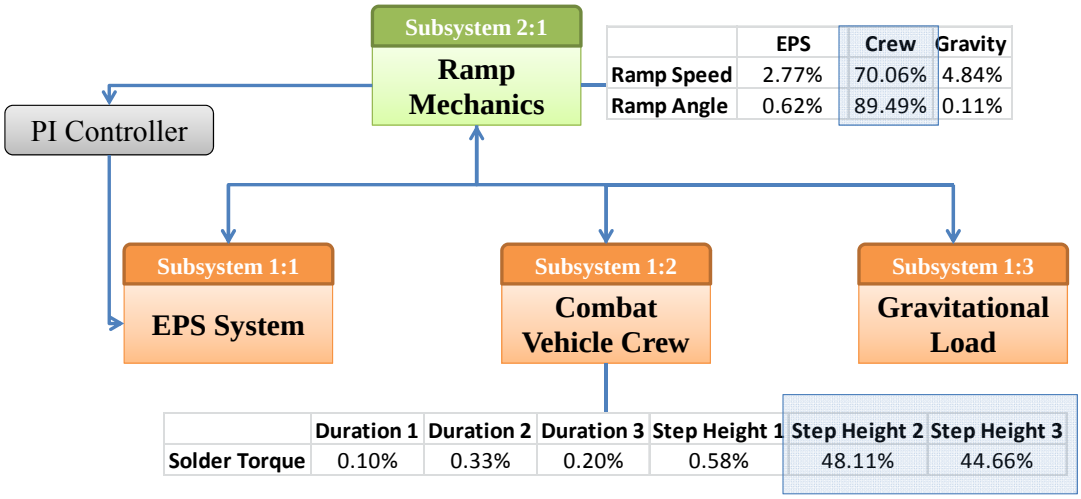


Figure 21. Example of Speed Deviation Requirement

Calculation of the individual probabilities and the joint probability of meeting a set of requirements enable the designer to target where design refinement is needed to better meet

requirements. To understand which components are driving uncertainties in meeting requirements, a hierarchical sensitivity analysis (SA) is conducted as shown in Figure 22. At the system level, the SA shows that the Combat Vehicle Crew accounts for most of the variation in the Ramp Speed (70.06%) and the Ramp Angle (89.49%). Next the Combat Vehicle Crew subsystem is analyzed to determine the drivers of variation in this subsystem. The results show that most of the variation is caused by the solder step height 2 (48.11%) and solder step height 3 (44.66%). This indicates to the designer than more focus must be placed on the solder step heights in the design to reduce variation in the system outputs to increase the probability of meeting requirements.



	EPS	Crew	Gravity
Ramp Speed	2.77%	70.06%	4.84%
Ramp Angle	0.62%	89.49%	0.11%

	Duration 1	Duration 2	Duration 3	Step Height 1	Step Height 2	Step Height 3
Solder Torque	0.10%	0.33%	0.20%	0.58%	48.11%	44.66%

Figure 22. Hierarchical Sensitivity Analysis

5.0 CONCLUSIONS

In this project, we have developed a framework and tool chain for model-based system engineering and design that integrates architectural synthesis and analysis of complex systems during the conceptual design phase. The incorporation of automated design space exploration methods with detailed functional and behavioral models enables architectural trade studies before costly design decisions are made. In future efforts, we plan to fully integrate and automate the architectural synthesis and analysis approaches described in this report. We are also planning to improve the coverage of requirements that are addressable using our approach and to measure the performance of our tools in comparison to human teams performing design tasks of similar complexity.

