24th Annual INCOSE International Symposium (IS2015)
Seattle, WA, July 10 – 16, 2015

Utilizing MBSE Patterns to Accelerate System Verification

David Cook
Moog Aircraft Group
dcook@moog.com

William D. Schindel
ICTT System Sciences
schindel@ictt.com

Abstract. In aerospace, automotive, health care, and other domains, the capability to effectively plan and perform system verification tests is increasingly a strategic differentiator. This paper reports on methods used to improve effectiveness of complex aircraft system tests, the gains achieved, and their connection to underlying methods and trends. These include use of Model-Based Systems Engineering (MBSE), leveraged by Pattern-Based Systems Engineering (PBSE) to generate configurable, re-usable system models--the focus of the INCOSE MBSE Patterns Challenge Team. (PBSE Team, 2014)

Distinctive aspects include (1) use of configurable, model-based patterns of system descriptions, including configurable system verification tests, (2) improved ability of model-based descriptions to integrate modeling tools and automated systems for test management, (3) use of models to describe both system under development and the development process, and (4) ability of these models to be integrated with other system life cycle management processes and information systems, for increased leverage.

I. Introduction

Enterprises involved in the development of technological systems have significant intellectual capital that has been accumulated through the execution of numerous development projects. Regardless of the methods employed, this existing knowledge is a key asset in the development of new systems. The utilization of model-based system patterns and Pattern Based Systems Engineering processes provide a disciplined and systematic approach to maximize the effective use of this intellectual capital in the development of new systems. Coupled with the capabilities of modern MBSE tools and techniques, the application of pattern based MBSE holds the promise of significant improvements in the development of modern technological systems.

This paper examines the application of model based system patterns to accelerate the verification of modern technological systems. The particular example for chosen for this paper is the testing of a safety critical aircraft subsystem, namely the flight control actuation system.

II. Challenges and Opportunities

It is understandable that the development of safety critical systems tends to be very risk averse. Processes for the development of safety critical aircraft systems are subject to scrutiny by the aircraft manufacturers (customers) and certification authorities to ensure that the development processes do not introduce unintended hazards into the system. For safety critical systems, like flight control actuation, modeling provides the ability to improve the requirements quality and overall system design integrity. However, regardless of the design rigor, extensive verification (primarily testing or combination of analysis and test) is required to verify the system meets the functional and safety related requirements. The effort required to develop and execute a rigorous test program results in significant development cost. Testing cost can be 20% to 50% of development effort (excluding hardware manufacturing costs) in many cases. Cutting testing

costs without sacrificing test effectiveness can result in significant competitive advantages. (Cook, 2014)

A primary goal of MBSE is to utilize the system models to capture and communicate all system related information in a clear and unambiguous fashion. If this can be accomplished with models then the documents used in traditional systems engineering are no longer necessary as the prime repository of information—they instead become snapshot artifacts that are views of the primary model database at the times of key milestones. The system model supersedes all the previous document artifacts as the most fundamental representation, but this model can be made to formally include model-based prose requirements and similar structures (Schindel, 2005). The current state of the industry and the extensive formal documentation associated with safety critical aircraft systems necessitate the use of conventional documents for many of the system development artifacts. As such, any MBSE implementation must be compatible with an environment where conventional engineering documentation is required. For verification tests, this includes formal test procedure and report documents.

III. MBSE and Patterns

III.1 MBSE Improves Basic Representation

Model-Based Systems Engineering (MBSE) changes and improves how we represent systems (Schindel, 2014). This applies to more than just the Product System to be developed. Benefits can also be gained from explicit models of different systems across the Product System's life cycle, potentially including:

- The Product System itself, along with aspects of the larger Application Domain System in which it will serve (*)
- The Test System that will perform formal verification tests (*)
- Other aspects of the System of Development (e.g., the ISO15288 Technical Processes)
- The Manufacturing System (the System of Production) that instantiates the Product
- The Distribution System (including installation) that distributes the Product
- The Operations System that helps support in-service use of the Product
- The Maintenance System that helps maintain the Product
- The Decommissioning and Disposal System

Those marked (*) above are the focus of this paper.

Basic benefits of using MBSE to represent a Product System under development are becoming better known. For example, when compared to earlier traditional prose approaches, model-based requirements are generally:

- more explicit, less ambiguous, more likely to be interpreted consistently;
- easier to audit for completeness and correctness;
- more objectively interpreted for planning of test.

(Schindel, 2005, 2014)

These general benefits apply across a variety of contemporary modeling languages and supporting tools. They are amplified when applied using the underlying S*Metamodel of Figure 1 (Schindel, 2014). Such S*Models associate system requirements with physical Interactions (Schindel, 2013), treat requirements as parameterized "transfer functions" (Schindel, 2005), and otherwise conform to that framework. Beyond the aerospace example of Section IV, these techniques are currently being demonstrated by the INCOSE Patterns Challenge Team, across multiple languages, tools, and domains (PBSE Team, 2014).

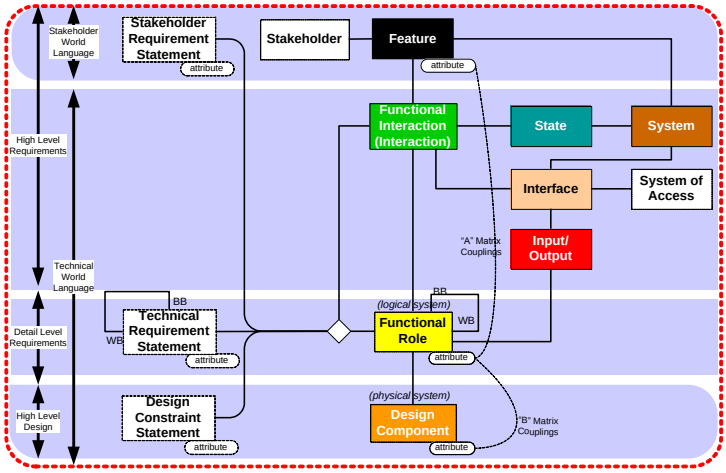

Figure 1: Underlying S*Metamodel Summary, Language & Tool Independent

III.2 MBSE Improves Test Representation

When MBSE is based on the S*Metamodel, all system requirements are seen to be described in the context of physical Interactions (exchanges of energy, force, mass, or information) with external domain actors, and each of the interacting entities has assigned Functional Roles that are modeled. As shown in Figure 2, this means that not only the requirements on the system under test are captured—so also are the related requirements (which could be called assumed behaviors) of the external systems. This has several advantages, but for purposes of this paper a key advantage is that it provides candidate behaviors of the Test System that will exercise the system under test. Having these role-based behaviors linked together as Interactions in the model from the outset reduces the effort of coordinating the test protocol with the requirements being tested. Since the Test System should also measure and capture the behavior of the System Under Test, this also provides an indication of expected behavior during test, for comparison.

