Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 28 (2014) 883 – 890
Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

# Virtual Design and Verification of Cyber-Physical Systems: Industrial Process Plant Design

Mark Blackburn[a]*, Peter Denno[b]

[a]*Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ, 07030, USA*
[b]*National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD, 20899-8265, USA*

## Abstract

This paper discusses a research project to support virtual design and verification of industrial process plant designs. Process plants are a class of cyber-physical systems (CPS), and these research results should generally apply to other types of CPS such as those associated with the Smart Grid. Modeling is an essential part of process plant design and integral in other applications such as manufacturing. Models produced in design have obvious roles in system implementation, deployment and certification. For manufacturing systems, models also have use in downstream activities including system certification, performance optimization, real-time diagnostics and prognostics, and maintenance. The paper discusses the results associated with a prototype that uses domain-specific models of different views of a system design that improves collaboration through integrated models and aligned semantics and provides examples of how the integration with formal methods can identify defects in designs, and automatically generate test vectors with requirement-to-test traceability.

* Corresponding author. Tel.: 561-637-3452; fax: +0-000-000-0000 .
*E-mail address:* mark.blackburn@stevens.edu

doi:10.1016/j.procs.2014.03.006

*Keywords:* Model Based System Engineering; formal methods; domain-specific modeling; test generation; smart manufacturing; verification

## 1. Introduction

Models should play a key role in the design, verification and validation (V&V) of process plants. Process plants, and manufacturing facilities generally, are classes of cyber-physical systems (CPS) and types of systems within a smart manufacturing system of system. The design of manufacturing systems involves collaborative effort among multiple viewpoints and disciplines. Process plants and CPS continue to evolve; robots are playing more significant roles and future systems will rely less on human decision-making and more on computational intelligence. The need for verification is expected to increase with the increasing complexity of systems and automation[1]. The complexity of manufacturing systems and their safety requirements place demands on system design, implementation, and V&V. Models can serve in roles downstream of design such as system certification, performance optimization, real-time diagnostics and prognostics, and maintenance. Model-based systems engineering (MBSE) is intended to support many of these roles and address these requirement-to-test traceability challenges. In the particular context of process plant design, however, neither the current generation of MBSE tools, nor commonplace practices are particularly well suited to these ends.[1]

The current generation of MBSE tools provide general-purpose viewpoints[2] that oftentimes are not efficient at conveying domain-specific information. Viewpoints and analysis tools are often not integrated and may not be consistent across models. More importantly, many types of models lack the semantic richness to allow designers to formalize critical aspects of the systems to leverage more rigorous analysis, simulation and synthesis capacities required to ensure both performance and dependability requirements are achieved.

Commonplace practices of process plant design rely heavily on paper-based specifications generally referred to as product data sheets. A data sheet provides information that specifies functional and physical characteristics of a component of a system, plant or facility. Data sheets are used both by the system integrator to specify requirements of equipment components, and by suppliers to describe how their product addresses these requirements. However, the relationships between properties stipulated in the data sheet and design rationale and decisions are oftentimes not obvious. The inability to trace from design commitments back to their rationale and requirements may have consequences not only in V&V but also in the maintenance and operation of the systems and the plant. For example, after years of operation, equipment identical to the original equipment used may become unavailable, making it necessary to reference design requirements to identify suitable replacement equipment. Models used in activities downstream of design, e.g., testing for safe operations, should be traceable to design decisions and operational intent.

Automated test generation and formal methods of V&V may provide value by ensuring test coverage and by linking tests to requirements. Knowledge of the testing performed relative to requirements can serve to further describe the requirements. But applying formal methods to process plant system engineering presents its own problems. The challenge is twofold: there is a burden on the engineer to specify the requirement in a form required by the formal method, and with increasing formality, it becomes increasingly difficult to interpret the results of testing in terms meaningful to the engineer. Both of these effects also impede the ability to relate testing back to requirements.

This paper discusses an approach that uses domain-specific modeling (DSM) for capturing and sharing equipment, system and facility-related requirements, design information, and constraints. DSM provides graphical notations intended to facilitate the translation of stakeholder concerns into the information requirements of the application. Our particular use of a DSM concerns the use of formal methods of V&V and associated tooling. It is important to not burden the user with the mechanics of the formalism. The DSM representations provide the semantic richness that is required to formalize requirements, design, and safety properties to support systems V&V using formal method automation.

