Modeling and Cross-Domain Dependability Analysis of Cyber-Physical Systems

Mark R. Blackburn,
Research Professor,
Stevens Institute of Technology,
Hoboken, NJ 07030, USA
E-mail: mblackbu@stevens.edu

Mark A. Austin,
Deparment of Civil Engineering,
University of Maryland,
College Park, MD 20742, USA
E-mail: austin@isr.umd.edu

Maria Coelho,
Ph.D. Candidate in Civil Systems,
Department of Civil Engineering,
University of Maryland,
College Park, MD 20742, USA
E-mail: memc30@hotmail.com

Abstract—This paper discusses a novel method of modeling and formal verification to support dependability analyses. The method is demonstrated in an example of a fault management capability of robots that interacts with equipment and humans. Hazard analyses produce derived requirements for fault management capabilities. These include safety critical functions for collision avoidance and temporary autonomy. Derived requirements are represented formally in models that are used to produce dependability evidence using theorem proving, model- based test vector generation, test execution with code coverage analysis, and requirement-to-test traceability. To address the challenges of heterogeneity of modeling tools and languages, Semantic Web Technologies are used for model composition and model transformation from modeling tools to formal analysis tools.

Keywords—*component, formatting, style, styling, insert (key words) formal methods, modeling, formal verification, dependability, fault management, cyber physical systems.*

I. Introduction

Robotic systems used in advanced manufacturing, are cyber physical systems (CPS), providing competitive advantage to manufacturers [1]. Such robots interact with other manufacturing equipment and with humans, and could pose risk to both. System requirements for these robots, therefore, must include nonfunctional dependability requirements for failsafe operation such as: safety, security, reliability, and availability [2]. These requirements are derived from fault and hazard analyses. CPS are smart, networked systems with embedded sensors, computer processors, and actuators that sense and interact with the physical world in real-time [3]. CPS systems are heterogeneous in design, and oftentimes provide critical services, work with unreliable networks and provide spatial mobility [4]. Due to dependability requirements on CPS, they often possess fault management capabilities that rely on software monitoring using various sensors, as well as control of actuators to help avoid mishaps.

Dependability is a cross-cutting system property. For many years it has been known that the processes used to ensure dependable systems are costly and that these costs increase with the size and complexity of the CPS [5]. Verification is a key element of this cost. NASA, for example, presented industry data indicating that verification is 88 percent of the cost to produce aircraft software meeting DO-178B [6] level A criteria (i.e., safety critical systems), and 75 percent of the cost of meeting level B criteria [7]. Further, the National Institute of Standards and Technology (NIST) Foundations for Innovation in Cyber-Physical Systems report [1] identifies 21 barriers and challenges for CPS dependability (i.e., reliability, safety, and security). Some of the high priority challenges include: 1) the need for increasing coverage of verification and validation (V&V) while reducing costs, 2) coping with complexity and scale of systems when performing V&V, and 3) the inability to apply formal methods at appropriate abstraction levels, especially for a typical engineer. A similar report from the European ARTEMIS Research agenda cited similar needs [8].

Oftentimes, significant manual effort and expertise have been required to verify safety-critical software. Formal methods, providing a more automatic means of verification, hold promise. However, despite advances made in theorem provers and model checkers aimed to support verification, challenges remain in making their use practical [1][9]. Some of the most cost-effective approaches use formal models to prove satisfiability, verify safety properties, and, as a side-effect, produce test vectors that include inputs and expected outputs to be used in testing the target software. The use of these, however, has been limited to specific modeling languages and tools [10]. Furthermore, while it is critically important to have defect-free specifications, the actual software implementation needs to be verified in its operational

978-1-5386-3664-0/18/$31.00 ©2018 IEEE

context. The Federal Aviation Administration (FAA), for example, still requires software testing with code coverage typically based on modified condition/decision coverage [11].

The goal of achieving dependability is further complicated because the task of engineering CPS is inherently multi-disciplinary, typically involving the integration of mechanical, electrical, thermal, software control, networks, reliability, safety, and cost, among others. The specification of CPS systems is distributed across models defined in these various viewpoints. Several challenges emerge. First, there may be "common sense" assumptions about dependability that are not explicitly characterized in any of these models. Second, in the process of creating analytical models, the modeler may, inadvertently or out of practical necessity, introduce constraints that are inconsistent with assertions made by models representing other viewpoints. Recognizing these inconsistencies across viewpoints can be a challenge [12]. Third, there is heterogeneity itself; a DARPA research project [13] cited challenges arising due to the diversity of domain engineering tools, and the span of design flow activities essential for the design of complex CPS. Fourth, tracing requirements through the various levels of abstraction and models is a challenge.

This paper discusses methods, models and tools to address some of these challenges. While the paper discusses these challenges in the context of the dependability analysis of a telepresence robot, the approach is generally applicable to CPS. Section II discusses strategies and methods contributing to CPS dependability analyses. Section III discusses a case study and provides an overview of the hazard analysis that produces derived requirements for the fault management capability of a telepresence robot. Section IV discusses formal verification of the fault management capabilities derived from the hazard analysis. This includes detailed behavioral models, model transformations to formal method tools supporting satisfiability analysis, test vector generation, test driver generation, test execution and results analysis, and code coverage analysis. Section V discusses the use of semantic web technologies (SWT) and topic maps to support cross-domain analysis. Section VI provides a scenario for using topic maps for composing requirements and formal models prior to model transformation into a representation for formal method tools. Section VII assesses the preliminary finds of the method and prototype, and provides some metrics. Section VIII provides some conclusions.