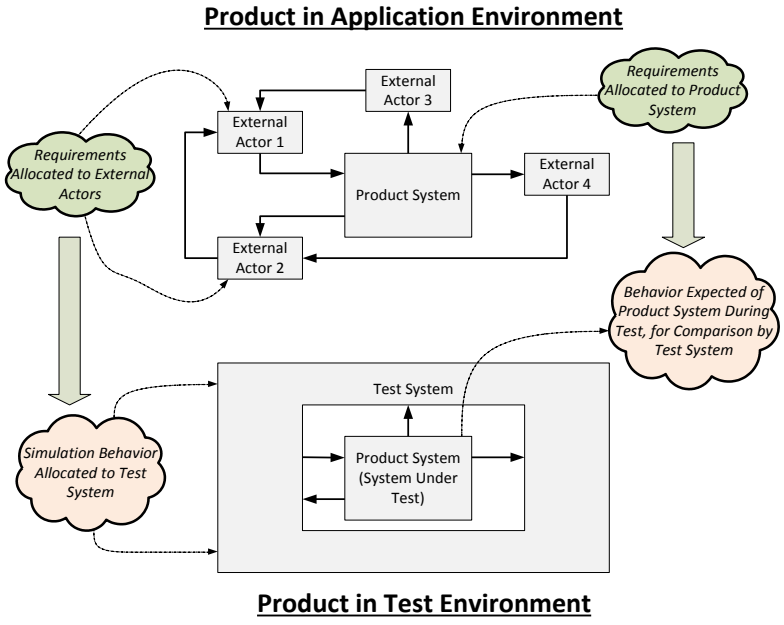

Figure 2: Models Describe In-Service Application Domain, As Well As its Test Simulation

In the context of Figure 2, there are several opportunities to use an integrated model:

1. To specify requirements of the Product System
2. To partition the overall Test Cases and specify the Test System behaviors in each
3. To specify the expected behavior of the System Under Test, for comparison (related to (1) above, but possibly also configured further for purposes of test versus service)
4. To specify the reports to be produced by the Test System

The above could be of some value if the Test System were completely manual (human performed), but becomes much more valuable if the Test System includes automation that can be driven directly by the MBSE models. This is illustrated in Section IV.

III.3 PBSE: Configurable Models Magnify Leverage and IP

When a systems enterprise focuses ongoing work in product lines or platform systems of a particular domain (e.g., automotive, aerospace, medical/health care, telecom systems, consumer products, etc.), the above gains may be further leveraged. The system models described in III.1-2 may be arranged to describe a general family of systems in a re-usable, configurable fashion. When S*Models described above are arranged in this way, they are referred to as S*Patterns (Peterson and Schindel, 2013; Patterns Team 2013-14). These can describe not only the Product System family, but also the other systems listed in Section III.1, and used for the test purposes described in Section III.2.

Figure 3 illustrates the idea that such S*Patterns describe abstract families of systems, using the same S*Metamodel. These system patterns may be specialized and configured for specific projects, as part of the development, verification, and other system life cycle processes.

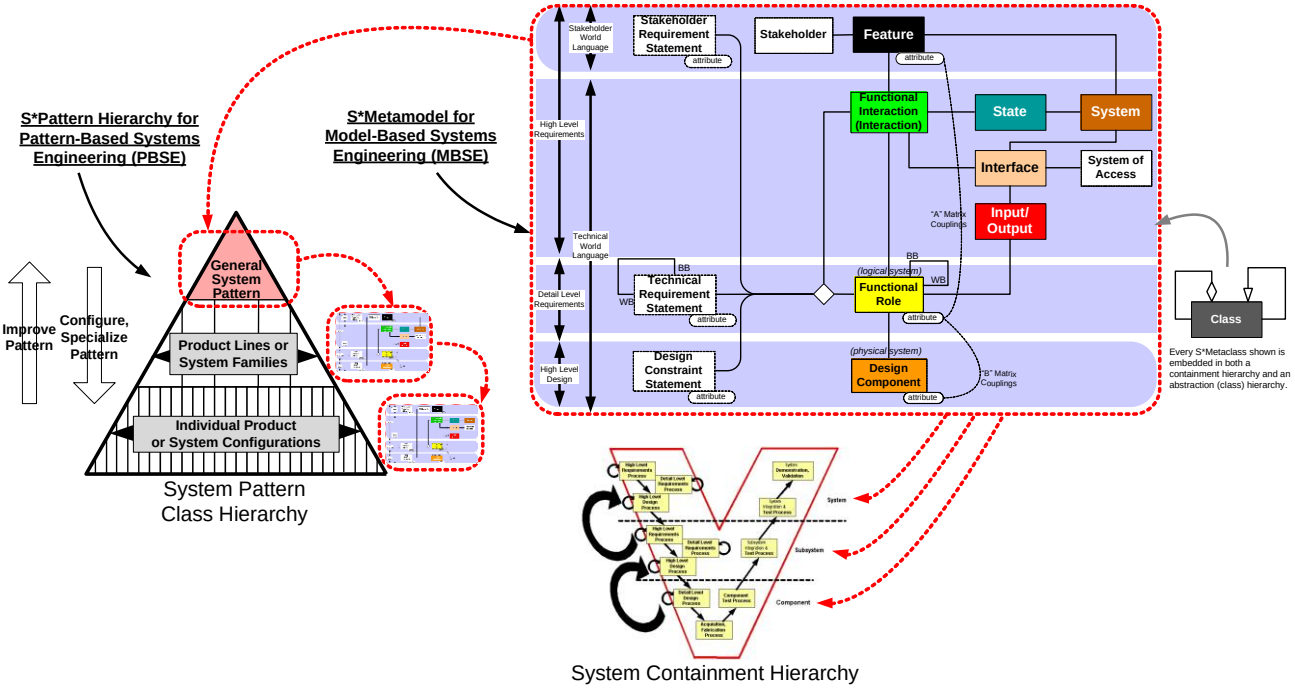

Figure 3: S*Patterns Are Re-usable, Configurable S*Models

In the case of S*Patterns, configurability is driven by the Stakeholder Features portion of the applicable patterns. This reflects the idea that the only reason for configuring a general system pattern to a specific configuration is to deliver specific stakeholder-realizable benefits or outcomes. For this reason, the delivery process of configuring the Features portion of an S*Pattern can be used to semi-automatically generate configured versions system requirements, system tests, and other configured models (Peterson and Schindel, 2013). Furthermore, if the Test System is arranged to be driven (or configurable) by the configured Test System Pattern, then additional benefits of test protocol development and validation may be gained.