The unique contributions of this research are a methodology and prototype that demonstrate how some types of formal methods (i.e., satisfiability, proof of properties) can be integrated into familiar manufacturing and systems engineering viewpoints using domain-specific modeling. The paper describes the prototype DSM-based toolchain, which supports different viewpoints of integrated models for a system design, and provides examples of how the integration with formal methods can identify defects in the design, and automatically generate test vectors with requirement-to-test traceability. These results use a process plant design example, but the results should be equally applicable to other CPSs such as smart grids. Finally, the DSM process described in this paper identifies how researchers and technology specialists can leverage this type of DSM platform to integrate their latest analysis and simulation capabilities for key CPS challenges in V&V.

## 2. Concept Overview- DSM Integration with Formal Methods

We seek to efficiently and effectively apply V&V to these increasingly complex systems. V&V activities reference requirements. The method of capturing and representing requirements must be comprehensible to the various stakeholders involved in process plant engineering. In the current design of process plants, the product data sheet is the key means by which equipment requirements are represented and communicated among suppliers and designer. A DSM enables engineers who do not have detailed knowledge of the formal methods used to specify V&V problems. Specifically, the engineer uses representations familiar to him such as piping and instrumentation diagrams (P&IDs) and properties typically found on product data sheets that are captured in the model.

Our prototype supports: 1) capturing process system requirements, design, and property specifications, 2) automating generation of system tests that represents the logical structure of an envisaged system, its components and their interconnection, and 3) verifying that interaction of components achieves system goals.

A closed-loop, heat transfer, liquid circulating (CHL) system shown in Fig. 1 was used in the prototype. The model uses P&ID graphical notations to represent components such as storage tanks, pumps, heat exchanges, sensors, and valves. The CHL P&ID is part of a document-based system breakdown specification that includes a system requirement tree with associated system, component and interface requirements. P&IDs are generated as views of more detailed models of the system, e.g., the 3D model of the physical configuration and network and logic models for control systems and operations.

DSM languages (DSML) are applied to the problem. A DSML is a modeling language defined using a metamodel and used to mediate between a user viewpoint and the information needs of a software application. DSMs are an emerging type of MBSE technology that enable domain subject matter experts (SME) to express application concepts and design intent using notations that precisely match the domain's semantics[3,4]. A DSM can provide relevant and intuitive graphical abstraction for specific domains, which flatten learning curves for users.

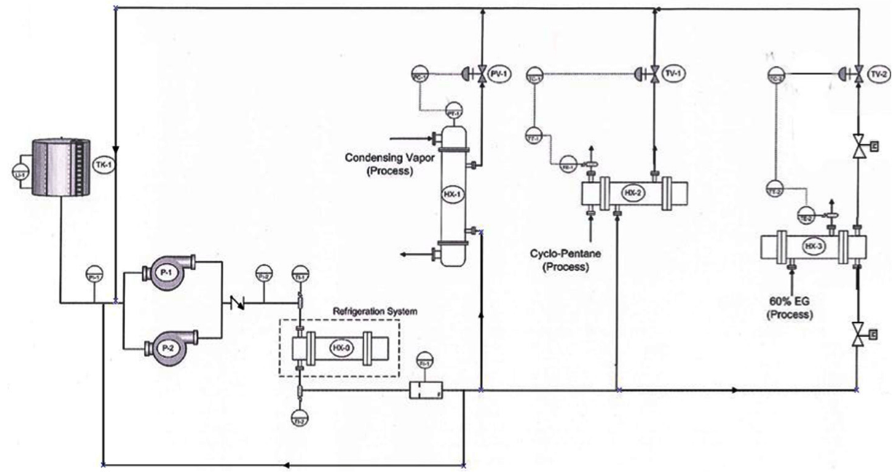
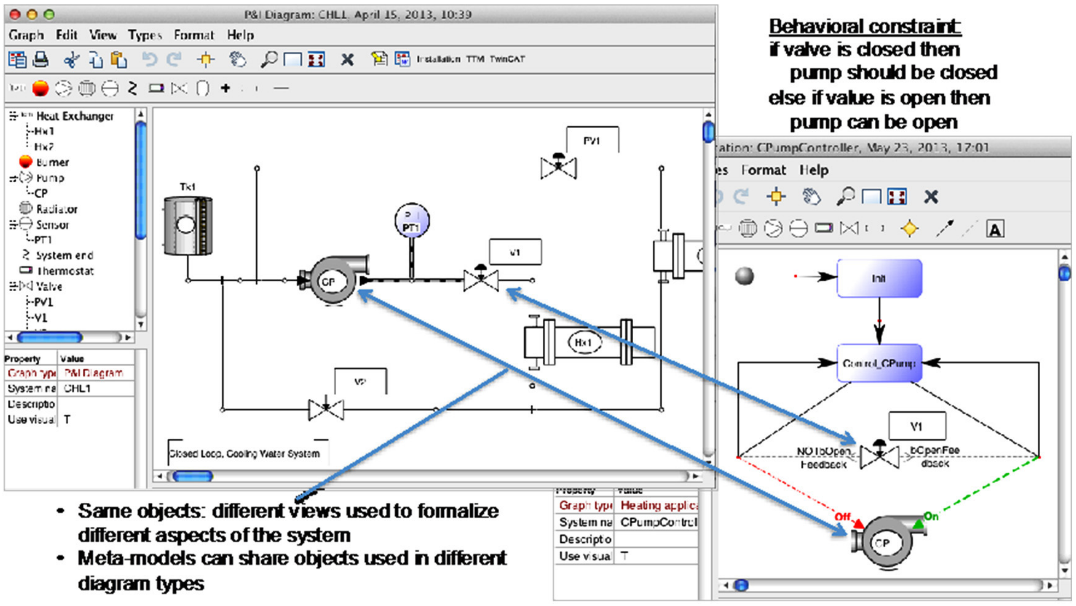