II. Accessing Dependability for CPS

A survey on the state-of-the-art in industrial automation cited strategies and methods that could contribute evidence of dependability [9]. In particular, the survey cited model-based engineering (MBE), formal methods, design patterns and generative programming as promising. A broad survey across industry, government and academia suggests that adoption of model-centric engineering (MCE) is accelerating [14]. MCE includes the discipline-specific design models typically associated with MBE, as well as Model Based System Engineering (MBSE) system models, models for "ilities," and models integrating software, hardware, surrogates and humans-in-the-loop.

A trend in MCE is to use graphical domain-specific modeling (DSM), where the underlying domain-specific language (DSL) provides precise semantics. DSMs are often more constrained than general purpose modeling languages because they are targeted at specific viewpoints associated with a specific domain and engineering disciplines. Further, DSM environments often provide more dynamic capabilities to support synthesis (e.g., code generation) and simulation.

Formal methods are fundamentally about proof, and consequently are often supported by theorem provers [15], model checkers [16] and Satisfiability Modulo Theory (SMT) solvers. Behaviors in CPS often involve some type of motion control, planning, and surveillance, which is modeled with nonlinear functions and constraints. SMT solvers can check hundreds of thousands of Boolean and integer clauses, but may be unable to determine the satisfiability or unsatisfiability of nonlinear formulas. While formal methods have traditionally been applied to discrete models, CPS also relies on continuous models used for dynamic systems. Certain hybrid logics, such as differential dynamic logic, support formal methods for continuous control systems [17]. Others have extended deductive formal methods with inductive machine learning capabilities [18].

While there are strong arguments for applying formal methods to dependability analysis, Church's Thesis shows that there is no absolute validation criterion [19]. Further, it is known that no amount of testing can guarantee a program's correctness. Even where formal methods have been applied to system models, systems have failed to operate correctly in the target environment [20]. Therefore, testing will play a role in the V&V of CPS for some time to come. Surveys of strategies to produce tests using model checkers [21] and SMT solvers [22] found that their fault-finding effectiveness was limited, primarily because the selected input values did not expose faults in the software.

There are other research needs and approaches that can contribute to more comprehensive dependability. Simulations of continuous behavior can contribute to verification evidence, but guaranteeing properties such as safety is challenging because the space of possible simulations is so large [17]. Rajhans et al. cite a number of approaches towards multi-model design and analysis of CPS, but notes that most are problem-specific [23]. Their research investigates frameworks to compose and reason about multiple types of models, where the various model views of the underlying system use structural and semantic mappings to ensure consistency and enable system-level verification in hierarchy and composition. Section V discusses a unique approach to address some of the same objectives for using SWT for model transformations [24], but the new approach uses topic maps for cross-domain integrations of domain ontologies.

Ontology-based knowledge and reasoning frameworks are being developed to support decision support for correct-by-design of CPS [25]. NASA is using design patterns in an extended MBSE framework that leverages an ontology to

verify compliance and avoid potential system engineering issues [26]. Finally, so as to improve dependability, CPS are increasingly designed to act autonomously. Toward this end, it is common to use embedded runtime capabilities such as fault management to ensure safe operation for both fault and failure scenarios. These runtime capabilities are key to failsafe operations, but they must be rigorously verified and validated.

III. CASE STUDY

The case study describes fault management capabilities for two safety-critical scenarios of a telepresence robot. In these scenarios, the telepresence robot provides services to an enterprise that spans multiple locations. The robot is used by various remote personnel to rapidly assess problems and make recommendations to technicians on site [27]. The remote operator uses wireless services to control the robot. However, the robot must operate autonomously and safely if the wireless connectivity is lost. In addition, the robot must be able to autonomously avoid collisions with objects and human regardless of the wireless connectivity. This section discusses details that yield requirements for fault management capabilities.

The case study is based on a four-course graduate series in CPS [28]. In addition, four master projects extended two prototypes using hazard analysis, fault tree analysis (FTA), and failure modes and effects analysis (FMEA). These established derived requirements for introducing combinations of hardware and software for design, simulation, implementation and model-based testing of the fault management software.

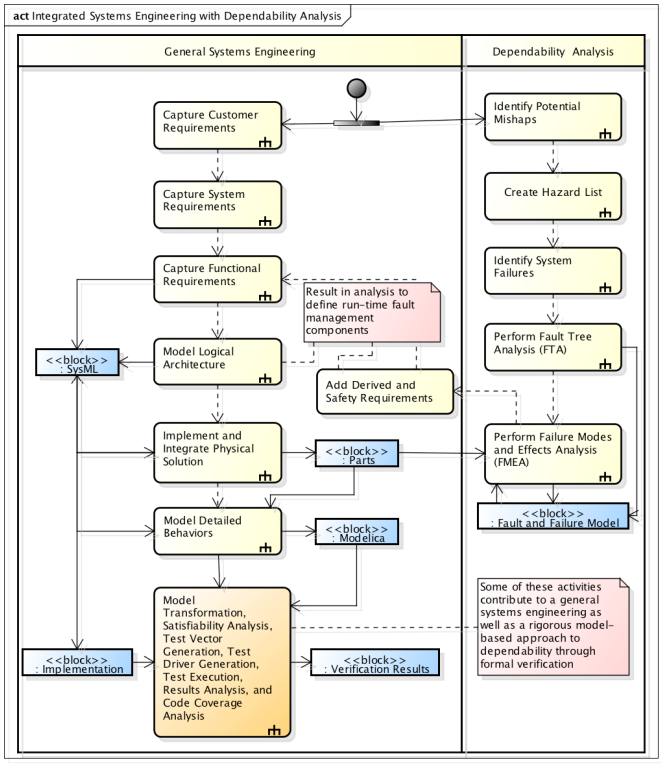

Fig. 1. Integrated system engineering and hazard analysis processes derives requirements that drive the fault management design and formal verification.