The application of these ideas to system verification test is illustrated in Section IV.

IV. Verification Application Examples

To realize the maximum effectiveness in the application of MBSE, the processes, tools, and methods have to be engineered in much the same way that the product system is engineered. There can generally be specific test needs based on the type of system being tested. The system in this example is a safety critical flight control actuation system for modern aircraft. In this example, the test platform and associated tool chain were selected to support the overall MBSE workflow. The combination of development and test tools were selected to optimize effectiveness over the entire life cycle. In much the same way that a globally optimized system is not necessarily made up of locally optimized system elements, a globally optimized development tool chain is not necessarily made of locally optimal tools. A depiction of the example workflow is shown in Figure 4. In the depicted workflow, the start is with System Design followed by Real-Time Simulation, then Prototype/Integration Testing, and finally Formal System Testing.

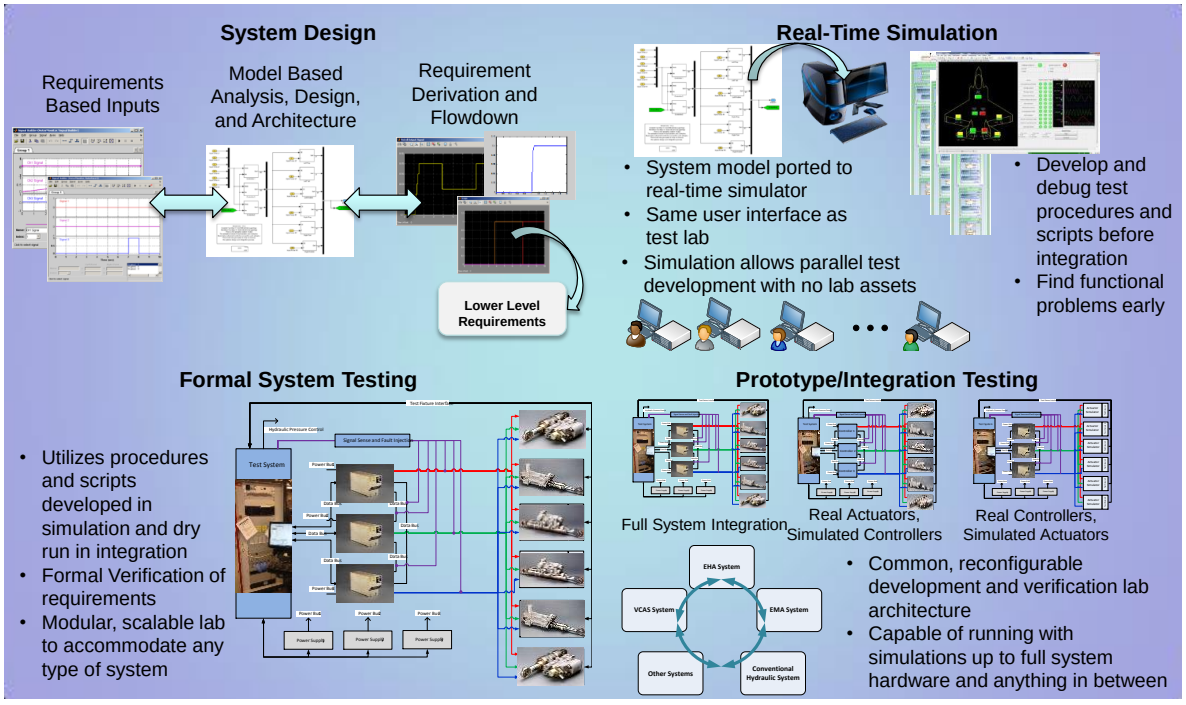

Figure 4. Example MBSE Workflow

Traditionally, test procedures are developed based on the system requirements and the test engineer's understanding of the test system functionality. The procedures are then implemented using test automation software (generally a test scripting language). The automation script would then be debugged in the test lab. Generally iterations of the procedure

and script would be required to arrive at test that is ready for formal verification testing. In the newer paradigm, the goal of the Real-Time Simulation step in the MBSE workflow is to utilize the system model to develop and debug system tests while utilizing the same user and automation environment as the test lab. This allows tests to be developed without the need for a test lab.

Further benefits are realized by applying system patterns to the system testing process. That is, there are repeatable patterns in both the system under test and the testing system. A representation of a portion of a generic testing pattern is shown in Figure 5. Once the pattern configuration is complete, a test procedure suitable for customer review and an automated script that can be executed in the physical test lab are automatically generated. The automation script also analyzes the test data and generates an automated test report.

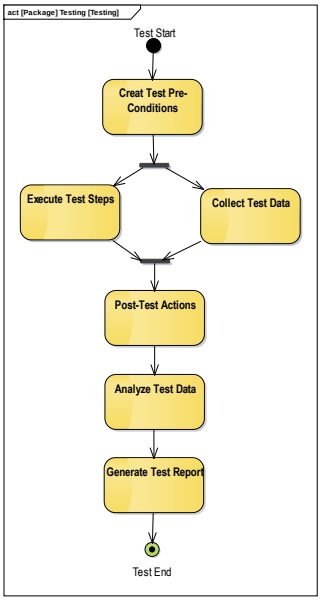

Figure 5. Portion of Generalized Testing Process Pattern

In implementing the pattern, there were two subcategories of testing patterns that were developed for the actuation system testing. The first type of pattern, referred to as template tests, involves tests that are run in basically the same fashion for each different type of system. The template test patterns are developed with configurable parameters to facilitate use with different types of systems. For actuation systems these tests tended to performance measurement and similar tests. The second type of pattern, referred to as vector tests, is more generic and has broader potential application. This is a generic pattern where system inputs are defined as a (configurable) function of time (time-based vectors). Expected results are also defined as a function of time. Expected results are defined as upper and lower bounds on the pertinent system outputs. To perform the test, the vectors of inputs vs. time are applied to the system. The pertinent system outputs are recorded and then compared against the defined bounds to determine a pass/fail result.