Fig. 1. Closed-loop, Heat transfer, Liquid circulating (CHL) Process Flow Diagram.

The DSM environment used for this research supports development of metamodels and associated graphics that are used to create application-specific models. Fig. 2 shows images of integrated application models' views from the prototype DSM. The DSM tool environment provides built-in generators for producing documentation, simulations, and representations to support analysis and test generation. Model concepts and properties can characterize the physical and software aspects of the target system, all of which reference the same set of objects represented. This type of approach supports evolvable modeling languages, which is critical as smart manufacturing components and technologies are evolving at a rapid pace. While DSM approaches have been used in other domains (e.g., Simulink for control engineers), they have not been used with formal methods and test generation automation to the extent provided by this research or for smart manufacturing. Manufacturing models go beyond software control and require analysis of other properties such as flow rate, pressure and temperature as described in Section 3.


P&I Diagram: CHLL April 15, 2013, 10:39
Graph Edit View Types Format Help
Heat Exchanger
Hx1
Hx2
Burner
Pump
CP
Radiator
Sensor
PT1
System end
Thermostat
Valve
PV1
V1
Closed Loop, Cooling Water System
Tk1
PV1
PT1
CP
V1
V2
Hx1
Behavioral constraint:
if valve is closed then
pump should be closed
else if valve is open then
pump can be open
Init
Control_CPump
V1
NOTisOpen Feedback
bOpenFee dback
Off
On
CP
Same objects: different views used to formalize different aspects of the system
Meta-models can share objects used in different diagram types

| Property | Value |
|---|---|
| Graph type | P&I Diagram |
| System name | CHL1 |
| Description | |
| Use visual | T |

| Property | Value |
|---|---|
| Graph type | Heating applic |
| System name | CPumpControl |
| Description | |
| Use visual | T |

Fig. 2. Integrated process flow and mechanical control views corresponding to a domain-specific metamodel for formal methods of analysis

Fig. 3 provides a high-level perspective of the prototype toolchain. The process involved the creation of metamodels (1). The prototype includes two metamodels (a), one for the process flow and another for mechanical control that shares objects, relationship, roles, etc. A diagram editor (2) uses the metamodels (a) to create application-specific models (b), which also share objects, as shown in Fig. 2. The generator (3) uses a template language (c) to extract modeled information from one or more diagrams to produce the input representation used by the formal methods tool called T-VEC (4)[6]. T-VEC is a theorem prover, which also produces analyses and other types of reports, including test vectors with associated model-to-test traceability information.