6.0 REFERENCES

1. Whitney, D., "Why Mechanical Design will never be Like VLSI Design," Research in Engineering Design, Vol. 8, pp. 125-138, 1996.
2. Sangiovanni-Vincentelli, A. "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design," Proceedings of the IEEE, Vol. 95, No. 3, pp. 467-506, March 2007.
3. Kurtoglu, T., Bunus, P., and de Kleer, J., "Simulation-Based Design of Aircraft Electrical Power Systems," presented at the 8th International Modelica Conference 2011, Dresden, Germany, March 20-22, 2011.
4. Kurtoglu, T., Campbell, M., "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping", Journal of Engineering Design, Vol. 20 (1), February 2009.
5. Kurtoglu, T., Jensen, D. C., and Tumer, I. Y., "A functional failure reasoning methodology for evaluation of conceptual system architectures", Research in Engineering Design, 21, pp. 209–234, 2010.
6. Kwiatkowska, M., Norman, G. and Parker, D., "PRISM 4.0: Verification of Probabilistic Real-time Systems," presented at the 23rd International Conference on Computer Aided Verification (CAV'11), Snowbird, Utah, USA, July 14-20, 2011.
7. Du, X., Sudjianto, A., and Chen, W., "An integrated framework for optimization under uncertainty using inverse reliability strategy". Journal of Mechanical Design, 126(4), pp. 562–570, 2004.
8. Hoyle, C., Tumer, I.Y., Kurtoglu, T., and Chen, W., "Multi-Stage Uncertainty Quantification for Verifying the Correctness of Complex System Designs," presented at the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2011), Washington, DC, USA, August 28-31, 2011.
9. Zhang, Y., Kurtoglu, T., Tumer, I.Y., and O'Halloran, B., "System-Level Reliability Analysis for Conceptual Design of Electrical Power Systems," presented at the Conference on Systems Engineering Research (CSER) 2011, Redondo Beach, CA, April 15-16, 2011.
10. Poll Scott, Ann Patterson-Hine, Joe Camisa, David Garcia, David Hall, Charles Lee, Ole J. Mengshoel, Christian Neukom, David Nishikawa, John Ossenfort, Adam Sweet, Serge Yentus, Indranil Roychoudhury, Matthew Daigle, Gautam Biswas, and Xenofon Koutsoukos. (2007). "Advanced Diagnostics and Prognostics Testbed." In Proceedings of the International Workshop on Principles of Diagnosis (DX-07). (Nashville, TN, May 2007, 2007).
11. NASA Ames Research Center (2006) "Advanced Diagnostics and Prognostics Testbed (ADAPT) System Description, Operations, and Safety Manual," February, 2006.

APPENDIX A: PROJECT TEAM

Palo Alto Research Center (PARC)

Serdar Uckun

Serdar Uckun is the Principal Investigator for the PARC META-II project. He is a Principal Scientist at PARC. Prior to PARC, he was at NASA Ames Research Center where he led the largest organization in the government focusing on prognostics and health management (PHM) research. Earlier, he served as Director of the Research Institute for Advanced Computer Science (RIACS), Director of Advanced Technology at Blue Pumpkin Software, and Assistant Director of Rockwell Science Center - Palo Alto Laboratory. Dr. Uckun has graduate degrees in Medicine and Biomedical Engineering, and he has completed post-doctoral studies in Computer Science at Stanford. His technical interests include diagnosis, prognostics, and optimization. He served as Associate Editor of the Artificial Intelligence in Medicine Journal and the General Chair of the 2008 International Conference on PHM. He is an Associate Editor of the International Journal on PHM. Additionally, he is founder and President of the PHM Society, a non-profit professional organization. He holds fourteen U.S. patents.

Tolga Kurtoglu

Dr. Tolga Kurtoglu is Principal Investigator for the PARC META-II project and area manager of the Automation for Engineered Systems (AES) group at PARC. His research focuses on the design and development of complex systems, design theory and methodology with a specialization in conceptual design, design automation and optimization, and artificial intelligence in design. He conducts research in the areas of development of prognostic and health management technologies, model-based diagnosis, automated reasoning, systems engineering, and risk and reliability-based design. Dr. Kurtoglu has published over 50 articles and papers in various journals and conferences and is an active member of ASME, AIAA, AAAI, ASEE, Design Society, and the Prognostics and Health Management Society. Prior to his work with PARC, he worked as a researcher at NASA Ames Research Center and as a systems design engineer and lead at Dell Corporation.

Christian Fritz

Dr. Christian Fritz is a research scientist at the Intelligent Systems Laboratory of PARC. His research interests include logic-based knowledge representation, symbolic reasoning, planning with preferences, execution monitoring, and workflow synthesis. Before joining PARC, Dr. Fritz did postdoctoral studies at the Information Sciences Institute. Dr. Fritz received his Ph.D. in computer science from the University of Toronto, Canada, in 2009, for which he received honorable mention of the Best Dissertation Award of the International Conferences on

Automated Planning and Scheduling (ICAPS) in 2010. He also holds a bachelor's and master's degree in computer science from the RWTH Aachen University, Germany (2003). For his Master's thesis, he was awarded the Springorum-Denkmuenze (University Medal) at RWTH Aachen in 2004. In 2007, Dr. Fritz received the Best Paper award at the ICAPS Doctoral Consortium. Dr. Fritz regularly serves as reviewer for prestigious journals in the area of artificial intelligence and on the (senior-) program committees of all of the top conferences in his field.