Taking advantage of the system model, vector test development begins in the desktop modeling environment. To verify a particular requirement, the input and expected output vectors are defined as time based vectors that are applied to the system simulation. Simulated outputs are checked against the expected outputs to determine if the designed system satisfies the given requirement. This capability to test in simulation provides a measure of validation for

the design and associated derived requirements. Once the test is developed and debugged in simulation, the test vectors are converted for execution in the real-time environment and the test lab. This conversion is a simple formatting change to put the input and expected output vectors in a format compatible the real-time simulation and test lab environments. The automation capability of the tool chain makes it possible to go from simulation on a desk top computer to execution on the physical system in a matter of seconds. This workflow is demonstrated for an actuator fault injection test in the following section.

IV.1 A Safety Critical System Verification Example

Safety critical systems typically have architectures and design features that are intended to allow safe operation in the presence of system hazards. One of the more significant hazards for flight control actuation systems is the loss of control of an actuator. One such failure mode for a hydraulic actuator is servo valve hardover failure (failure at max output). This failure causes the actuator to move at maximum velocity until it contacts the actuator mechanical travel limits. The result is an aircraft with a control surface fixed at maximum deflection which typically has catastrophic results for the vehicle. Due to the serious effects of the failure, the system design includes functionality to detect and mitigate failures such as this. For this example a test is developed to test the fault response of a rudder actuator to a servo valve hardover failure. For this test, the key system requirements are:

1. The FCAS shall limit surface transients to less 5% of the actuator stroke.
2. The FCAS shall transition a failed surface to the fail-safe mode in less than 250 msec.
3. The FCAS shall annunciate failures to the flight control computer.

FCAS = Flight Control Actuation System

The system model is developed to include the simulation of key fault conditions. Once the model fidelity is sufficient to represent the system design and the associated fault conditions, a test approach is developed. A set of system input and expected output vectors are defined based on the test approach. The system inputs and expected outputs for this example are shown in Figure 6.

Once the test vectors are defined, additional meta-data is associated with the test. Test objectives and a brief description of the test methodology is created and associated with the test vectors. This meta-data aids in review and maintenance of the tests. Links are also created to link the test to the requirements in the requirements database.

When the test is executed in simulation, the system response is compared to the expected outputs. The simulated test results are displayed and archived automatically using the tool automation features. An example of the simulated test results is shown in Figure 7.

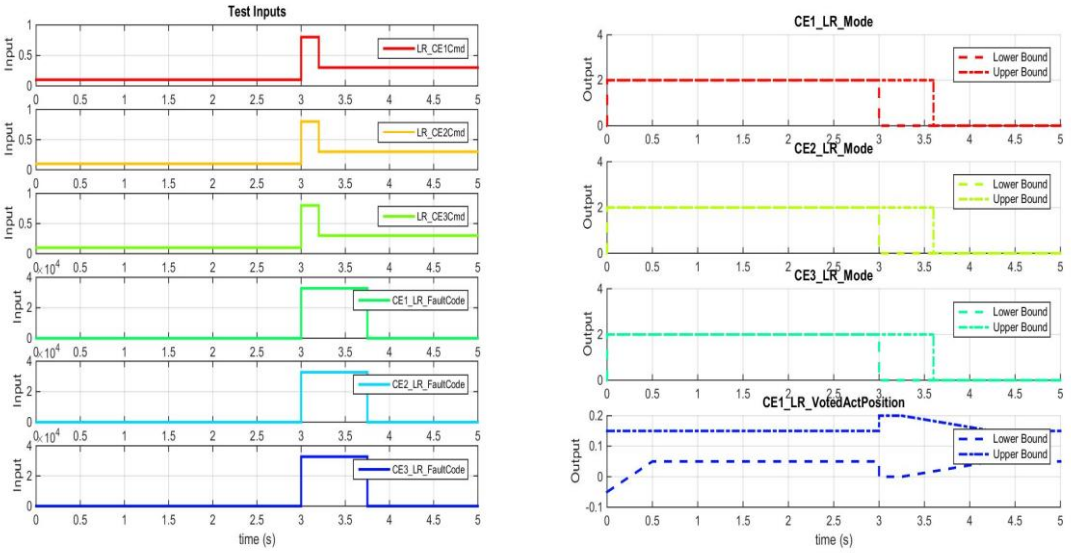

Figure 6. System Input/Expected Output Vectors for Hardover Failure Test

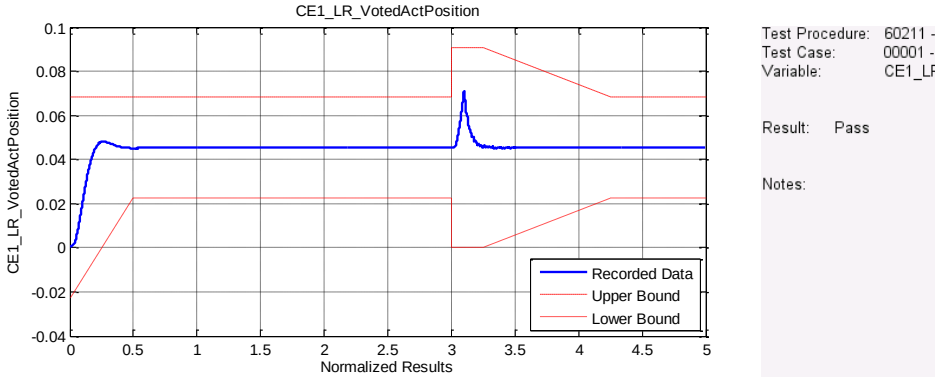

Figure 7. Results for Simulated Hardover Failure Test

Once the test definition is complete, a test procedure is automatically generated using the test vectors, test meta-data, and the linked requirements. A snapshot of the automated procedure is shown in Figure 8.