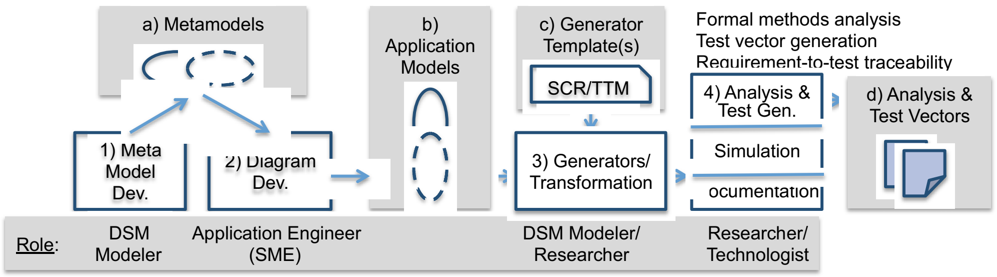
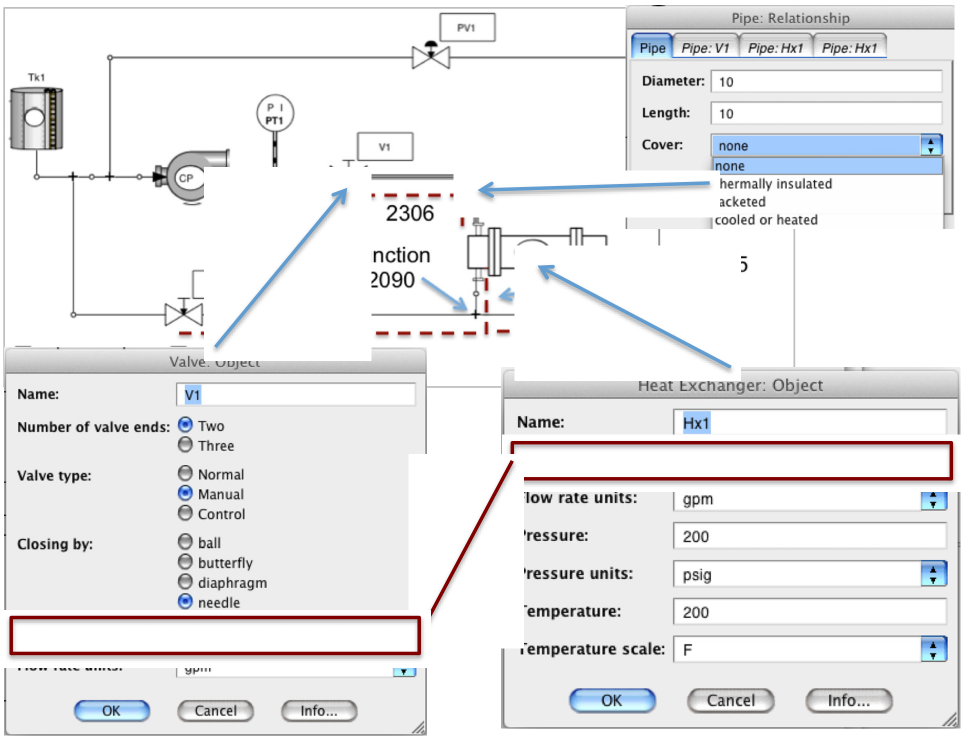


Fig. 3. Conceptual Representation of the DSM and Formal Methods Toolchain

There are at least three roles involved in the development, usage, and evolution of this type of toolchain as shown in Fig. 3. The DSM modeler creates and evolves metamodels. This requires collaboration between the DSM modeler and domain SMEs, such as process plant, mechanical controls, and safety engineers. Application engineers and SMEs use the created DSMs to produce the application-specific models. The last role involves development of generators to transform the models to a form suitable for the simulation applications, analysis and test generation. This effort can require collaboration between the DSM modeler, researchers and technology specialists.

When the design is elaborated in the process flow model, as shown in Fig. 4, a properties menu is displayed requesting the user to enter property values (e.g., flow rate, flow rate units, valve type of: Normal, Manual, or Control).