Fig. 1 is an activity diagram illustrating the relationship between general systems engineering tasks and the tasks of dependability analysis. Not depicted in the diagram are the feedback loops between dependability analysis and systems engineering. The partition on the left side of Fig. 1 represents the activities that produce a SysML model of the telepresence system starting from continuous development and refinement of the customer needs, system and functional requirements, logical architecture and ultimately detailed behavior specification [29]. The final activity in the left partition relates to the formal verification of the fault management capabilities, which is discussed in Section IV.

Fig. 2 depicts a partial FTA for the system. The two capabilities derived from the hazard analysis that are believed to have the highest safety risk are:

Network Signal Failure: The robot's wifi network connection either loses signal strength or loses connection with the Internet server altogether.

Collision Sensor Failure: Collision sensors fail to respond to the object in a path; this may result in the collision of the robot with that object.

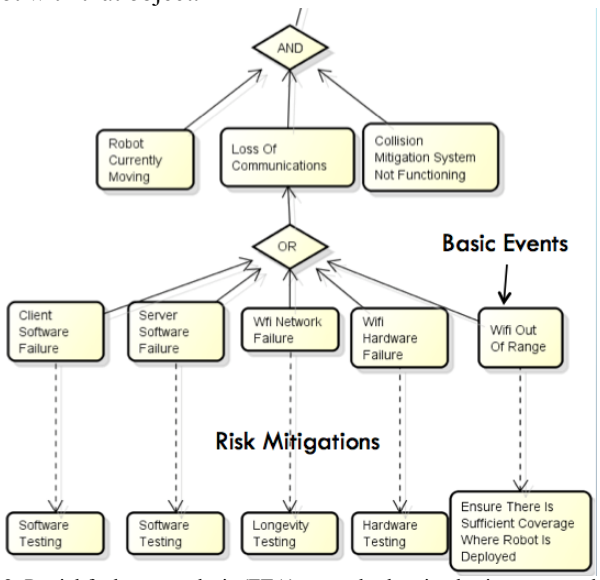

Fig. 2. Partial fault tree analysis (FTA) example showing basic events and risk mitigations [27].

The partition on the right side of Fig. 1 illustrates a top-down approach to modeling dependability. It consists of identifying mishaps, creating a hazard list, prioritizing system failures, and performing fault tree analysis to identify the basic events that could lead to a system failure. Concurrent with design decisions in the left partition, there is an activity to perform FMEA to assess the risk and impacts of components that could be involved in fault or failure scenarios. These analyses combine top-down FTA, with bottom-up FMEA, to produce derived requirement that are integrated with existing requirements. The results initiate activities to update use cases, behaviors, and the product's logical and physical design.

A metamodel for an FMEA is shown in Fig. 3. Starting with the design of a component, FMEA attempts to quantify the risks of failure of the component's function, including the effect and cause. The motivation for the FMEA is to design

controls to mitigate risks. A risk value (risk priority number, RPN) is calculated based on the severity (S) of the failure, probability (P) of occurrence, and detectability (D) prior to failure (e.g., RPN = S x O x D). The fault management models characterize both potential faults and failures relative to the sensor inputs and system states, and specify controls for motion control.

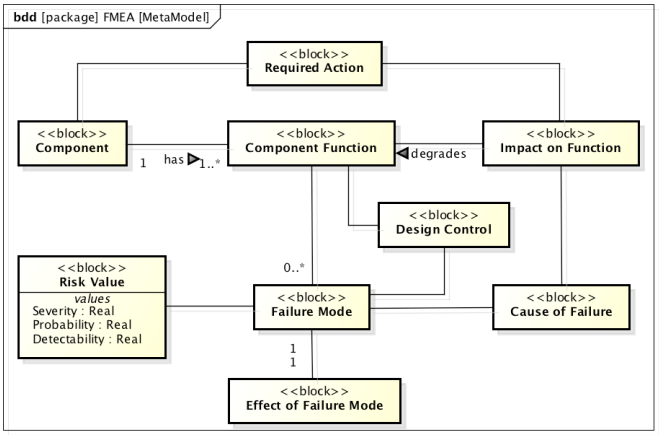

Fig. 3. Metamodel illustrates the conceptual elements involved in Failure Modes and Effects Analysis (FMEA).

IV. Model-Based Formal Verification

The hazard analysis produces derived requirements that factor into the fault management design and implementation. The SysML models discussed in Section III provide an abstract representation of the system blocks, internal block and their relationships. The behaviors represented in SysML activity, sequence, and state diagrams are adequate to identify the threads associated with the software control and data flows, however those specifications are not sufficiently formal to precisely model behaviors. Hence they are not sufficiently formal to apply automated methods of verification. Therefore, detailed behavioral models were created.