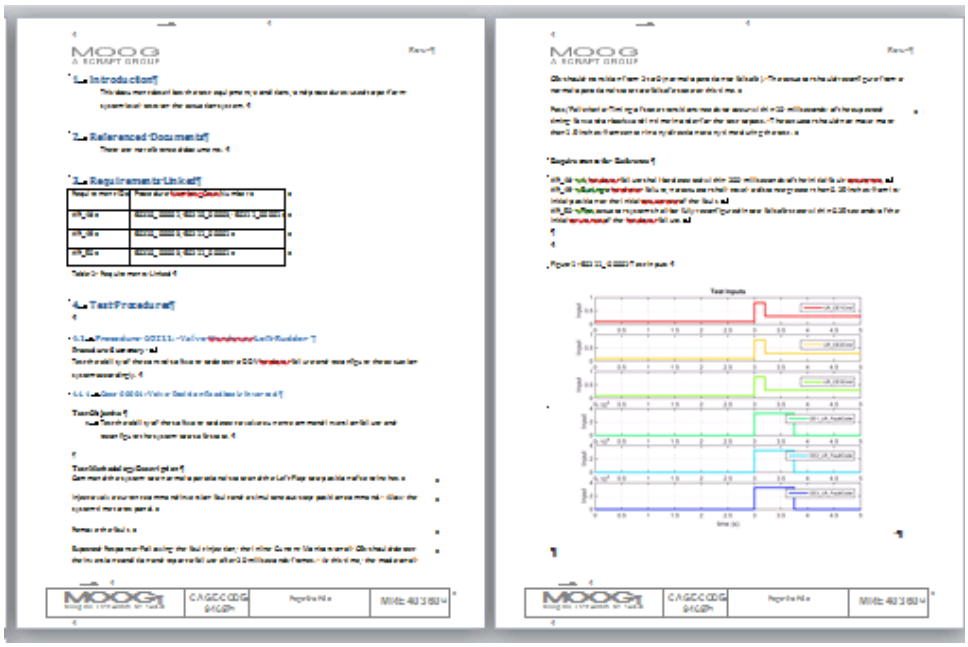

Figure 8. Auto-Generated Procedure Excerpt for Hardover Failure Test

When the test is ready to be executed in real-time simulation or the system test lab, the test is converted into a format that is compatible with the test environment. The test is executed by a script that utilizes the test vectors for real-time execution. Data is collected and compared against the expected outputs. An example excerpt from the auto-generated test report is shown in Figure 9.

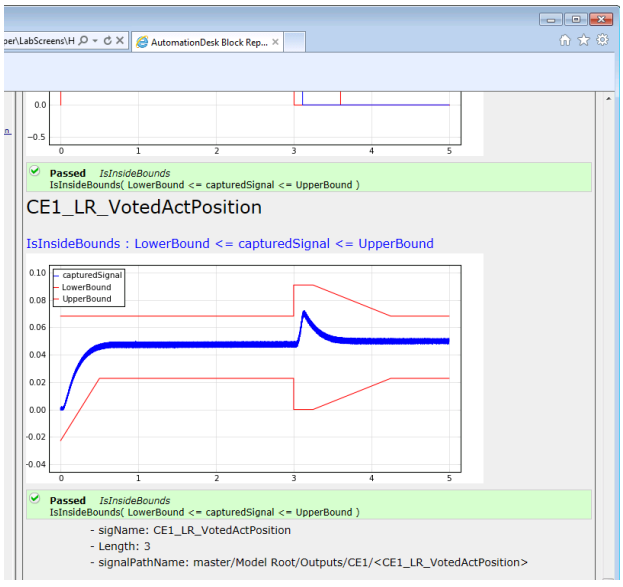

Figure 9. Auto-Generated Test Report Excerpt for Hardover Failure Test

*IV.2 Representation Using Formal S*Patterns*

Both re-usability and configurability are maximized when both the Product System and the Test System of the above example are represented by configurable S*Patterns:

- The Product System Pattern represents the general class of product systems for the developer enterprise, and this may be specialized to individual product line platforms, and then configured for specific project deliverables.

- The Test System Pattern represents the general test system (including automated as well as manual aspects) of the developer enterprise, and may be specialized to individual test platforms and further configured to specific project tests.