Diameter: 10
Length: 10
Cover: none
none
hermally insulated
acketed
cooled or heated

Name: V1
Number of valve ends: [✓ Two] Three
Valve type: Normal [✓ Manual] Control
Closing by: ball butterfly diaphragm [✓ needle]

Name: Hx1
low rate units: gpm
ressure: 200
ressure units: psig
emperature: 200
emperature scale: F

Fig. 4. Conceptual Design integrates Process Diagram with Formal Methods Tools

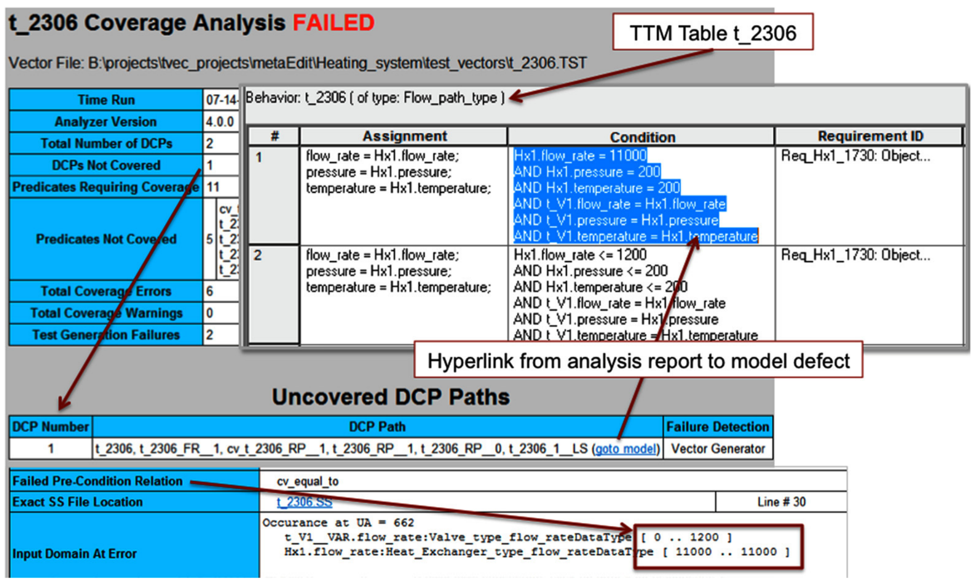
## 3. Model Representations, Analysis and Test Generation Details

A goal of the DSM approach is to hide from the user irrelevant details required to leverage formal methods. For purposes of explanation, this section describes some of these details about the DSM transformation and representations required to support formal method analysis, test generation, and requirement-to-test traceability. One example used by the project focused on flow path analysis and test generation, and uses images of translated models and analysis reports from the T-VEC tools. Future efforts are planned to bring this type of information back into the DSM environment.

The DSM generator ((3) in Fig. 3) allows the extracted model information to be formatted in a precise way syntactically and semantically. The generator includes support to produce artifacts for a TwinCAT simulator[9] and T-VEC Tabular Modeler (TTM). TTM has a modeling language based on the Naval Research Laboratory Software Cost Reduction (SCR) language[7], and has been extended to include other features for linking textual requirements, and modeling constructs such as first-order functions and assertions[6, 8]. TTM integrates with T-VEC through a translator, but this is hidden from the user; T-VEC provides formal method analysis and test vector generation. The use of TTM and T-VEC in this prototype is similar to the work of the DARPA DSM initiative called the Producible, Adaptive Model-based Software (PAMS)[3, 4].

The generator extracts information associated with DSM metamodel concepts, such as diagrams, objects, relations, roles and ports. The extracted information is transformed into TTM model constructs such as requirements, types, constants, assertions, state machines, inputs, and event and condition tables. A TTM requirement is created for every diagram, component, and pipe relation. Requirements in the TTM model are linked to the modeled information, as depicted in Fig. 5 so that when test vectors are generated, the requirement is traced and linked to the requirements providing requirement-to-test traceability. A TTM type is generated for each component based on the properties. For example, a Heat_Exchanger_type is represented as a structure that includes flow_rate, pressure, and temperature. Units are a formal property of each type. A TTM input is created for each component and associated with a type (e.g., Hx1 is of the type Heat_Exchanger_type).

A TTM table represents each flow path through the model, which is derived from each pipe that relates components. A TTM table can be thought of as a precondition and postcondition pair. The precondition describes constraints on inputs and the postcondition describes the output relations in terms of the constrained inputs. Each component has a corresponding TTM table representation with a precondition that constrains the input to the limits associated with the attributes (e.g., Hx1.flow_rate <= 11000 AND Hx1.pressure <= 200 AND Hx1.temperature <= 200) and a postcondition associated with the possible outputs for that component, based on the inputs. Each TTM table is labeled with a "t_" prefix (e.g., valve V1 has a table named t_V1). Pipes are named using the object identifier that is generated by the DSM tool. For example, the TTM table for pipe t_2306 relates valve V1 and heat exchanger Hx1 as shown in Fig. 4.