Peter Bunus

Dr. Peter Bunus' primary interest is in investigating model-based techniques for system engineering. His research philosophy involves tackling practical problems that are relevant and important to the current issues in model-based systems research and propose foundational solutions to them for good impact. Currently, his research is focused on modeling techniques, modeling language development, and model-driven engineering with direct applications to diagnostics and automated program and model verification. Peter has vast experience in conducting research activities with the ability to bring research ideas into production and development. He enjoys acting as liaison between sales, marketing, and technology groups to support product positioning and customer demand as part of new product development. Prior to joining PARC, Peter was involved in several start-up high technology ventures as a co-founder and has been actively involved in building their businesses. Peter also holds an Associate Professor position in Computer Science at Linköping University Sweden where he is teaching Computer Science and Software Engineering courses at graduate and undergraduate level, and performs research in the area of model-driven engineering.

Peter Jarvis

Peter is a seasoned engineer and scrum master with a track record of delivering quality software products on time and budget. He came to PARC in December 2010 after a seven year stint at NASA where he developed the next generation plotting software for shuttle and space station flight controllers. Prior to NASA, Peter spent four years at SRI International working on DARPA and ARDA programs. He holds a PhD in Artificial Intelligence from The University of Brighton (UK) and has over twenty publications in international conferences.

Oregon State University

Irem Tumer

Dr. Irem Y. Tumer is an Associate Professor at Oregon State University (OSU), where she leads the Complex Engineered System Design Laboratory. Her research focuses on the overall problem of designing highly complex and integrated engineering systems with reduced risk of failures, and developing formal methodologies and approaches for complex system design and analysis. Her expertise touches on topics such as risk-based design, systems engineering,

function-based design, failure analysis, and model-based design. Since moving to OSU in 2006, her funding has largely been through NSF, AFOSR, DARPA, and NASA. Prior to accepting a faculty position at OSU, Dr. Tumer led the Complex Systems Design and Engineering group in the Intelligent Systems Division at NASA Ames Research Center, where she worked from 1998 through 2006 as Research Scientist, Group Lead, and Program Manager. She received her Ph.D. in Mechanical Engineering from The University of Texas at Austin in 1998.

Chris Hoyle

Dr. Christopher Hoyle is currently an Assistant Professor in the area of Design in the Mechanical Engineering Department at Oregon State University. His current research interests are focused upon decision making in engineering design, with emphasis on the early design phase when uncertainty is high and the potential design space is large. His research contributions are to the field of Decision-based Design, specifically in linking consumer preferences and enterprise-level objectives with the engineering design process. His areas of expertise are uncertainty propagation methodologies, Bayesian statistics and modeling, stochastic consumer choice modeling, optimization and design automation. He is currently coauthoring the book Decision-Based Design: Integrating Consumer Preferences into Engineering Design, to be published in 2012. He received his PhD from Northwestern University in Mechanical Engineering in 2009 and his Master's degree in Mechanical Engineering from Purdue University in 1994. He served as an Adjunct Professor of Mechanical Engineering at Illinois Institute of Technology in 2009 and was a Summer Intern at the National Aeronautics and Space Administration (NASA) Ames Research Center in 2006. He was previously a Design Engineer, an Engineering Manager, and a Program Manager at Motorola, Inc. for 10 years before enrolling in the PhD program at Northwestern University.

David Jensen

David Jensen is a PhD student and research assistant at Oregon State University. He earned both a Bachelor and Master of Science from Oregon State University in Mechanical Engineering in 2008 and 2009 respectively. His research to date has focused on function-based failure simulation and analysis. He has worked as a summer intern in Intelligent System Division at NASA Ames Research Center and as a researcher at Aalto University in Helsinki, Finland. Previous work in complex software-hardware product design and testing has provided him with a background and interest in failure identification and mitigation. His research has been published in the ASME International Mechanical Engineering Congress and Expo and International Design Engineering Technical Conferences, the IEEE Aerospace Conference and Prognostics and Health Management Society Conference.

Smart Information Flow Technologies (SIFT)

Dave Musliner

For his Ph.D. dissertation, Dr. Musliner designed and implemented the Cooperative Intelligent Real-Time Control Architecture (CIRCA), one of the first AI planning and control architectures capable of reasoning about and interacting with dynamic, hard real-time domains. CIRCA is designed to make rigorous safety guarantees about its plans, using formal verification methods to ensure that its plans (reactive controllers) avoid predicted failures while also achieving system goals.