There are a few tools suitable to formally model and verify fault management behaviors [10] [22]. Among those, the standard language Modelica and the OpenModelica tool [30] were chosen. Modelica is equation-based and object-oriented. Modelica is acausal, meaning that the direction of information flow between model components is not specified a priori. Modelica is commonly used for modeling and simulating the physical parts of a CPS to investigate properties of a possible system designs. The continuous-time semantics of the models are specified using differential-algebraic equations, which can be further composed and connected into hierarchical model structures. Modelica also supports hybrid models, combining discrete and continuous-time semantics [31].

Modelica is integrated into the dependability analysis using SWT. SWT includes standard languages such as the Web Ontology Language (OWL) [32]. OWL provides a tool-neutral means of specifying domain ontologies that map to metamodels [33] of modeling and analysis tools. The ontology-based approach uses a repository of linked graphs where an ontology provides a semantically precise conceptualization of a domain, but in a modeling tool-independent way [34].

The prototype discussed herein uses SWT capabilities to transform a Modelica model into a representation suitable to a formal methods tool. An overview of the process that extends Fig. 1 is shown in Fig. 4. The SysML model along with derived requirements from the FTA and FMEA analyses were used to develop Modelica models for the behaviors that are implemented in the telepresence robot using Python. The SWT prototype performs a transformation from Modelica to the DSL of a toolset that performs satisfiability analysis, test vector generation and test driver generation [10]. The generated test driver wraps the implementation of the fault management code and executes all of the test vectors. The test driver captures the actual outputs, which are then compared against the expected outputs produced by the test vector generator. A Python code coverage tool [36] checks to determine if the test vectors have fully exercised all code statements and decisions. This was an iterative process both to correct unsatisfiable constraints in the Modelica model, and to improve the testability of the implementation to increase the test coverage.

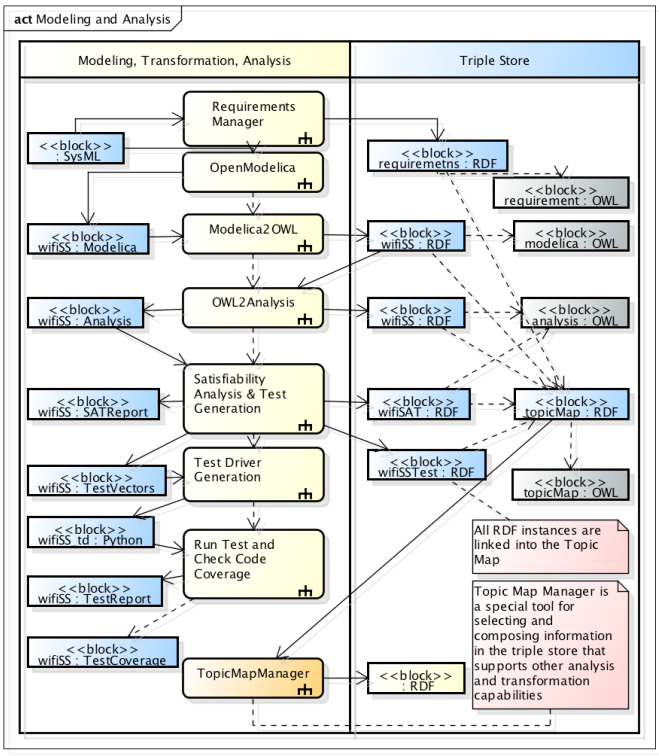

Fig. 4. Notional process flow for the modeling and analysis in the context of triple store repository that includes ontologies and RDF linked through the topic map, which integrates related element across different domains and viewpoints.

Fig. 4 shows the interactions between modeling and analysis activities and their underlying OWL representation in the triple store, which is used to store and relate information as Resource Description Framework (RDF) triples (i.e., subject, predicate, object) [37]. The triples in the triple store can be viewed from two perspectives: as OWL, the triples represent an ontology, and reasoners can be used against the content to ensure that it is well-formed. As RDF, the triples are structures that can be used by SPARQL Protocol and RDF Query

Language (SPARQL) [38].

The left partition in Fig. 4 illustrates some of the modeling, transformation and analysis functions as Call Behavior Actions. Starting from the upper left of the figure, OpenModelica is used to produce Modelica models for the functions such as the wifi Signal Strength (wifiSS). The Modelica models are imported into to the triple store through a transformation into RDF that must be compliant with the Modelica ontology. The specific elements (i.e., individuals in OWL, and instances in RDF) are also linked into the topic map as RDF to create the cross-domain associations, discussed in Section V.

Model transformations such as OWL2Analysis directly use the information in the triple store to produce transformed representations into the DSL of the analysis tool. Satisfiability analysis ensures that each thread of the model is satisfiable. The transformed representation into the DSL of the analysis tool is a first order logical formula represented in Disjunctive Normal Form (DNF). The satisfiability analysis first proves that the constraints (precondition) in each disjunct (logically AND'd set of Boolean-valued conditions) have a non-null input space, from which test inputs are selected and then those inputs are used with the input-to-output relationship (postcondition) to produce the expected outputs for the test. The identification of any unsatisfiable thread is added to an analysis graph and linked in the topic map in the triple store, as discussed in Section V. For all satisfiable threads, a test vector (inputs and expected outputs) is produced. The test driver generator uses the test vectors to produce a Python test driver that wraps the target code and then runs with code coverage analysis.

V. Cross-Domain Integrations using Topic Maps

Ensuring consistency across domains of a CPS contributes evidence of dependability. A new and unique contribution of the research is the use of SWT for cross-domain integration that composes information from different viewpoints, and their associated domain ontology, and then transforms the composed information into representations suitable to other tools. Related approaches [25] [35] for associating domain ontologies that share common information, include: Merging: Ontologies for similar domains are merged into one single coherent ontology. Alignment: Complementary domains ontologies are linked, resulting in two or more ontologies. Integration: Ontologies from different domains are merged in one single ontology.

The prototype discussed herein uses an approach that is a hybrid of alignment and integration. The prototype relates information across domain viewpoints using an ontology-based representation of the Topic Map Data Model [39] [40]. Topic maps possess predefined properties to organize concepts and occurrences into a hypergraph, including: names of subjects, and occurrences and roles played in associations with other topics; these are somewhat analogous to the concept of using an index in a book to find specific topics that are located in different contexts (sections) within the book. Topic maps were designed to ease merging, and to support interoperability of ontologies. The rationale for using an ontology-based formalization of topics maps is to leverage the SWT for querying and automated reasoning using the hybrid approach to alignment and integration.

The TopicMapManager, shown in Fig. 4 is a tool that allows users to select elements from different graphs in the triple store or other linked repositories. The key concepts of topic maps include: Topics, Associations, and Occurrences. Topics correspond to the subject or object of an RDF triple. An occurrence is a link to an element in another ontology or to an external resource denoted by an Internationalized Resource Identifiers (IRI). An association is a relationship between two or more topics. This allows subjects to be related, even if they have different names in different viewpoints (through the use of a name variant). This addresses the common problem in integration scenarios where the same logical element has different names in different viewpoints.

Some aspects of the topic map concept are formalized here to describe a problem to be resolved with topic maps. As reflected in (1) below, for most CPS there are a set of domain viewpoints (DV) (e.g., network, electrical, software control) that are defined in terms of a metamodel (MM) for each domain viewpoint and an associated application model (AM) (e.g., wifi Modelica model), where the AM_i conforms to the MM_i. For each domain viewpoint there is a mapping to an ontology namespace (NS), ontology (O), and set of instances (I) (e.g., RDF triples, subject-predicate-object) that must conform to the ontology O_{DVi} in the context of the namespace NS_{DVi}. NS_{DVi} is actually a set of namespaces used in the context of the ontology, where the default context is the union of the elements. This is inherently represented as a graph (G) G_i in the triple store repository.

$$DV_i: \{MM_{DVi}, AM_{DVi}\} \rightarrow G_i: \{NS_{DVi}, O_{DVi}, I_{DVi}\} \quad (1)$$

Any NS_j contains a set of namespace prefixes that when used to characterize information in the triples I_j must conform to O_j an OWL ontology. As reflected in (2) below, a transformation (t) t_{DVi} in the context of a namespace, ontology, and instances using SWT SPARQL queries, rules and inferencing produces information in the form of application models AM_j that conforms to the MM_j of the DV_i of the analytical function a_j (e.g., satisfiability analysis). An example of this mapping is the OWL2Analysis transformation, shown in Fig. 4.

$$t_{DVi}(NS_{DVi}, O_{DVi}, I_{DVi}, AK_{DVi}) \rightarrow a_j(DV_i) \wedge i \neq j \quad (2)$$

Note, that while it is convenient and possibly more elegant to use rules and inferencing to support aspects of a model transformation, doing so hides the details. In contrast, the use of SPARQL queries to accomplish the model transformations makes the actual details of the transformation more explicit, reviewable and testable. For example, the SPARQL queries can be run against a repository as a form of testing using a web browser.

In the process of composing information for a cross-domain analysis, the only elements of the component viewpoints

queried are those relevant to the composed analysis. Each topic in a topic map has subject identifiers, which are used in merging of topics for cross-domain analysis. This provides a cross-domain linking capability. Assume (3) that there is a set of graphs G_n where n > 2. Assume, that there is a special graph G_{TM} that is defined by the namespace for the topic map (NS_{TM}), ontology for the topic map (O_{TM}) and a set of instances I_{TM}.

$$G_{TM}: \{NS_{TM}, O_{TM}, I_{TM}\} \quad (3)$$

As formalized in (4), there exists a topic map defined by the named individuals I_{TMk} in G_{TM}, and for each subject (s) for all triples (tr) tr_k in every $G_i \neq G_{TM}$ for all $i \leq n$, if and only if those subjects (s) and objects (o) are related by a predicate (p), and p is in the set of topic map-relevant predicates (P). The set I_{TM} can be populated using SWT rules.

$$G_{TM}: \{NS_{TM}, O_{TM}, I_{TM}\} \ni$$
$$[I_{TMk} = x: \{x \mid s \vee o\} \Leftrightarrow \forall k \text{ in } tr_k(s,p,o) \wedge p$$
$$\in P \wedge \forall i\ [i \leq n \wedge I_i \neq I_{TM}]] \quad (4)$$

There were few resources that use Modelica and SWT [41]. Therefore, integrating Modelica into the prototype (Modelica2OWL in Fig. 4) required the development of a Modelica ontology and an RDF model export capability from OpenModelica. The Open Modelica Grammar developed in ANTLR [42] provided the basis to parse the Open Modelica files into an Abstract Syntax Tree (AST), which provided the needed capabilities to parse and create a RDF representation from OpenModelica. The generality of the Modelica language, the low-level of information provide by the AST, and the object-oriented nature of Modelica allowed information to be represented in RDF at a various level of granularity (e.g., Model, Block, Equations, Expression).

The current prototype is configured to automatically populate topic map individuals for a subset of predicates `rdf:type` when the object is `modelica:Block`. The individual Associations candidates are inferred by name association, but can be user-specified. This allows information from different ontologies to be composed with Modelica at the Block level. Modelica blocks contain variable definitions as well as equations. If the topic map is configured to include objects of the type `modelica:Equations` this would permit information to be composed from different domains at the Equation level. This approach provides a way to have composition at various levels of granularity. An example is discussed in Section VI in the context of linking requirements to Modelica blocks and generated test vectors.

VI. Requirements and Traceability

Requirement-to-test traceability is another useful, and sometimes required [6] [11], criteria for assessing completeness, and for tracing derived requirements from various levels of abstraction and fidelity to tests. The tool supporting test vector generation [10] has a built-in requirement management capability that allows requirement statements to be linked at the block level or to low-level statements (i.e., pre-condition, post-condition). The requirement is also linked to the generated test vector as shown in. The requirements for the telepresence robot were captured as SysML requirement blocks, but there was no direct way to link them to the Modelica model.

The prototype includes a RequirementManager as shown in Fig. 4 that is associated with a requirements ontology derived from a goal-driven approach to requirements engineering [43]. Requirements associated with the fault management capability were added as RDF instances associated with the requirement ontology. The TopicMapManager was used to associate those requirements with blocks from the Modelica model. The OWL2Analysis tool has an option to use associations, such as requirements, defined in the topic map to be included during the model transformation process. This provides a means to link requirement to transformed Modelica blocks in the DSL of the analysis and test vector generation tool. An example of the namespace PREFIX and SPARQL query for getting a requirement linked through the topic map is shown below in equation (5). This query finds all of the requirement associations for the elements in the topic map that has the role of a source (e.g., Modelica block) that is currently being processed. The variables `?association`, `?role`, and `?playedBy` are returned by the query, for the `modelica:Block f_triggerMode`. An example of two of the 18 test vectors with linked requirements is shown in Fig. 6. While the MCE study [14] suggests that there will be increased use of formal models, it is likely that subject matter experts from various discipliens will still use high-level requirements or constraints that need to be traced to the detailed models. The integration of a requirement ontology with the topic map provides a means to compose information and transform viewpoints into one or more analytical models to provide the needed V&V evidence.

VII. Method Assessment and Metrics

The process undertaken by the students demonstrated that in one semester, masters-level students could perform comprehensive dependability analysis spanning the entire systems engineering life cycle. The process starting from the hazard analysis was iterative in nature. In addition, the development of the fault management modeling and analysis was also iterative, both in learning the tools and methods, but also in using the formal analysis to improve jointly the testability of the design and its implementation. There were numerous satisfiability issues (i.e., contradictions in the model) detected as the model and implementation were evolved.

Design-for-testability has been a key tenet in hardware design for many years, but it is not so well followed in software. The initial fault management software was not implemented in a manner that allowed the generated tests to achieve a high degree of code coverage; it was re-factored several times. Each modification to the design often resulted in small modifications to the detailed behavior model and the implementation. When the test results ultimately showed that

the actual values from the implementation matched the expected values for all test vectors, a Python code coverage tool was used to verify the completeness. This is depicted in Fig. 5. The gaps in code coverage are due to code that directly received input socket connections or interactions with the hardware that could not be easily simulated. These types of checks were verified by physical testing under combinations of conditions where collision avoidance scenarios were introduced and the wifi signal was lost. The current prototype for performing OWL2Analysis transformations places some idiosyncratic constraints on the formulation of the Modelica model. Some of these constraints may disappear as the prototype matures. Some appear necessary to leverage automated capabilities from formal analysis to code generation tools. Modeling guidelines may be needed to best leverage tool capabilities [13].

Coverage report: 89%

Module	*statements*	*missing*	*excluded*	*coverage*
ModelRobotCommand.py	82	43	0	48%
ModelRobotProxy.py	22	4	0	82%
RobotCommand.py	29	0	0	100%
RobotProxy.py	27	10	0	63%
RobotState.py	33	0	0	100%
StreamPacketFactory.py	172	102	0	41%
TestMonitor.py	652	1	0	99%
TestMonitor_limit_wheelspeed_direct.py	652	0	0	100%
TestMonitor_read_sensor_vld.py	3066	1	0	99%
create.py	532	396	0	26%
test_all.py	3	0	0	100%
Total	**5449**	**582**	**0**	**89%**

Fig. 5. Code Coverage Metrics

Model and requirements engineering (RE) are inherently interactive. Inconsistencies, satisfiability issues, and coverage deficiency (issues) are added to another part of the analysis ontology and linked to the topic map. Future work will extend this information to use "issue" management as a means to drive a system effort to completion. This applies to RE, which is inherently a process for moving from goals to precise statements, but in this process the requirements can be inherently incomplete and inconsistent. Ontologies provide a means for inferring information that can be used to move from incomplete to complete requirements [43]. Ontologies provide a means to continuously check for, and manage both incompleteness and inconsistencies, and this can provide to new types of objective measures. Such an approach can also be stochastic where modeling patterns are applied to assess likelihood of inconsistencies in models [12].