**t_2306 Coverage Analysis FAILED**

Vector File: B:\projects\tvec_projects\metaEdit\Heating_system\test_vectors\t_2306.TST

| | |
|---|---|
| Time Run | 07-14- |
| Analyzer Version | 4.0.0 |
| Total Number of DCPs | 2 |
| DCPs Not Covered | 1 |
| Predicates Requiring Coverage | 11 |
| Predicates Not Covered | 5 cv_ t_2 t_2 t_2 t_2 |
| Total Coverage Errors | 6 |
| Total Coverage Warnings | 0 |
| Test Generation Failures | 2 |

Behavior: t_2306 ( of type: Flow_path_type )

| # | Assignment | Condition | Requirement ID |
|---|---|---|---|
| 1 | flow_rate = Hx1.flow_rate;<br>pressure = Hx1.pressure;<br>temperature = Hx1.temperature; | Hx1.flow_rate = 11000<br>AND Hx1.pressure = 200<br>AND Hx1.temperature = 200<br>AND t_V1.flow_rate = Hx1.flow_rate<br>AND t_V1.pressure = Hx1.pressure<br>AND t_V1.temperature = Hx1.temperature | Req_Hx1_1730: Object... |
| 2 | flow_rate = Hx1.flow_rate;<br>pressure = Hx1.pressure;<br>temperature = Hx1.temperature; | Hx1.flow_rate <= 1200<br>AND Hx1.pressure <= 200<br>AND Hx1.temperature <= 200<br>AND t_V1.flow_rate = Hx1.flow_rate<br>AND t_V1.pressure = Hx1.pressure<br>AND t_V1.temperature = Hx1.temperature | Req_Hx1_1730: Object... |

**Uncovered DCP Paths**

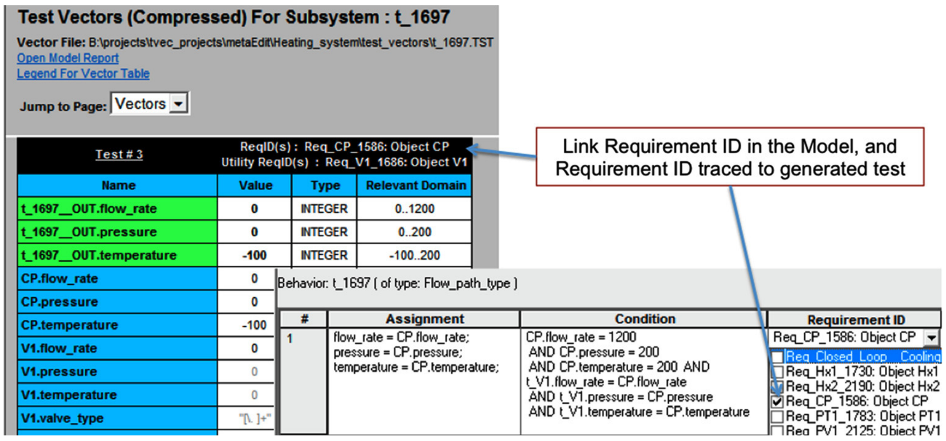
| DCP Number | DCP Path | Failure Detection |
|---|---|---|
| 1 | t_2306, t_2306_FR__1, cv_t_2306_RP__1, t_2306_RP__1, t_2306_RP__0, t_2306_1__LS (goto model) | Vector Generator |

| | | |
|---|---|---|
| Failed Pre-Condition Relation | cv_equal_to | |
| Exact SS File Location | t_2306.SS | Line # 30 |
| Input Domain At Error | Occurance at UA = 662<br>t_V1__VAR.flow_rate:Valve_type_flow_rateDataType [ 0 .. 1200 ]<br>Hx1.flow_rate:Heat_Exchanger_type_flow_rateDataType [ 11000 .. 11000 ] | |

Fig. 5. Each Pipe Has Instance Specification Plus Flow Dependencies where Flow Dependencies can be Incompatible