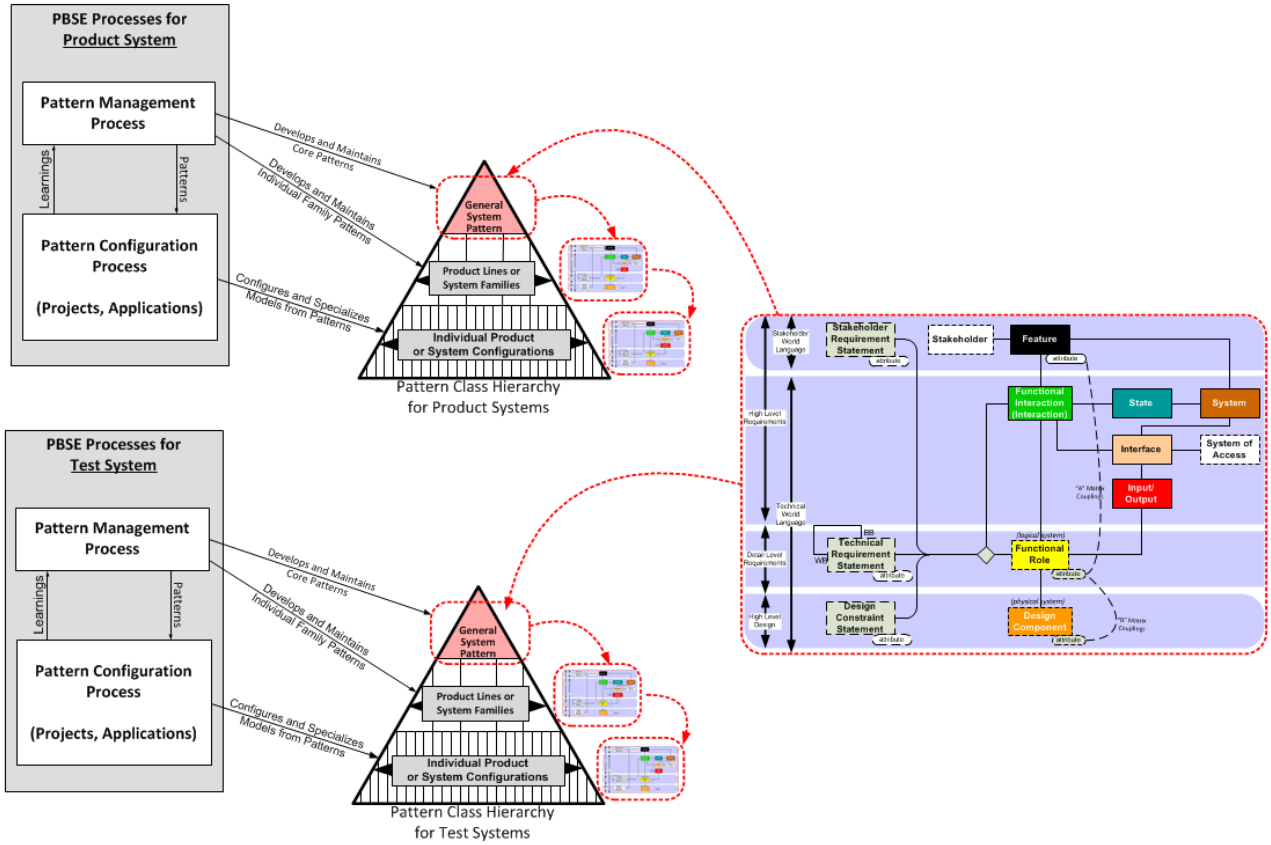

Figure 10. Product System Pattern and Test System Pattern

Each of these S*Patterns consists of generalized configurable models that conform to the S*Metamodel of Figure 1. Figures 11 and 12 are extracts of the ISO15288 model portion of the System of Innovation Pattern, with Figure 12 focused on its Verification Process, itself driven by patterns. Example extracts of the Feature and Interaction portions of these two patterns are summarized in Figure 13.

During a project, the applicable System Pattern is configured for the specific product and product test plans. The main input to this configuration is the configuration of the applicable Features from the pattern. Figure 14 illustrates a typical feature configuration process user interface, which can be attached to any third party toolset used as the pattern repository.