VIII. Conclusion

This paper has discussed models, methods and tools required to achieve dependability in CPS. Specifically, dependability builds on techniques for hazard analysis such as fault tree analysis to identify the basic events that could lead to a mishap. After design of the various subsystems and component, FMEA is needed in order to target where best to apply monitors and controls that would address fault and failure modes. Models supporting these fault management capabilities need to be carefully designed and analyzed. The resulting implementations should be rigorously verified to ensure that the monitoring and control capabilities work flawlessly. This paper describes novel use of tool-neutral SWT and topic maps as a means to integrate models for cross-domain dependability analysis using formal verification tools. The SWT infrastructure relates information in a semantically precise way so as to improve semantic consistency across domains and viewpoints. This should strengthen the dependability argument for CPS in which the tool is applied.

References

[1] National Institute of Standards and Technology, Foundations for Innovation in Cyber-Physical Systems, Workshop Report, 2013.

[2] Laprie, J.C., "Dependability Evaluation of Software Systems in Operation." *IEEE Transactions on Software Engineering* SE-10 (1984): 701-714.

[3] National Academy of Science Interim Report on Cyber-Physical Systems Education, 2015.

[4] R. A. Baheti & H. Gill, 2011, Cyber-physical Systems, The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), Accessed from: www.ieeecss.org.

[5] G. E., Stark, Technologies for Improving the Dependability of Software-Intensive Systems: A Review of NASA Experience and Needs. Houston, Texas: The Mitre Corporation, 1994.

[6] RTCA, DO-178B/ED-12B - Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Corporation for Aeronautics Special Committee 167 (RTCA) December, 1992.

[7] G. Brat, V & V of Flight-Critical Systems, Safe & Secure Systems & Software Symposium, June 2010.

[8] ARTEMIS-GB-2012-D.46 – Annex 2, 2013. https://ec.europa.eu/research/participants/portal/desktop/en/opportunities/fp7/calls/artemis-2013-1.html.

[9] V. Vyatkin, Software Engineering in Industrial Automation: State-of-the-Art Review. IEEE Transactions on Industrial Informatics 9, no. 3 (August 2013): 1234–49. doi:10.1109/TII.2013.2258165.

[10] M. R. Blackburn, M., R. Busser, A. Nauman, and T. Morgan. "Life Cycle Integration Use of Model-Based Testing Tools," 2:10.D.4–1 – 10.D.4–13. IEEE, 2005.

[11] RTCA, DO-178C - Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Corporation for Aeronautic 2011.

[12] S. J. I. Herzig, A. Qamar, and C. J. J. Paredis, "An Approach to Identifying Inconsistencies in Model-based Systems Engineering," *Procedia Computer Science*, vol. 28, pp. 354–362, 2014.

[13] T. Bapty, S. Neema, J. Scott, Overview of the META Toolchain in the Adaptive Vehicle Make Program, Vanderbilt, ISIS-15-103, 2015.

[14] M. A. Bone, M. R. Blackburn, G. Witus, R. Cloutier, E. Hole, Model-Centric Engineering, Conference on Systems Engineering Research, March 2016.

[15] S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system," in 11th International Conference on Au- tomated Deduction (CADE), ser. Lecture Notes in Artificial Intelligence, D. Kapur, Ed., vol. 607. Springer-Verlag, June 1992, pp. 748–752.

[16] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in Logic of Programs, 1981, pp. 52–71.

[17] S. Mitsch, J.-D. Quesel, and A. Platzer, "Refactoring, Refinement, and Reasoning," in *FM 2014: Formal Methods*, vol. 8442, C. Jones, P. Pihlajasaari, and J. Sun, Eds. Cham: Springer International Publishing, 2014, pp. 481–496.

[18] S. A. Seshia. New Frontiers in Formal Methods: Learning, Cyber-Physical Systems, Education, and Beyond. CSI Journal of Computing, 2(4):R1:3–R1:13, June 2015.

[19] Mili, A., An Introduction to Formal Program Verification, New York, New York: Van Nostrand Reinhold, 1985.

[20] S. Mitra, T. Wongpiromsarn, and R. M. Murray, "Verifying Cyber-Physical Interactions in Safety-Critical Systems," *IEEE Security & Privacy*, vol. 11, no. 4, pp. 28–37, Jul. 2013.

[21] G. Fraser, F. Wotawa, P.E. Ammann, Testing with model checkers: a survey, Software Testing, Verification and Reliability, Volume 19, Issue 3, pages 215–261, September 2009.
[22] M. R. Blackburn, S. Ray. Reducing Verification Costs through Practical Formal Methods: A Survey, System and Software Consortium Technical Report, 2011.
[23] A. Rajhans, A. Bhave, I. Ruchkin, B. H. Krogh, D. Garlan, A. Platzer, and B. Schmerl, "Supporting Heterogeneity in Cyber-Physical Systems Architectures," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3178–3193, Dec. 2014.
[24] M. R. Blackburn and P. O. Denno, "Using Semantic Web Technologies to Integrate Models to Analytical Tools," International Conference on Complex Systems Engineering (ICCSE), 2015.
[25] L. Petnga and M.A. Austin, "An Ontological Framework for Knowledge Modeling and Decision Support in Cyber-Physical Systems," *Advanced Engineering Informatics*, vol. 30, no. 1, pp. 77–94, Jan. 2016.
[26] Jenkins, J. S., N. F. Rouquette, Semantically-rigorous systems engineering modeling using SysML and OWL, Jet Propulsion Laboratory, California Institute of Technology, 2012.
[27] A. Atherton, Dolen, C., Hernandez, E., Romano, N., Cyber Physical Systems Final Report, TBD Robotics Presents: The Dopplebot *System* Model Document, <blind affiliation>, July 2015.
[28] <blind authors>, Project-Based Education for the Systems Engineering of Cyber-Physical Systems, 2016 Conference on Systems Engineering Research, March 2016.
[29] J. Broadbent, W. Diaz, T. Patalano, Cyber Physical Systems Final Report, TrueVisit System Model Document, <blind affiliation>, July 2015.
[30] OpenModelica, https://www.openmodelica.org/.
[31] Broman, David, Edward A. Lee, Stavros Tripakis, and Martin Törngren. "Viewpoints, Formalisms, Languages, and Tools for Cyber-Physical Systems," 49–54. ACM Press, 2012. doi:10.1145/2508443.2508452.
[32] World Wide Web Consortium. OWL 2 Web Ontology Language Document Overview, December 2012. http://www.w3.org/TR/owl2-overview/.
[33] Walter, Tobias, Fernando Silva Parreiras, and Steffen Staab. "An Ontology-Based Framework for Domain-Specific Modeling." Software & Systems Modeling 13, no. 1 (February 2014): 83–108.
[34] Grüninger, M., Joseph Kopena: Semantic Integration through Invariants. AI Magazine 26(1): 11-20, 2005.
[35] F. Song, G. Zacharewicz, and D. Chen, "An ontology-driven framework towards building enterprise semantic information layer," *Advanced Engineering Informatics*, vol. 27, no. 1, pp. 38–50, Jan. 2013.
[36] https://wiki.python.org/moin/CodeCoverage.
[37] RDF – Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, February 2014, https://www.w3.org/TR/rdf11-concepts/.
[38] World Wide Web Consortium. SPARQL 1.1 Overview, March 2013, http://www.w3.org/TR/sparql11-overview/.
[39] ISO/IEC 13250-2: Topic Maps – Data Model, 2005-12-16.
[40] Cregan, A.: Building Topic Maps in OWL DL. In: Proceedings of the Extreme Markup Languages® 2005 Conference, Montreal, Canada, pp. 1–29, 2005.
[41] P. A. Fritzson, *Introduction to modeling and simulation of technical and physical systems with Modelica*. Hoboken, N.J: Wiley : IEEE Press, 2011.
[42] ANTLR (ANother Tool for Language Recognition), http://www.antlr.org/.
[43] K. Siegemund, E. J Thomas, Y. Zhao, J. Pan, and U. Assmann. Towards ontology-driven requirements engineering. In *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference (ISWC), Bonn*, 2011.

Mark R. Blackburn (M'2011) holds a Ph.D. from George Mason University in Information Technology, M.S. in Mathematics (emphasis in C.S.) from Florida Atlantic University, and a B.S. in Mathematics (C.S. option) Arizona State University. He is an Associate Professor in the School of Systems and Enterprise from Stevens Institute of Technology. Dr. Blackburn research focuses on methods, models, visualization and automated tools for reasoning about complex systems of systems. He is the Principal Investigator on a Systems Engineering Research Center sponsored by NAVAIR investigating model-centric engineering. He has received research funding from the National Science Foundation, Federal Aviation Administration, and National Institute of Standards and Technology.

Maria Coelho received a B.S. in Civil Engineering from the University of Maryland, College Park, in 2015, and the M.S. in Civil Systems from the same institution in 2017. Maria is now working toward her Ph.D. in Civil Systems at Maryland.

Mark Austin is an Associate Professor of Civil and Environmental Engineering at the University of Maryland, College Park, with a joint appointment in the Institute for Systems Research (ISR). Mark has served as Technical Director of the Master of Science in Systems Engineering (MSSE) Program at ISR. Mark has a Bachelor of Civil Engineering (First Class Honors) from the University of Canterbury, Christchurch, New Zealand, and M.S. and Ph.D. degrees in Structural Engineering from UC Berkeley.

```
PREFIX rdf:            <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:            <http://www.w3.org/2002/07/owl#>
PREFIX topicmap:       <http://example.net/topicmap#>

SELECT * WHERE {?association topicmap:type topicmap:at_requirement ;
                topicmap:belongsToTopicMap <http://example.net/topicmap#tm_MyTopicMap> ;
                topicmap:hasRole ?role .
                ?role topicmap:type topicmap:art_sourceElement ;
                topicmap:playedBy ?playedBy .
                ?playedBy topicmap:subjectIdentifier <http://example.net/SignalStrength#f_triggerMode>
               }
```

Test Vectors For Subsystem : f_triggerMode

Vector File: B:\projects\SWT\SWT_demos\Modelica_SWT_wifi\test_vectors\f_triggerMode.TST
Open Model Report
Legend For Vector Table

Jump to Page: Vectors <Previous Page 1 of 1 Next Pa

Test #	Vector #'s	f_triggerMode__OUT	wifiStatus	wifiStrength	standbyMode	backupCount	signalStrengthStatus	Requirement IDs
1	1	FALSE = 0	1	2	1	0	TRUE = 1	ReqID(s) : TABLE: The_Terrain_Subsystems_shall_accept_directions_from_python_library TABLE: Wifi_Signal_Lost_Scenario
2	2	FALSE = 0	1	2147483647	1	0	TRUE = 1	ReqID(s) : TABLE: The_Terrain_Subsystems_shall_accept_directions_from_python_library TABLE: Wifi_Signal_Lost_Scenario

Fig. 6. Test vectors to illustrate the association of requirements to test vec.