The DSM generator adds constraints such as a flow rate limit in the transformation of each pipe relation; these constraints support formal analysis to ensure the flow path output flow rate, pressure, and temperature of a component is not too high for downstream elements such as valves or other heat exchangers. To confirm the flow rate limit, the generator creates two disjunctions for each pipe relation (e.g., t_2306); this is represented by the two different rows in the table, as shown in Fig. 5. The Condition (i.e., precondition) describes the constraints. For example, the first row has six Boolean-valued constraints. The first three establish constraints that are used to test the boundary of the maximum value for the heat exchange H1 (e.g., Hx1.flow_rate = 11000 gps). These values are extracted from the component properties of the model. The last three constraints confirm that the transformed model associated with the valve can satisfy these constraints. The variable t_V1 is a reference to the TTM table that has the component constraints for the value V1 (i.e., V1.flow_rate <= 1200). Row #2 of the table t_2306 uses the constraint with a "<=" relation, which allows for all values between the upper and lower-bound to be satisfied. In addition, it causes both upper and low-bound values to be generated by the test vector generation system as shown in Fig. 6. The Assignment column (i.e., postcondition) is associated with the flow properties out of the pipe. The generator performs a type of transitive closure operation to ensure that every pipe and junction is related to every other pipe and component to ensure the analysis covers all of the flow paths.

The T-VEC analysis identifies those components that violate the properties, which could impact operational usage or safety. Fig. 5 shows images of three-hyperlinked report elements produced by T-VEC. The main report captures the t_2306 coverage analysis, indicating that a particular path (i.e., called a domain convergence path [DCP]) through the model is not satisfiable. This error report has a hyperlink to the highlighted Condition column for Row #1 of the t_2306 table. The image at the bottom of Fig. 5 shows that the failed pre-condition relation occurred under the condition when the flow rate for the heat exchange H1 is 11,000 gallons per minute (gpm) and the flow rate for the valve has only possible values between 0 to 1,200 gpm, which cannot satisfy the equality constraint. This is a seeded defect used in this example, but this could result from an input entry error or an actual situation where the actual valve does not have adequate capacity. Conceptually, each model path characterizes one set of component connections. This analysis is based on the assumption that it is important to know if there are any elements where this flow rate (pressure and temperature) can exceed the capacity (property) of other elements in the flow path. This example shows how a formal methods tool can identify where a requirement is not met, which can be a potential safety or operational issue that can be corrected during design.


**Test Vectors (Compressed) For Subsystem : t_1697**

**Vector File:** B:\projects\tvec_projects\metaEdit\Heating_system\test_vectors\t_1697.TST
Open Model Report
Legend For Vector Table

**Jump to Page:** Vectors

| Test # 3 | ReqID(s) : Req_CP_1586: Object CP<br>Utility ReqID(s) : Req_V1_1686: Object V1 | | |
|---|---|---|---|
| **Name** | **Value** | **Type** | **Relevant Domain** |
| t_1697__OUT.flow_rate | 0 | INTEGER | 0..1200 |
| t_1697__OUT.pressure | 0 | INTEGER | 0..200 |
| t_1697__OUT.temperature | -100 | INTEGER | -100..200 |
| CP.flow_rate | 0 | | |
| CP.pressure | 0 | | |
| CP.temperature | -100 | | |
| V1.flow_rate | 0 | | |
| V1.pressure | 0 | | |
| V1.temperature | 0 | | |
| V1.valve_type | "[\. ]+" | | |

Behavior: t_1697 ( of type: Flow_path_type )

| # | Assignment | Condition | Requirement ID |
|---|---|---|---|
| 1 | flow_rate = CP.flow_rate;<br>pressure = CP.pressure;<br>temperature = CP.temperature; | CP.flow_rate = 1200<br>AND CP.pressure = 200<br>AND CP.temperature = 200 AND<br>t_V1.flow_rate = CP.flow_rate<br>AND t_V1.pressure = CP.pressure<br>AND t_V1.temperature = CP.temperature | Req_CP_1586: Object CP<br>☐ Req_Closed_Loop_Cooling<br>☐ Req_Hx1_1730: Object Hx1<br>☐ Req_Hx2_2190: Object Hx2<br>☑ Req_CP_1586: Object CP<br>☐ Req_PT1_1783: Object PT1<br>☐ Req_PV1_2125: Object PV1 |

Link Requirement ID in the Model, and Requirement ID traced to generated test

Fig. 6. Generated Test Vector with Requirement Traceability

When the seeded defect is corrected by changing the Hx1 heat exchanger flow rate to 1200 gps, the model errors are resolved. There were a total of 36 test vectors produced for all of the paths through the pipes and components. An example test vector, with requirement traceability is shown in Fig. 6 for the pipe relation t_1697. As shown in the Requirement ID column of table t_1697, the model generation associates the requirement link with the generated TTM tables. For example, the requirement includes both the pump object CP and the valve object V1. During the test vector generation process, the linked requirements are output and associated with the corresponding test vector to provide requirement-to-test traceability. A test vector includes the expected outputs and test input values, type, and associated domain.