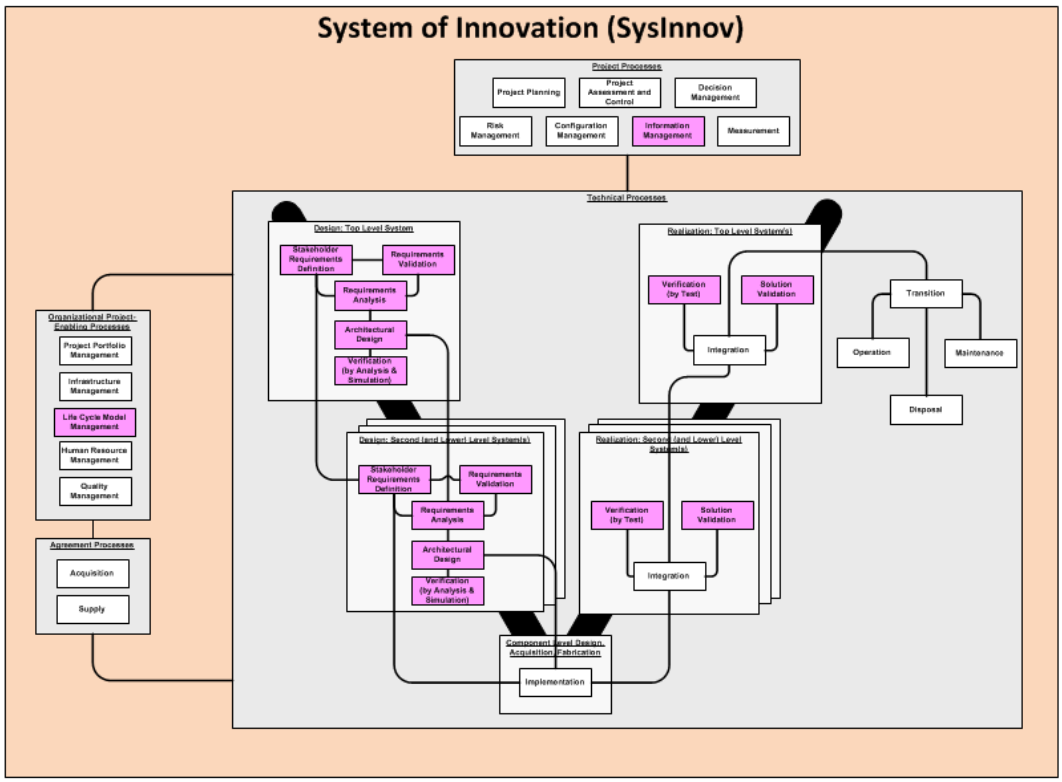

Figure 11. From the System of Innovation Pattern: ISO15288 Technical Processes

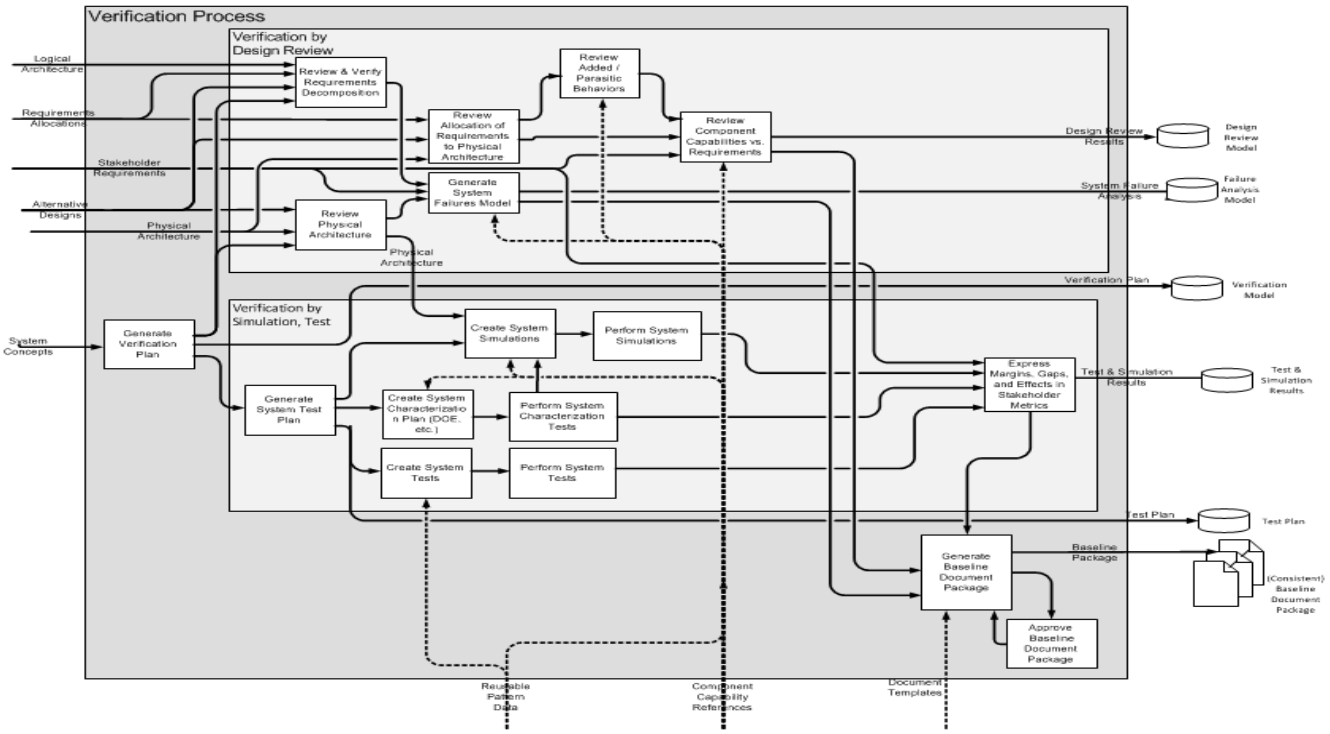

Figure 12. From the System of Innovation Pattern: Verification Process, Pattern-Driven

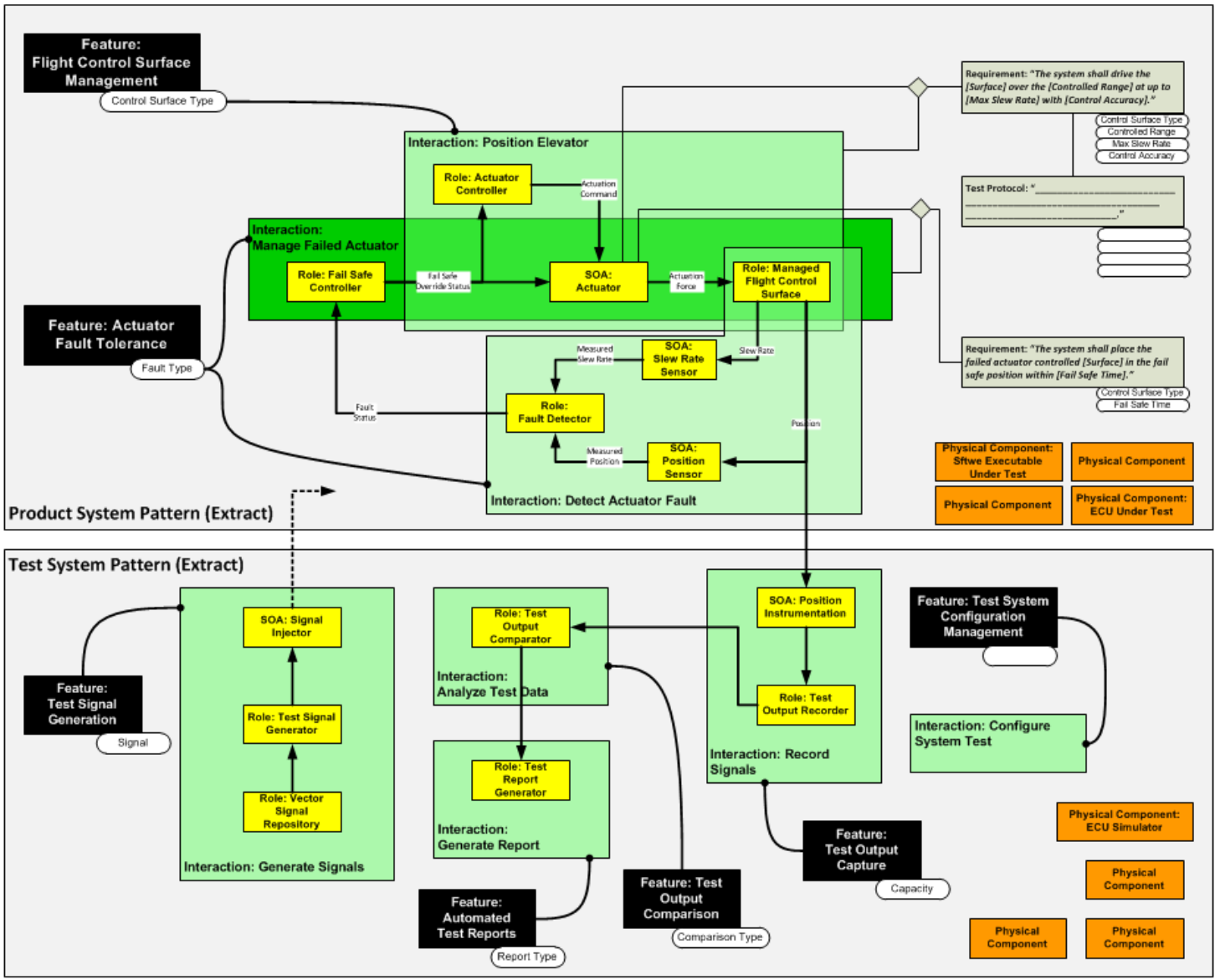

Figure 13. Selected Extracts from Product Pattern and Test System Pattern

Mandatory, Optional, or Other Configuration Rule	Populate? (YES/NO)	Feature Name	Feature Attribute Primary Key (PK) Attribute Name	Feature Attribute PK Value #1	Feature Attribute PK Value #2	Feature Attribute PK Value #3	Feature Attribute PK Value #4	Feature Attribute PK Value #5
Optional	YES	**Test Signal Generator Feature**	Signal Stream ID	CE1_LR_Voted ActPosition	CE3_RRA_Vote d ActPosition	CE5_BA_Voted ActPosition		
Optional	YES	**Automated Test Reports Feature**	Report Type	Auto Desk Blk Rept	Test Case Summary			
Optional	YES	**Test Output Comparison Feature**	Comparison ID	CE2_LR_Mode	CE1_LR_			
Optional	YES	**Test Output Capture Feature**	Output Stream ID	CE2_LR_Mode	CE1_LR_			
Mandatory	YES	**Test System Configuration Management Feature**	System Test ID	Voting Test Case 1	Voting Test Case 2	Voting Test Case 3	Voting Test Case 4	Voting Test Case 5
Optional	YES	**Test Procedure Auto Generation Feature**	Procedure Format Type	Procedure Summary	Procedure Detail			