In collaboration with other current SIFT researchers, Dr. Musliner incorporated formal model checking methods into CIRCA. Taking advantage of the unique aspects of CIRCA's real-time plans, this research team developed a novel model checking algorithm that improved system scalability by two orders of magnitude and led to a U.S. patent on incremental verification. Dr. Musliner also led the extension of CIRCA to reasoning about probabilistic systems, developing the first statistical model checking systems for use in verifying CIRCA's automatically-designed controllers. In addition to research work on CIRCA, Dr. Musliner has applied intelligent control and model checking verification technologies to various real-world industrial problems, including advanced control of oil refineries and aircraft software.

Eric Engstrom

Eric Engstrom has a broad expertise in software engineering and specific interests in the areas of model based development and formal methods. During his 17 years at Honeywell's corporate research center, one of Mr. Engstrom's major contributions is the creation of DOME (Domain Modeling Environment), as highly-regarded open source meta-modeling research platform providing a user-extensible model-based development environment. In collaboration with other NASA and Honeywell researchers, Mr. Engstrom used the explicit-state model checking tool SPIN and the theorem proving tool PVS to support automated verification of time partitioning in the Honeywell DEOS real-time avionics scheduling kernel.

Mission Critical Technologies

Jacquelyn Nagel

Jacquelyn K. Nagel, Ph.D. is an engineering contractor at Mission Critical Technologies working on engineering design focused projects, and has six years of diversified engineering design experience, both in academia and industry. Dr. Nagel has experienced engineering design in a range of contexts, including: product design, biomimetic design, electrical and control system design, manufacturing system design and design for the factory floor. She earned her Ph.D. in Mechanical Engineering from Oregon State University, and her M.S. and B.S. in Manufacturing Engineering and Electrical Engineering, respectively, from the Missouri University of Science

and Technology (formerly known as University of Missouri-Rolla). As a student, she worked at Kimberly-Clark, Motoman, and Intel and gained cooperative education experience in the areas of industrial automation, manufacturing, and design. Dr. Nagel's long-term goal is to drive engineering innovation by applying her multidisciplinary engineering expertise to design, analysis, instrumentation, and manufacturing challenges.

David Kluck

Dave Kluck graduated from Cal Poly, San Luis Obispo with a B.S. degree in Computer Science. He has worked as a software developer for the past 25 years. Dave started out in the image processing and analysis field writing a product line of software used by various universities and medical institutions. For the past 16 years, he has worked for Mission Critical Technologies, Inc. designing and developing a multitude of systems using an ever-increasing array of technologies. These technologies have included, but are not limited to, the more traditional languages of assembly, BASIC, FORTRAN, COBOL, C, and C++ to the web-oriented languages and frameworks of PHP, Python, Django, ASP, HTML, and Javascript. Dave also has extensive knowledge and experience working with a variety of databases including, but not limited to, DB2, Oracle, SQL Server, and MySQL. Dave has designed and developed systems for DOS, Windows, and UNIX ranging from sales tracking, reporting, and analysis to e-commerce solutions to e-learning to website design and maintenance to Web 2.0 applications. These systems have been designed for a wide array of businesses including, but not limited to, the entertainment industry, food services industry, aerospace industry, medical industry, and the government.

Carnegie Mellon University

Matt Knudson

Dr. Matt D. Knudson is currently a research scientist at the NASA Ames Research Center with Carnegie Mellon University. His research interests are in autonomous and adaptable systems, particularly robotics and complex multiagent organizations. These include low-resource intelligent decision making processes for robot navigation and coordination methodologies for large heterogeneous multiagent systems, such as teams of exploration robots and distributed sensor networks. His expertise is in the fields of electronics and computer science, particularly artificial intelligence. Specifically, he develops representations for system states and actions, as well as algorithms that operate on such representations, for agents to make intelligent decisions with as little information as possible. In addition, he designs objective functions for agents that promote inherent coordination, without the need for communication. Prior to accepting a position at NASA Ames, he worked as a post-doctoral researcher at Oregon State University, where he was the supervisor of the Autonomous Agents and Distributed Intelligence laboratory. During this time, he supervised or assisted in the research of graduate students working under NSF, AFOSR, and DoE funding.

LIST OF ACRONYMS AND ABBREVIATIONS

ACRONYM	DESCRIPTION
ADAPT	Advanced Diagnostics and Prognostics Testbed
DARPA	Defense Advanced Research Projects Agency
DDTE	Design, development, test, and evaluation
EPS	Electrical Power System
FFA	Functional Failure Analysis
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects, and Criticality Analysis
IC	Integrated circuit
PARC	Palo Alto Research Center
PCC	Probabilistic Certificate of Correctness
PoC	Probability of Correctness
SA	Sensitivity Analysis
UP	Uncertainty Propagation