## 4. Conclusions

This paper describes a research project to support virtual design and verification of industrial process plants' designs. The prototype developed for this project uses DSMs and DSMLs of differing views of a system design, and provides examples of how the integration with formal methods can identify defects in the design, and automatically generate test vectors with requirement-to-test traceability.

The project research involved three main roles: 1) developing the DSM metamodel for integrated system designs, 2) creating application-specific models, using two graphical DSMLs, and 3) producing the generator required to demonstrate analysis and test generation. This type of DSM-based toolchain approach supports model evolution when new technologies are created in the domain (e.g., cyber-physical systems). This is particularly important as new technologies for smart manufacturing, especially those with computational intelligence are emerging at an accelerating rate. When new or enhanced products or systems are created, they can be characterized in the metamodel(s), and then are available in the application editors for users of the system. Updates to the metamodel will often require updates to the generators to support analysis for these new types of enhanced capabilities.

While it is possible to find P&ID tools to create these types of diagrams, there are significant advances needed in the industry to support a broad range of formal analysis. Further, the research is focused on viewpoint integration and leveraging appropriate types of formal methods to address the various types of complexity in these systems. The need for such integration is relevant to CPS in general.

It is too early to quantify the efficiencies and benefits of this approach for smart manufacturing. However, the conceptual approach proposed for this project is similar to the DARPA Disruptive Manufacturing Technologies PAMS initiative[3]. This same pattern was used on the PAMS project for an avionic flight control DSM, where quantitative comparisons against legacy development processes indicate greater than 60 percent reduction in development time and greater than 80 percent reduction in lifecycle cost[4]. These efforts represent the tip of the iceberg, as there are many other types of analyses needed for CPS. Future efforts will continue to extend the V&V support and work to create a better veneer to increase the usability that integrates the modeling and analysis information.

This integrated prototype provides a higher-level representation that leverages two types of transformations to provide engineers a more natural interface for performing process plant engineering and getting rigorous analysis at design time. A key benefit of the DSM approach is that it raises the level of abstraction and hides details that are embedded within the generators. Finally, these efforts reflect how this DSM approach can support new contributions and collaboration made by researchers and technology specialists across many domains where they can more easily leverage and integrate advanced analysis and computer automation through model-integrated platforms.

## References

1. Foundations for Innovation in Cyber-Physical Systems, http://events.energetics.com/NIST-CPSWorkshop/index.html.
2. Matei, I., C. Bock, SysML Extension for Dynamical System Simulation Tools, National Institute of Standards and Technology, NISTIR 7888, http://dx.doi.org/10.6028/NIST.IR.7888, October 2012, http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7888.pdf
3. DARPA, Producible Adaptive Model-based Software (PAMS) technology to the development of safety critical flight control software. PAMS has been developed under the Defense Advanced Research Projects Agency (DARPA) Disruptive Manufacturing Technologies program. Contract # N00178-07-C-2011, http://www.isis.vanderbilt.edu/projects/PAMS
4. Ray, S., G. Karsai, K. McNeil, Model-Based Adaptation of Flight-Critical Systems, Digital Avionics Systems Conference, 2009.
5. MetaEdit+ Tool Suite, http://www.metacase.com/.
6. T-Vec Tool Suite, http://www.t-vec.com/.
7. Software Cost Reduction (SCR) Toolset http://www.nrl.navy.mil/itd/chacs/5546/SCR
8. Blackburn, M., R. Busser, A. Nauman, T. Morgan, Life cycle integration use of model-based testing tools, Digital Avionics Systems Conference, 2005.
9. TwinCAT Suite, http://www.beckhoff.com/.

[i] References to proprietary products are included in this paper solely to identify the tools actually used in the industrial applications. The identification does not imply any recommendation or endorsement by NIST as to the actual suitability of the product for the purpose.