CE1_LR_Voted ActPosition
CE2_LRA_Voted ActPositio
CE3_RRA_Voted ActPositio
CE4_RRA_Voted ActPositio
CE5_BA_Voted ActPosition

Figure 14. Configuring the Pattern Features for Specific Product and Test Systems

IV.3 Initial Gains Experienced

The application of MBSE patterns to system verification efforts can yield significant improvements in the effectiveness and efficiency of the testing program. The benefits realized through the presented workflow implementation are:

- Scalability of test development. The ability to develop and debug tests and the automated scripts independent of the system test lab means that test development can be economically parallelized.
- Test development in simulation. Verification tests can be developed independent of test hardware. This eliminates the test lab as a potential bottleneck in the development of verification tests.
- Focus on technical content. Engineers focus on test content and system responses rather than the administrative tasks of generating tests, test procedures, and test reports. This process also eliminates much of the test rework effort because tests can be effectively debugged using the system model.
- Rapid test development. Tests can be implemented quickly in simulation then converted to run in real-time on the test system in seconds.

The presented application of MBSE patterns as presented in this paper are expected to reduce integration and test effort by more than 25% for the testing of flight control actuation systems.

IV.4 Future Applications and Extensions

Future extensions include the further expansion of the system patterns into the design and modeling of the product system. The goal here is to expand the approach developed and presented here into the analysis and design phases of development. Plans are to demonstrate and mature the patterns for portions of the system and expand the pattern progressively over time while continuing to deliver increased value to current development efforts.

V. Summary and Conclusions

The application of Pattern-Based Systems Engineering (PBSE) promises to improve the efficiency of system development efforts by leveraging existing intellectual capital in new development efforts. The application of S* Patterns provide a robust framework for effectively

creating effective system patterns. The application of MBSE tools and techniques can help to realize greater pattern value. This paper presents the application of MBSE system patterns to the testing of flight control actuation systems and associated benefits. The presented approach holds promise for testing of similar technological systems.

While significant benefits have been realized from the current MBSE implementation, the potential exists for significant additional gains. The further application of S* patterns to similar systems is one of the goals of the INCOSE PBSE Challenge Team.

References

1. Cook, David. "Electronics Take Off: Real-Time Testing of Modern Actuation Systems at Moog", dSpace Magazine, Paderborn, Germany, January, 2014.
2. W. Schindel, "Maps or Itineraries? A Systems Engineering Insight from Ancient Navigators", to appear in Proceedings of the INCOSE 2015 International Symposium, Seattle, WA, July, 2015.
3. W. Schindel, "System Life Cycle Trajectories: Tracking Innovation Paths Using System DNA", to appear in Proceedings of the INCOSE 2015 International Symposium, Seattle, WA, July, 2015.
4. W. Schindel, S. Lewis, J. Sherey, S. Sanyal, "Accelerating MBSE Impacts Across the Enterprise: Model-Based S*Patterns", to appear in Proceedings of the INCOSE 2015 International Symposium, Seattle, WA, July, 2015.
5. W. Schindel, "Requirements statements are transfer functions: An insight from model-based systems engineering", Proceedings of INCOSE 2005 International Symposium, (2005).
6. W. Schindel, "What Is the Smallest Model of a System?", Proc. of the INCOSE 2011 International Symposium, International Council on Systems Engineering (2011).
7. Schindel, W. "The Difference Between Whole-System Patterns and Component Patterns: Managing Platforms and Domain Systems Using PBSE", INCOSE Great Lakes Regional Conference on Systems Engineering, Schaumburg, IL, October, 2014
8. Schindel, W. "Interactions: At the Heart of Systems", INCOSE Great Lakes Regional Conference on Systems Engineering, W. Lafayette, IN, October, 2013.
9. W. Schindel, and V. Smith, "Results of applying a families-of-systems approach to systems engineering of product line families", SAE International, Technical Report 2002-01-3086 (2002).
10. W. Schindel, "Pattern-Based Systems Engineering: An Extension of Model-Based SE", INCOSE IS2005 Tutorial TIES 4, (2005).
11. J. Bradley, M. Hughes, and W. Schindel, "Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns" Proceedings of the INCOSE 2010 International Symposium (2010).
12. W. Schindel, "The Impact of 'Dark Patterns' On Uncertainty: Enhancing Adaptability In The Systems World", in Proc. of INCOSE Great Lakes 2011 Regional Conference on Systems Engineering, Dearborn, MI, 2011

13. W. Schindel, "Introduction to Pattern-Based Systems Engineering (PBSE)", INCOSE Finger Lakes Chapter Webinar, April 26, 2012.
14. INCOSE/OMG MBSE Initiative: Patterns Challenge Team 2013-14 Web Site: http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
15. Bill Schindel, Troy Peterson, "Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques", in Proc. of INCOSE 2013 International Symposium, Tutorial, June, 2013.
16. Eric Berg, "Affordable Systems Engineering: An Application of Model-Based System Patterns To Consumer Packaged Goods Products, Manufacturing, and Distribution", at INCOSE IW2014 MBSE Workshop, 2014.
17. ISO/IEC 15288: Systems Engineering—System Life Cycle Processes. International Standards Organization (2014).
18. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 4, International Council on Systems Engineering (2014).
19. ISO/IEC 26550 "Software and Systems Engineering—Reference Model for Product Line Engineering and Management", 2013.
20. ISO/IEC42010 "Systems and Software Engineering—Architecture Description", 2011.
21. Rick Dove, Ralph LaBarge, "Fundamentals of Agile Systems Engineering—Part 1" and "Part 2", INCOSE IS2014, July, 2014.
22. Rick Dove, Bill Schindel, "Agile Modelling and Modelling Agile Systems" session at INCOSE IW2015 MBSE Workshop, Torrance, CA, January 24, 2015.

Biography

David Cook is the Systems Chief Engineer for the Moog Aircraft Group. He has over 17 years of experience in the development of safety critical flight control actuation systems. He has been an active advocate and change agent in the introduction MBSE his organization. David is currently a member of the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative.

William D. (Bill) Schindel is president of ICTT System Sciences. His engineering career began in mil/aero systems with IBM Federal Systems, included faculty service at Rose-Hulman Institute of Technology, and founding of three systems enterprises. Bill co-led a 2013 project on the science of Systems of Innovation in the INCOSE System Science Working Group. He co-leads the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative.