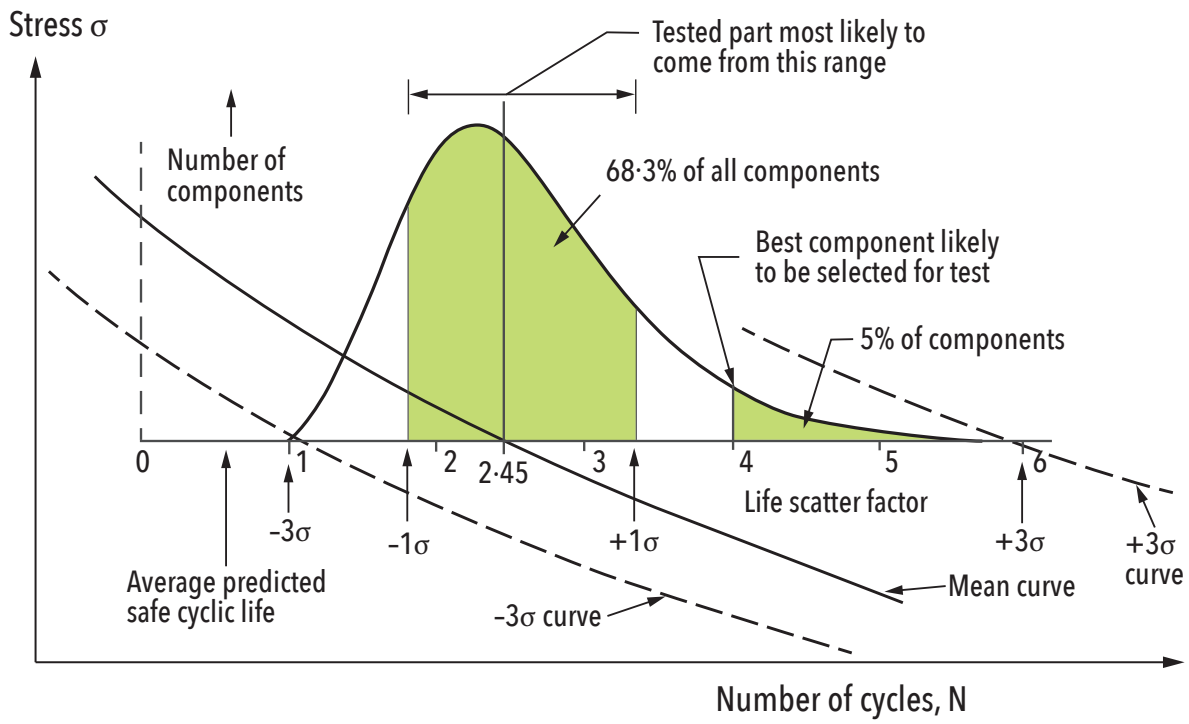
INSIGHT

This Issue's Feature:
Model-Based Test and Evaluation



"I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind."

—William Thomson, Lord Kelvin

Illustration credit: from the article
You Don't Save Money by Doing Less Testing – You Save Money by Doing More of the Right Testing!
by Andrew C Pickard, Richard Beasley, and Andy J Nolan (page 70)

MARCH 2023
VOLUME 26 / ISSUE 1

A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

Systems Engineering: The Journal of The International Council on Systems Engineering

Call for Papers

The *Systems Engineering* journal is intended to be a primary source of multidisciplinary information for the systems engineering and management of products and services, and processes of all types. Systems engineering activities involve the technologies and system management approaches needed for

- definition of systems, including identification of user requirements and technological specifications;
- development of systems, including conceptual architectures, tradeoff of design concepts, configuration management during system development, integration of new systems with legacy systems, integrated product and process development; and
- deployment of systems, including operational test and evaluation, maintenance over an extended life-cycle, and re-engineering.

Systems Engineering is the archival journal of, and exists to serve the following objectives of, the International Council on Systems Engineering (INCOSE):

- To provide a focal point for dissemination of systems engineering knowledge
- To promote collaboration in systems engineering education and research
- To encourage and assure establishment of professional standards for integrity in the practice of systems engineering
- To improve the professional status of all those engaged in the practice of systems engineering
- To encourage governmental and industrial support for research and educational programs that will improve the systems engineering process and its practice

The journal supports these goals by providing a continuing, respected publication of peer-reviewed results from research and development in the area of systems engineering. Systems engineering is defined broadly in this context as an interdisciplinary approach and means to enable the realization of successful systems that are of high quality, cost-effective, and trustworthy in meeting customer requirements.

The *Systems Engineering* journal is dedicated to all aspects of the engineering of systems: technical, management, economic, and social. It focuses on the life-cycle processes needed to create trustworthy and high-quality systems. It will also emphasize the systems management efforts needed to define, develop, and deploy trustworthy and high quality processes for the production of systems. Within this, *Systems Engineering* is especially concerned with evaluation of the efficiency and effectiveness of systems management, technical direction, and integration of systems. *Systems Engineering* is also very concerned with the engineering of systems that support sustainable development. Modern systems, including both products and services, are often very knowledge-intensive, and are found in both the public and private sectors. The journal emphasizes strategic and program management of these, and the information and knowledge base for knowledge principles, knowledge practices, and knowledge perspectives for the engineering of systems. Definitive case studies involving systems engineering practice are especially welcome.

The journal is a primary source of information for the systems engineering of products and services that are generally large in scale, scope, and complexity. *Systems Engineering* will be especially concerned with process- or product-line–related efforts needed to produce products that are trustworthy and of high quality, and that are cost effective in meeting user needs. A major component of this is system cost and operational effectiveness determination, and the development of processes that ensure that products are cost effective. This requires the integration of a number of engineering disciplines necessary for the definition, development, and deployment of complex systems. It also requires attention to the lifecycle process used to produce systems, and the integration of systems, including legacy systems, at various architectural levels. In addition, appropriate systems management of information and knowledge across technologies, organizations, and environments is also needed to insure a sustainable world.

The journal will accept and review submissions in English from any author, in any global locality, whether or not the author is an INCOSE member. A body of international peers will review all submissions, and the reviewers will suggest potential revisions to the author, with the intent to achieve published papers that

- relate to the field of systems engineering;
- represent new, previously unpublished work;
- advance the state of knowledge of the field; and
- conform to a high standard of scholarly presentation.

Editorial selection of works for publication will be made based on content, without regard to the stature of the authors. Selections will include a wide variety of international works, recognizing and supporting the essential breadth and universality of the field. Final selection of papers for publication, and the form of publication, shall rest with the editor.

Submission of quality papers for review is strongly encouraged. The review process is estimated to take three months, occasionally longer for hard-copy manuscript.

Systems Engineering operates an online submission and peer review system that allows authors to submit articles online and track their progress, throughout the peer-review process, via a web interface. All papers submitted to *Systems Engineering*, including revisions or resubmissions of prior manuscripts, must be made through the online system. Contributions sent through regular mail on paper or emails with attachments will not be reviewed or acknowledged.

All manuscripts must be submitted online to *Systems Engineering* at ScholarOne Manuscripts, located at:

https://mc.manuscriptcentral.com/SYS

Full instructions and support are available on the site, and a user ID and password can be obtained on the first visit.

INSIGHT

A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

MARCH 2023 VOLUME 26 / ISSUE 1

Inside this issue

About This Publication

INFORMATION ABOUT INCOSE

INCOSE's membership extends to over 20,000 individual members and more than 200 corporations, government entities, and academic institutions. Its mission is to share, promote, and advance the best of systems engineering from across the globe for the benefit of humanity and the planet. INCOSE charters chapters worldwide, includes a corporate advisory board, and is led by elected officers and directors.

For more information, click here:
The International Council on Systems Engineering
(www.incose.org)
INSIGHT is the magazine of the International Council on Systems Engineering. It is published four times per year and

OVERVIEW

features informative articles dedicated to advancing the state of practice in systems engineering and to close the gap with the state of the art. ***INSIGHT*** delivers practical information on current hot topics, implementations, and best practices, written in applications-driven style. There is an emphasis on practical applications, tutorials, guides, and case studies that result in successful outcomes. Explicitly identified opinion pieces, book reviews, and technology roadmapping complement articles to stimulate advancing the state of practice. ***INSIGHT*** is dedicated to advancing the INCOSE objectives of impactful products and accelerating the transformation of systems engineering to a model-based discipline.
Topics to be covered include resilient systems, model-based systems engineering, commercial-driven transformational systems engineering, natural systems, agile security, systems of systems, and cyber-physical systems across disciplines and domains of interest to the constituent groups in the systems engineering community: industry, government, and academia. Advances in practice often come from lateral connections of information dissemination across disciplines and domains. ***INSIGHT*** will track advances in the state of the art with follow-up, practically written articles to more rapidly disseminate knowledge to stimulate practice throughout the community.

Editor-In-Chief William Miller
insight@incose.net +1 908-759-7110

Assistant Editor Lisa Hoverman
lisa.hoverman@incose.net

Theme Editors

Layout and Design Chuck Eng
chuck.eng@comcast.net

Member Services INCOSE Administrative Office
info@incose.net +1 858 541-1725

Officers
President: Marilee Wheaton, *INCOSE Fellow, The Aerospace Corporation*
President-Elect: Ralf Hartmann, *INCOSE Fellow, proSys*
Secretary: Don York, ESEP, *SAIC*
Treasurer: Michael Vinarcik, *ESEP, SAIC*

Directors
Director for Academic Matters: Alejandro Salado, *University of Arizona*
Direector for Marketing and Communications: Honor Lind, *Hart Initiative, Inc.*
Director for Outreach: Kirk Michealson, *Tackle Solutions, LLC*
Director for Americas Sector: Renee Steinwand, *ESEP, Booz Allen Hamilton*
Director for EMEA Sector: Sven-Olaf Schulze, *CSEP, Huennemeyer Consulting GmbH*
Director for Asia-Oceania Sector: Serge Landry, *ESEP, Equilibrant Force*
Chief Information Officer (CIO)**:** Barclay Brown, *ESEP, Raytheon*
Technical Director: Olivier Dessoude, *Naval Group*
Deputy Technical Director: Erika Palmer, *Cornell University*
Services Director: Richard Beasley, *ESEP, Rolls-Royce plc, retired*
Deputy Services Director: Heidi Davidz, *CSEP, ManTech International Corporation*
Director for Strategic Integration: David Long, *INCOSE Fellow, ESEP, Blue Holon*
Corporate Advisory Board Chair: Ronald Giachetti, *Naval Postgraduate School*
Corporate Advisory Board Co-Chair: Michael Dahhlberg, *ESEP, KBR*
CAB Co-chair: Ron Giachetti, *Naval Postgraduate School*
Chief of Staff: Andy Pickard, *Rolls Royce Corporation, retired*

Executive Director: Steve Records, *INCOSE*

PERMISSIONS

*** PLEASE NOTE: If the links highlighted here do not take you to those web sites, please copy and paste address in your browser.**

Permission to reproduce Wiley journal Content:
Requests to reproduce material from John Wiley & Sons publications are being handled through the RightsLink® automated permissions service.

Simply follow the steps below to obtain permission via the Rightslink® system:

- Locate the article you wish to reproduce on Wiley Online Library (http://onlinelibrary.wiley.com)
- Click on the 'Request Permissions' link, under the ‹ARTICLE TOOLS› menu on the abstract page (also available from Table of Contents or Search Results)
- Follow the online instructions and select your requirements from the drop down options and click on 'quick price' to get a quote
- Create a RightsLink® account to complete your transaction (and pay, where applicable)
- Read and accept our Terms and Conditions and download your license
- For any technical queries please contact customercare@copyright.com
- For further information and to view a Rightslink® demo please visit www.wiley.com and select Rights and Permissions.

AUTHORS – If you wish to reuse your own article (or an amended version of it) in a new publication of which you are the author, editor or co-editor, prior permission is not required (with the usual acknowledgements). However, a formal grant of license can be downloaded free of charge from RightsLink if required.

Photocopying
Teaching institutions with a current paid subscription to the journal may make multiple copies for teaching purposes without charge, provided such copies are not resold or copied. In all other cases, permission should be obtained from a reproduction rights organisation (see below) or directly from RightsLink®.

Copyright Licensing Agency (CLA)
Institutions based in the UK with a valid photocopying and/or digital license with the Copyright Licensing Agency may copy excerpts from Wiley books and journals under the terms of their license. For further information go to CLA.

Copyright Clearance Center (CCC)
Institutions based in the US with a valid photocopying and/or digital license with the Copyright Clearance Center may copy excerpts from Wiley books and journals under the terms of their license, please go to CCC.

Other Territories: Please contact your local reproduction rights organisation. For further information please visit www.wiley.com and select Rights and Permissions.
If you have any questions about the permitted uses of a specific article, please contact us.

Permissions Department – UK
John Wiley & Sons Ltd.
The Atrium,
Southern Gate,
Chichester
West Sussex, PO19 8SQ
UK
Email: Permissions@wiley.com
Fax: 44 (0) 1243 770620
or

Permissions Department – US
John Wiley & Sons Inc.
111 River Street MS 4-02
Hoboken, NJ 07030-5774
USA
Email: Permissions@wiley.com
Fax: (201) 748-6008

ARTICLE SUBMISSION

insight@incose.net

Publication Schedule. ***INSIGHT*** is published four times per year. Issue and article submission deadlines are as follows:

- March 2023 issue – 2 January 2023
- June 2023 issue – 1 April 20023
- September 2023 issue – 1 July 2023
- January 2023 issue – 1 October 2023

For further information on submissions and issue themes, visit the INCOSE website: www.incose.org

© 2023 Copyright Notice.
Unless otherwise noted, the entire contents are copyrighted by INCOSE and may not be reproduced in whole or in part without written permission by INCOSE. Permission is given for use of up to three paragraphs as long as full credit is provided. The opinions expressed in ***INSIGHT*** are those of the authors and advertisers and do not necessarily reflect the positions of the editorial staff or the International Council on Systems Engineering.
ISSN 2156-485X; (print) ISSN 2156-4868 (online)

ADVERTISE

Readership
INSIGHT reaches over 20,000 individual members and uncounted employees and students of more than 100 CAB organizations worldwide. Readership includes engineers, manufacturers/purchasers, scientists, research and development professionals, presidents and chief executive officers, students, and other professionals in systems engineering.

Issuance	Circulation
2022, Vol 25, 4 Issues	100% Paid

Contact us for Advertising and Corporate Sales Services
We have a complete range of advertising and publishing solutions professionally managed within our global team. From traditional print-based solutions to cutting-edge online technology the Wiley-Blackwell corporate sales service is your connection to minds that matter. For an overview of all our services please browse our site which is located under the Resources section. Contact our corporate sales team today to discuss the range of services available:

- Print advertising for non-US journals
- Email Table of Contents Sponsorship
- Reprints
- Supplement and sponsorship opportunities
- Books
- Custom Projects
- Online advertising

Click on the option below to email your enquiry to your nearest office:

- Asia and Australia corporatesalesaustralia@wiley.com
- Europe, Middle East and Africa (EMEA) corporatesaleseurope@wiley.com
- Japan corporatesalesjapan@wiley.com
- Korea corporatesaleskorea@wiley.com

USA (also Canada, and South/Central America):

- Healthcare Advertising corporatesalesusa@wiley.com
- Science Advertising Ads_sciences@wiley.com
- Reprints Commercialreprints@wiley.com
- Supplements, Sponsorship, Books and Custom Projects busdev@wiley.com

Or please contact:
Marcom@incose.net

CONTACT

Questions or comments concerning:

Submissions, Editorial Policy, or Publication Management
Please contact: William Miller, Editor-in-Chief
insight@incose.net

Advertising—*please contact:*
Marcom@incose.net

Member Services – *please contact:* info@incose.org

ADVERTISER INDEX

March Volume 26-1

CORPORATE ADVISORY BOARD — MEMBER COMPANIES

Aerospace Corporation, The
Airbus
AM General LLC
Analog Devices, Inc.
ARAS Corp
Arcfield
Australian National University
AVIAGE SYSTEMS
Aviation Industry Corporation of China
BAE Systems
Ball Aerospace
Bechtel
Becton Dickinson
Belcan Engineering Group LLC
Boeing Company, The
Bombardier Transportation
Booz Allen Hamilton Inc.
C.S. Draper Laboratory, Inc.
CACI, Inc - Federal
California State University Dominguez Hills
Carnegie Mellon University Software Engineering Institute
Change Vision, Inc.
Colorado State University Systems Engineering Programs
Cornell University
Cranfield University
Cubic
Cummins Inc.
Cybernet MBSE Co, Ltd
Dassault Systèmes
Defense Acquisition University
Deloitte Consulting, LLC
Denso Create Inc
Drexel University
Eindhoven University of Technology
EMBRAER
Federal Aviation Administration (U.S.)
Ford Motor Company
Fundacao Ezute
General Dynamics
General Electric Aviation
General Motors
George Mason University
Georgia Institute of Technology
IBM
Idaho National Laboratory
ISAE - Supaero
ISDEFE
ITID, Ltd
IVECO SPA
Jacobs
Jama Software
Jet Propulsion Laboratory
John Deere
Johns Hopkins University
KBR
KEIO University
L3Harris Technologies
Lawrence Livermore National Laboratory
Leidos
Lockheed Martin Corporation
Los Alamos National Laboratory
Loyola Marymount University
Mahindra University
ManTech International Corporation
Maplesoft
Marquette University
Massachusetts Institute of Technology
MBDA (UK) Ltd
MetaTech Consulting Inc.
Missouri University of Science & Technology
MITRE Corporation, The
Mitsubishi Heavy Industries, Ltd
Modern Technology Solutions, Inc.
National Aeronautics and Space Administration (NASA)
National Reconnaissance Office (NRO)
National Security Agency Enterprise Systems
Naval Postgraduate School
Nissan Motor Co, Ltd
Northrop Grumman Corporation
Pacific Northwest National Laboratory
Pennsylvania State University
Peraton
Petronas Nasional Berhad
Prime Solutions Group, Inc
Project Performance International (PPI)
Purdue University
QRA Corp
Raytheon Corporation
Rolls-Royce
Saab AB
SAIC
Sandia National Laboratories
Saudi Railway Company
Siemens
Sierra Nevada Corporation
Singapore Institute of Technology
SPEC Innovations
Stevens Institute of Technology
Strategic Technical Services LLC
Swedish Defence Materiel Administration (FMV)
Systems Planning and Analysis
Tata Consultancy Services
Thales
The University of Arizona
Torch Technologies
TOSHIBA Corporation
Trane Technologies
Tsinghua University
UC San Diego
UK MoD
University of Alabama in Huntsville
University of Arkansas
University of Connecticut
University of Maryland
University of Maryland, Baltimore County
University of Michigan, Ann Arbor
University of New South Wales, The, Canberra
University of Southern California
University of Texas at El Paso (UTEP)
US Department of Defense
Veoneer
VG2PLAY
Virginia Tech
Vitech
Volvo Cars Corporation
Volvo Construction Equipment
Wabtec Corporation
Woodward Inc
Worcester Polytechnic Institute- WPI
Zuken Inc

FROM THE EDITOR-IN-CHIEF

William Miller, insight@incose.net

We are pleased to announce the March 2023 *INSIGHT* issue published cooperatively with John Wiley & Sons as the systems engineering practitioners' magazine. The *INSIGHT* mission is to provide informative articles on advancing the practice of systems engineering and to close the gap between practice and the state of the art as advanced by *Systems Engineering*, the Journal of INCOSE also published by Wiley.

The issue theme is *model-based test and evaluation* and is a follow-up to the March 2017 *INSIGHT* that was published in collaboration with the March 2017 issue of the *International Test and Evaluation Association (ITEA) Journal* on the common theme of the engagement of systems engineering with test and evaluation.

The recent December 2022 *INSIGHT* on the Archimedes initiative has several articles on model-based test and evaluation and verification and validation. In particular, the overview article "TNO-ESI – Systems Engineering Methodologies for Managing Complexity in the High-Tech Equipment Industry: Our Roadmap" by Wouter Leibbrandt, Jacco Wesselius, and Frans Beenker references the TorXakis modeling language and tool for model-based testing (https://torxakis.org/ and https://torxakis.org/userdocs/stable/). The reference cites "Model-Based Testing with TorXakis" by Jan Tretmans and Piërre van de Laar, both with TNO-ESI, presented at the 2019 Central European Conference on Information and Intelligent Systems, Varaždin, Croatia (http://archive.ceciis.foi.hr/app/public/conferences/2019/Proceedings/QSS/QSS3.pdf).

This issue of *INSIGHT* features relevant articles selected from past INCOSE symposia papers by authors representing all three INCOSE sectors: Americas; Europe, Middle East, and Africa (EMEA); and Asia-Oceania. Our intent is to encourage and stimulate our systems engineering community to focus more energy on test and evaluation. We thank the authors and their sponsoring organizations for their contributions. Articles referencing research and commercial systems engineering tools and products does not represent *INSIGHT* and INCOSE endorsement of referenced tools and products.

The March 2023 *INSIGHT* leads off with "The Challenge of Enabling Dynamic Innovation with Rigor" by John Frederick, Columb Higgins, and Angela Moore. This article examines lessons learned from recent verification and validation (V&V) summits and technical interchange meetings (TIMs) held by the US Federal Aviation Administration (FAA) V&V Strategies and Practices Branch, exploring the challenges of being agile and dynamic (in step with the pace of technology) while being effectively systematic and rigorous.

"Determining Reliability Requirements and Testing Costs in the Early Stages of Single Use Medical Product Design" by Fritz Eubanks examines methods for determining reliability requirements, the cost of reliability testing for single use medical devices in the design input phase of product development, and how the costs of testing and potential errors can be used to perform trade-off analysis between reliability tolerance and confidence level.

"A Concept for Set-based Design of Verification Strategies" by Pen Xu and Alejandro Salado presents an approach to apply set-based design to the design of verification activities to enable the execution of dynamic contracts for verification strategies, ultimately resulting in more valuable verification strategies than current practice.

"Formalizing the Representativeness of Verification Models using Morphisms" by Paul Wach, Peter Beling, and Alejandro Salado explores the use of system theoretic morphisms to mathematically characterize the validity of representativeness between verification models and corresponding system design.

"Verification and Validation of SysML Models" by Myron Hecht and Jaron Chen describes a methodology for performing verification and validation on models written in SysML. Both manual and automated methods are used to verify and validate these requirements. Manual methods are necessary where knowledge of the domain and other extrinsic characteristics are necessary. Automated methods can be used where the requirements cover the use of SysML. Examples from a public domain SysML model of a satellite are presented to demonstrate application of automated requirements verification.

"From Model-based to Model and Simulation-based Systems Architectures – Achieving Quality Engineering through Descriptive and Analytical Models" by Pierre Nowodzienski and Juan Navas leverages model-based systems engineering (MBSE) approaches and complement them with simulation techniques to improve the quality of system architecture definition to come up with innovative solutions.

"System Verification and Validation Approach Using the MagicGrid Framework" by Aurelijus Morkevicius, Aiste Aleksandraviciene, and Zilvinas Strolia proposes an approach to perform verification and validation of a system using system models developed with the Systems Modeling Language (SysML) and in accordance with the MagicGrid (formerly known as MBSE Grid) framework. The approach covers system testing activities beginning with verification of the lowest modeled system elements against system requirements and finishing with validation of the system as a whole, against stakeholder needs.

"Configuration Management for Model Based Systems Engineering—An Example from the Aerospace Industry" by Adriana D'Souza and Phanikrishna Thota explores the use of configuration management for modeling and simulation in an aerospace setting, with a specific example involving landing gear and its surrounding systems.

"You Don't Save Money by Doing Less Testing—You Save Money by Doing More of the Right Testing!" by Andrew Pickard, Richard Beasley, and Andy Nolan examine the prediction of the fatigue lives of critical parts in gas turbine engines, to illustrate the more general case of performing tests to calibrate models that then have general applicability across multiple projects, rather than focusing testing on the needs of a specific project. In some circumstances, testing may not even be the best approach to take; if some level of error escape into service is acceptable (unlike the life prediction example given in this paper) then more focus on requirements validation and design review may provide a more cost-effective approach. This is where the linkage in a systems engineering model between requirements, functions, failure modes and effects analysis, verification test cases, and available calibrated models can help with identifying opportunities and risks.

"Inconsistent and Incomplete Datasheet: The Case for Systematic Use of Requirement Engineering" by Lorraine Brisacier-Porchon and Omar Hammami explores the public user datasheet relevance compared to the system engineering requirements that are the artifacts of system design architecture. The use of connecting components off-the-shelf (COTS) user manual to system requirements is discussed, even more if the systems are to be re-used in a system production line. For example, the rising complexity of unmanned ground vehicle (UGV) systems imposes engineering steps that would ensure both capabilities of the system and resilience to its future inclusion in a system-of system context. During its operational usage, the UGV is supposed to be maneuvered for specifically designed purposes following user manual datasheet of the COTS that are integrated.

"Exploring the Test and Evaluation Space using Model Based Conceptual Design (MBCD) Techniques" by David Flanigan and Kevin Robinson offers an approach to extend the MBCD methodology addressing the system concept in the early stages in the lifecycle as published in the December 2014 *INSIGHT* (Volume 17 Issue 4) to better consider the T&E space.

"Framework for Formal Verification of Machine Learning Based Complex System-of-Systems" by Ramakrishnan Raman, Nikhil Gupta, and Yogananda Jeppu describes a novel approach applying machine learning based classifiers and formal methods for analyzing and evaluating emergent behavior of complex system-of-systems that comprise a hybrid of constituent systems governed by conventional models and machine learning models. The approach develops a machine learning classifier model that learns on potential negative and positive emergent behaviors, and predicts the behavior exhibited. A formal verification model is developed to assert negative emergent behavior. The approach is illustrated through the case of a swarm of autonomous UAVs flying in a formation, and dynamically changing the shape of the formation, to support varying mission scenarios. The effectiveness and performance of the approach are quantified.

We hope you find *INSIGHT*, the practitioners' magazine for systems engineers, informative and relevant. Feedback from readers is critical to *INSIGHT*'s quality. We encourage letters to the editor at insight@incose.net. Please include "letter to the editor" in the subject line. *INSIGHT* also continues to solicit special features, standalone articles, book reviews, and op-eds. For information about *INSIGHT*, including upcoming issues, see https://www.incose.org/products-and-publications/periodicals#INSIGHT. For information about sponsoring *INSIGHT*, please contact the INCOSE marketing and communications director at marcom@incose.net. ■

MARCH 2023
VOLUME 26/ ISSUE 1

The Challenge of Enabling Dynamic Innovation with Rigor

John Frederick, *Manager, FAA V&V Strategies and Practices Branch*, john.frederick@faa.gov; **Columb Higgins,** columb.g-ctr.higgins@faa.gov; and **Angela Moore,** angela.ctr.moore@faa.gov
Copyright ©2023 by John Frederick, Columb Higgins, and Angela Moore. Published by INCOSE with permission

ABSTRACT
How do we incubate and accelerate innovation? This article examines lessons learned from recent Verification and Validation (V&V) summits and Technical Interchange Meetings (TIMs) held by the Federal Aviation Administration (FAA) V&V Strategies and Practices Branch, which explored the challenges of being agile and dynamic (in step with the pace of technology) while being effectively systematic and rigorous.

KEYWORDS: innovation, knowledge convergence, agile, test and evaluation, verification and validation

Innovation is critical in this transformational period where technological advancement and subsequent obsolescence are moving at a rapid pace. Intel® claims that by 2030, there will be circuits with transistor counts of a trillion, roughly 10 times the number of transistors currently available on modern CPUs. If this is any indication of the pace of change, planners, designers, and developers cannot say with confidence what conditions will be or what users may need years from now. With this extreme pace of technology, the challenge for innovators in this dynamic environment is how to explore new opportunities and remain open to new concepts. All stakeholders have to be open to innovate, not just analyze, and that requires accepting uncertainty, embracing knowledge convergence, allowing space for various stakeholders to contribute, and not fearing change.

How do we create the space—physically and mentally—for innovation to take place? This article includes lessons learned concerning accelerating innovation through knowledge convergence and agile principles presented during the 17th Annual Verification and Validation (V&V) Summit. The summit is hosted by the V&V Strategies and Practices Branch of the Federal Aviation Administration's (FAA's) William J. Hughes Technical Center (WJHTC) in Atlantic City, New Jersey, US. V&V Strategies and Practices Branch manager John Frederick started the summit to provide a collaborative environment for convergence: the assembling of different ideas from different groups to contribute knowledge and context to a complex idea so that commonalities and similarities become visible, and synthesis of understanding may occur. This year's summit, held September 21–22 in a hybrid attendance format, featured presenters and attendees at the National Aerospace Research and Technology Park (NARTP) adjacent to the WJHTC and others participating remotely via Zoom. Speakers and presentations delved into the theme: "Enabling Dynamic Innovation with Rigor."

Figure 1. The 2022 V&V Summit theme—'Enabling Dynamic Innovation with Rigor'

In addition to the V&V Summit, a companion Technical Interchange Meeting (TIM) was conducted (in a smaller breakout group) with a specific focus to address the challenges and complexities inherent within agile principles and practices. There

were four presentations and a roundtable discussion in which moderators led brief discussions seeking discovery and clarification on the topic.

Approximately 260 people attended the summit and TIM, with 17 speakers from the FAA, United States Space Force (USSF), United States Air Force (USAF), Department of Defense (DOD), National Aeronautics and Space Administration (NASA), Carnegie Mellon University (CMU) Software Engineering Institute (SEI), Stevens Institute of Technology, Verizon, Science Applications International Corporation (SAIC), and the Volpe National Transportation Systems Center. The post-summit reporting captures ideas and values from these speakers to help enterprises innovate and adapt.

CONVERGENCE: A SYNTHESIS OF IDEAS, CONCEPTS, AND PERSPECTIVES

Summits, conferences, and conventions can be powerful fora for knowledge convergence. Convergence is a natural synthesis that occurs when individual contributions, like strands of a web, are brought in context to converge and intertwine knowledge into a comprehensive innovative solution. Imagine solving a complex problem where each contributor has a segment of the solution. Separately, each solution element is only understood by its originator. By creating spaces where convergence can occur, one or many big pictures and their possibilities take shape. Placement, orientation, and the value of each element may inspire and spawn new associations to other segments of the solution and concepts previously overlooked, disassociated, or misunderstood.

Innovation cannot occur until the convergence among different kinds of knowledge are synthesized to spawn innovative solutions. For instance, the first operational computer did not appear until 1946, even though all the necessary knowledge was available in 1918 (Drucker 2002). Sometimes a spark or catalyst (some may call it luck) is needed to create the innovative combustion. To accelerate the synthesis of ideas, we must be intentional about seeking and fostering knowledge gathering and exchange, challenging old concepts, and improving perceptions about potentials. The V&V Summit aims to establish a network and community of V&V and Test and Evaluation (T&E) experts to exchange ideas, explore new concepts for future needs, promote continuous learning, and foster innovative collaboration.

Recent summit themes focused on nurturing innovation, creativity, and collaboration. The 2021 summit theme, "The Fusion of Art and Science," addressed how V&V professionals will have to be more than scientists and engineers as they develop innovative and creative solutions for the future of aviation. The summit sought to foster new perspectives, increase awareness, and inspire notions of curiosity and discovery for projects and organizations. The immediate past summit theme was "Enabling Dynamic Innovation with Rigor," which examined the challenges of innovating with agility at the pace of technology while ensuring that mission-critical systems and services are safe, effective, and secure.

Innovation requires mental space as well; it must be both conceptual and perceptual, requiring not only cognition but imagination. Speakers at the 2021 V&V Summit highlighted the value of integrated arts and sciences to the advancement of innovation, focusing on new concepts and improved perceptions; learning through interacting, observing, and listening; challenging the status quo; and creating environments where good scientists (who are cognizant of their ignorance) can gain knowledge. Innovation requires a fine balancing act between rapid change and known stability. People resist change, doubt the new, and abhor complexity. They will often settle for a known deficiency rather than accept products that require a mental stretch. But while simplicity and certainty foster comfort and confidence, they can also lead to stagnation. Innovation requires being open to uncertainty and the unexpected.

The poet John Keats coined the term "Negative Capability," by which he meant the ability of someone to hold opposing ideas in their mind simultaneously — "of being in uncertainties, mysteries, doubts, without any irritable reaching after fact and reason." Negative capability is a powerful tool in literature because it leaves a key element of the story unexplained, a tension or strategic opacity that invites the reader to fill in a backstory using their own imagination, or reconcile seemingly contradictory motivations and rationales. This technique releases "an enormous energy that had been at least partially blocked or contained by familiar, reassuring explanations" (Greenblatt 2004). In science, analysts seek to explain. But in innovating they need to be open to uncertainty. Uncertainty is uncomfortable, but it is often where the "magic" occurs—at the intersection of art and science; the what if and what is.

Mental and Physical Modeling for Convergence and Innovation

Digital twins, simulators, sandboxes, and virtual reality can also be powerful tools for convergence. A digital twin is a virtual representation of an object or system with real-time, bidirectional data flow enabled by Artificial Intelligence (AI) and/or Machine Learning (ML). It can model cities or whole ecosystems. Users have the ability to simulate decisions and outcomes, allowing them to work together to create scenarios and model systems, integrate data from many different sources, and experiment without risk. Digital twin models and simulators provide environments where it is encouraged (and safe) to innovate through creation, building, play, trial, prototyping, and even destruction. Having facilities, laboratories, skunkworks, and environments where ideas can be grown is critical to learning, training, and eventually implementing those new ideas. A digital twin can accelerate contextual understanding, buy down risks for operational changes, and enable proactive and prescriptive adaptation instead of reactive management.

A digital twin can also serve as a virtual space for convergence between human users and AI. As AI develops, using explanatory models and reinforcement learning, relationships between human users and AI will be increasingly based on performance and trust. Effective digital twins and simulators must also engage the user in an immersive experience. Numbers, statistical analysis, etc., are only a small part of the story told by digital twins and simulators.

Figure 2. Innovation requires convergence among different kinds of knowledge

What matters most is the story told about the real-life applications of these numbers.

Creating a space for knowledge convergence—physically or mentally—means accepting that there may not be one right answer. Indeed, there may be no immediate answer at all. Instead, convergence welcomes many different perspectives with the aim of stimulating responsible discussion. Leadership plays an important role in creating a space for knowledge convergence. At the WJHTC, FAA leadership has fostered internship programs; employee engagement teams and mentoring; Aviation Science, Technology, Engineering, and Mathematics (AvSTEM) outreach efforts; and the Innovation and Technology Advisory Council (ITAC) to allow for the constant cross-pollination of ideas. Cooperative research agreements between the FAA, Department of Defense (DOD), and Department of Homeland Security (DHS), and events like the Cyber Rodeo and Tech Center Showcase, are also valuable avenues for knowledge convergence.

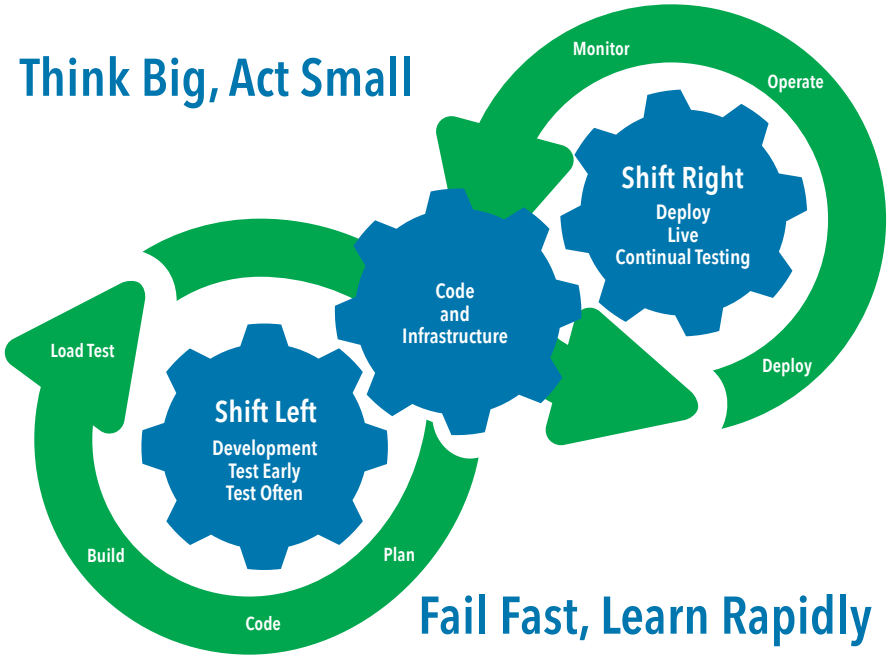


Figure 3. The agile process (Blomberg 2018)

AGILE: CONTINUAL MANAGEMENT OF THE TRAJECTORY TO STAY ON TARGET

Successful agile principles and practices will be critical to innovating at the pace of technology and obsolescence. Participants at the TIM on agile principles and practices did not arrive at one single definition for "Agile," but they did describe consistent characteristics by offering their thoughts, lessons learned, and challenges. Discussion focused on three major areas: 1) defining agile, 2) culture change, and 3) developing the right metrics.

Defining Agile

Defining agile can be difficult, as different people have different ideas of what it means. However, there were some consistent characteristics of agile principles and practices raised throughout the TIM. During a breakout session, TIM participants used interactive polling software to define what agile meant to them. The following terms were most commonly identified:

- Iterative
- Adaptive
- Discipline
- Flexible
- Collaborative
- Communication
- Multidisciplinary.

There are many myths and notions about what agile is, how it is applied, whether a framework is needed, and, if so, which one. One myth is that agile is always faster. Using an iterative process, teams can hone in on customer requirements more efficiently but this may not necessarily shorten the overall project timeline. Another myth is that agile reduces documentation. Actually, scope, content, and frequency of documentation may be different from conventional large documentation efforts at the end of a milestone.

Agile is a set of values and principles. Agile practitioners value: 1) individuals and interactions over processes and tools, 2) working software over comprehensive documentation, 3) customer collaboration over contract negotiation, and 4) responding to change over following a plan. Agile is about thinking big and acting small, failing fast and learning rapidly.

Agile is iterative. It is a responsive and constant process; more than just rescheduling testing earlier in time. Through iterations, agile teams narrow the cone of uncertainty from the starting point of maximum ignorance, to a Minimum Viable Product (MVP), and then completion. They learn along the way and are smarter when they are ready to deploy.

Agile embraces the idea that there is a lot of uncertainty about the solution at the beginning of a project, so that is not the time to lock in scope and requirements. Instead, as knowledge converges and concepts synthesize, refined requirements are developed, thereby increasing the probability of success to build the intended product. Agile practitioners incrementally assure viability, which requires knowledge of the system, how it will be used, and the domain in which it will be used. We learn "what we don't want" is just as valuable as "what we think we want." This allows development efforts to adjust trajectory early and fosters an adaptive approach. Agile narrows the cone of uncertainty and facilitates staying on target and on time. Ultimately, successful agile acquisition delivers what is really needed instead of what was thought to be needed.

A video by actor and comedian John Cleese on "The Importance of Mistakes," shown during the 2022 V&V Summit, helps explain parts of an agile process. In the video, Cleese presents Gordon the Guided Missile. There can be two different approaches to sending Gordon to hit its target: either plan everything out and set its target before firing, or ask Gordon along the way how it is doing. The launch has a fixed direction and a setting. In a traditional waterfall program, we are expected to have perfect aim. However, by asking along the way and correcting course, it may appear we are "making mistakes," but we are actually learning by doing while adjusting so we hit the bullseye.

Agile requires management of uncertainty. Agile practitioners must assess outcomes that may not necessarily be part of the plan. They need permission from leadership and stakeholders to change their minds later on. One key to agile development is asking illuminating questions early, which allows a team to build viability into the system. Agile practices should involve and integrate stakeholders into development. Data and test results should be available before deadlines so that examination can occur earlier when corrective actions are less expensive to make and options are more varied. V&V

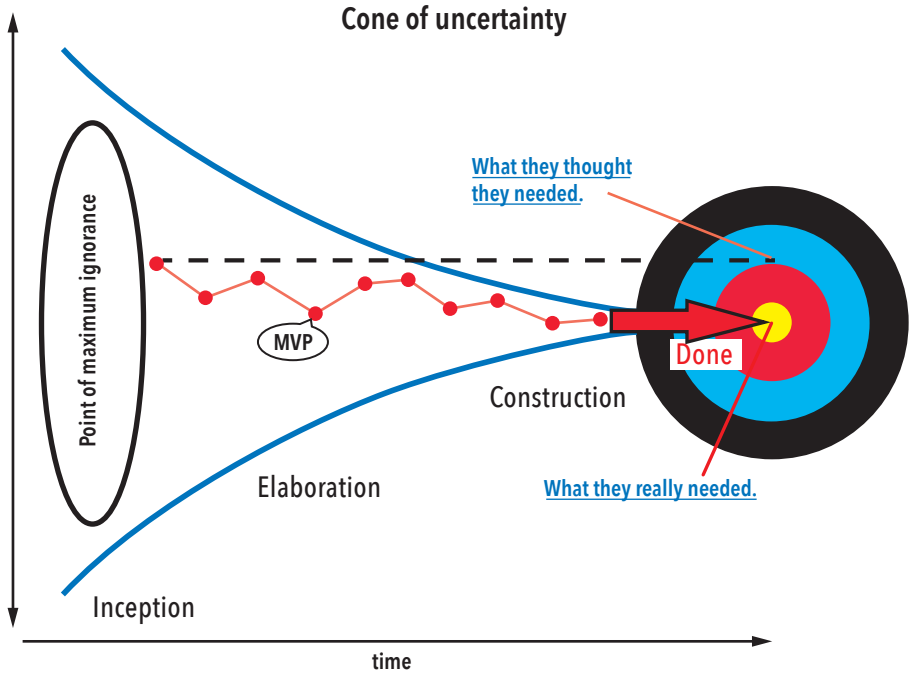


Figure 4. Narrowing the cone of uncertainty (Montemurro 2021)

should be involved in testing and validating the system as it is being developed, working with the product owner and team in writing acceptance criteria and validating the definition of "done." Stakeholder feedback has to be a continual process and not just a gate that you pass through at the end.

Agile done poorly can circumvent controls that are in place. Agile done well uses interdisciplinary teams that leverage their expertise early and often to build a system. Organizations cannot just focus on adopting some terminology and agile ceremonies without changing the way they perform the detailed technical work. Instantiating the methods as if they are templates to be filled out can lead organizations to miss the intended benefits.

Culture Change

Implementing and integrating all these values and ideas requires an organizational culture prepared to adopt agile principles and practices. Speakers at the TIM identified culture as the biggest barrier to adoption of agile practices. Traditional waterfall development methods and organizational silos are comfortable and culturally accepted, but they may prevent an environment where ideas and people are free to fail, learn, adjust, and repeat without fear from punishment or other negative outcomes. To counteract this, agile teams need to be smaller, with natural lines of communication; self-empowered; and built for change.

Software engineer Scott Ambler offers the following definition of agile in regards to software development, which captures the organizational culture and mindset conducive to agile:

> Agile (adj.): An iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with "just enough" ceremony that produces high-quality software in a cost-effective and timely manner which meets the changing needs of its stakeholders (Ambler 2013).

Agile organizational cultures welcome change requirements, deliver working products frequently, and consist of teams with high motivation and empowerment to get work done. By contrast, waterfall development is a serial process with multiple handoffs between different groups. Components are developed separately and integrated later with testing deferred until the end. Agile organizations produce fully tested, production-ready code at each iteration with MVPs serving as part of the assurance process. Put another way, traditional waterfall development finds bugs, while agile organizations seek to prevent bugs.

Agile teams should consist of equal stakeholders at the table. If only one entity has all the power, then the scope, requirements, metrics, and verification will be slanted in their favor. Many times, whoever has the money controls the direction of a project. But what is best for the end user? To this end, communication within and between teams is crucial. Teams and stakeholder groups need to be comfortable communicating with each other, and organizations shouldn't prevent groups from talking to each other for the sake of perceived independence.

One of the main features of a traditional waterfall approach is nothing is visible until everything is done. Enterprises tend to wait until integration tests to understand the individual contributors to the capability of the system because they don't have another way of doing it. Often, this comes about because of the nature of the organizational structure. Dependencies exist on either side of boundaries, and capabilities are similarly sequestered by this boundary. Organizations that are successful view that both sides of a dependency have to operate together in a system where they succeed or fail together. Dependency encourages collaboration. Otherwise, the whole enterprise will be let down.

Enterprises should look to communicate and be transparent with sponsoring organizations. Speakers at the TIM discussed the benefits of an institutionalized stakeholder engagement plan, which will be important moving into an agile world where stakeholders, including testers, have to work together to make incremental success stories. In a stakeholder engagement plan, teams would have a specific cadence with associated schedules for each stakeholder. Everyone has to understand when to lead or follow, depending on the lifecycle phase.

There also have to be escalation rules for disagreements, where parties understand that nothing bad will happen to those who don't get their way.

Meaningful Metrics

It is important that agile practices develop and use the right metrics to control trajectory. Who owns the metrics? What supports them? If something changes along the way, can metrics change?

First, metrics have to be meaningful to everyone relative to target or goal. Appropriate metrics should measure and be motivated by the benefit of the public or end user. Metrics cannot be viewed from a defensive posture—i.e., once a stakeholder is happy with a picture, the team can go back to the real work. Institution- or enterprise-wide metrics can be somewhat antagonistic to the agile process. Sometimes institutional metrics "silo" you into reporting a given metric; it is common that problems persist far down the chain because teams force processes to fit institutionalized metrics. At the TIM, an example was given related to military readiness reporting where every model development is smashed into metrics spoken at the con-

gressional level. Such practices can inhibit an organization's ability to transition to agile types of processes.

Successful agile organizations consume data closer to the point where work is performed rather than building a large data set entirely intended for an external audience. The agile user story considers what the system needs to do in order to achieve a desired outcome, and then specifies metrics to reach that outcome. If the need is properly decomposed at each level, measurable characteristics will emerge. In this way, we give assurance to the overall system that we've contributed the right values and right measurements for each of the components. As someone drills down, they can look and see what the inputs are for each metric.

To make sure they are meaningful, teams must also V&V their metrics. If the metrics are invalid, you can hit every mark and still fail. Like measuring a patient's vital signs, body temperature could come back within an acceptable threshold, but the patient could still have high blood pressure. Successful projects establish and validate what must be measured early and then march to those measurements. The sooner and more frequently the measures can be performed, the sooner adjustments and corrective actions can be performed. As any good project manager will attest, leading metrics allow one to "go on the offensive" rather than rely on lagging metrics that require defensive action and often generate waste.

Attendees discussed the level of organizational metrics versus developmental metrics. Astute filtering of metrics and measurement data must be tiered, and to a level of granularity appropriate for various reporting levels. For example, if airport noise (metric) is measured in decibels (measurement) and durations (measurement) over a given location (measurement), what are the acceptable ranges and tolerances for each individual measurement to prevent the metric from being out of bounds? If the public cares about airport noise, then the top levels in the agency will track that metric. However, at the gemba* where the work gets done, the separate design specifications that contribute to the individual measurements may be inadequate (when combined) to perform under the allowable noise threshold. Organizations must understand the allocation of metrics and their construction at each tier so that meaningful and relative information on each can be managed and monitored.

Agile teams must not be afraid to approach leadership when requirements or metrics need to be changed. This goes back to the importance of organizational culture and leadership. Stakeholders have to presume good faith and have a common perspective. As soon as we lose those, metrics become counterproductive.

LEADERSHIP

It is critical for innovators to act in a culture of leadership with peers who are agile, flexible, and poised for change. Leadership is key to creating spaces for knowledge convergence and developing an organizational culture primed for innovation.

There are distinct differences between a leader and a "boss." A leader communicates the vision and shapes perceptions so people are able to see the future landscape. A leader helps to motivate people to want to innovate and sources training on how to implement change and empower individuals and teams to experiment without negative consequences. In contrast, a boss simply moves pawns on the chessboard, ignoring the ideas and perceptions of the staff. A boss hinders innovation by frowning on new philosophies and methods, preventing action on concepts she or he does not comprehend, and creates a punitive culture if an idea does not bring success based on financial metrics or personal glory.

In cultures of innovation, leadership combines the people, tools, training, and environments where freedom of thought is encouraged and rewarded. A servant-leader approach is valuable, providing the psychological safety needed in an organization to nurture self-empowered teams. Leaders are careful to "move out of the way" in the best interest of future success, even if it requires them to hand off power or lose organizational headship. Institutions that understand this operating model make provisions for such behavior by providing lateral opportunities, new project assignments, financial incentives, or promotion.

Leadership needs to emphasize soft-side skills too. Effective leaders celebrate internal milestones with the whole team, making public recognitions so that team members understand the desired behavior. Training should be regular and focus on how the enterprise wants to implement action, as well as equip the user with the right-sized tool set. But just training people on tools is counterproductive. If you give people a hammer, they are going to want to hit things. Valuable training doesn't just provide information, it provides perspective. Leadership needs to ensure people understand the "why" so they are equipped to address new issues with appropriate responses. To impact culture, you want to impact behavior. Perspective feeds behavior.

CONCLUSION

Defining a single way (a silver bullet) that addresses all needs while adopting agile practices with the rigor needed for safety and efficiency critical systems and services may not be 100% possible, especially in government acquisitions. On the other hand, even partial adoption of these principles and best practices would reap benefits for innovation and the implementation of complex systems and services. A key takeaway from the V&V Summit and the TIM is that these concepts will be hard to institute, but federal agencies, industry, and academia will need to increasingly adopt these principles and practices as more complex systems of systems and indeterminate systems are developed.

Institutions and enterprises, in whole or in part, must adapt and change their organizational culture to innovate. They must embrace increased communication and stakeholder engagement; smaller, more empowered teams; and iterative product deliveries/releases. They need to be dynamic, flexible, and agile while maintaining a rigorous and disciplined approach to innovation, and they should expect disruption and the need to continually alter perspectives and methods. Healthy organizations are open to change and are adept at transforming business and technical strategies to be successful in a world with warp-speed technology. ■

*Gemba (現場) is the Japanese term for "actual place," often used for the shop floor or any place where value-creating work actually occurs.

REFERENCES

- Drucker, P. 2002. "The Discipline of Innovation." *Harvard Business Review* https://hbr.org/2002/08/the-discipline-of-innovation.
- Greenblatt, S. 2004. *Will in the World: How Shakespeare Became Shakespeare*. New York, US-NY: W. W. Norton & Company. Kindle edition.
- Blomberg, B. 2018. "Why Your Performance Testing Strategy Needs to Shift Left." *dotcom-monitor*, 27 November. https://www.dotcom-monitor.com/blog/2018/11/27/why-your-performance-testing-strategy-needs-to-shift-left/.
- Montemurro, M. 2021. "Agile, meet adaptive: How global development assistance and software engineering both thrive

on an iterative approach." catalpa, 8 September. https://catalpa.io/blog/agile-meet-adaptive-how-global-development-and-software-engineering-programs-both-thrive-on-an-iterative-approach/.
- Ambler, S. 2013. "Disciplined Agile Software Development: Definition." *The Agile Modeling Method*. http://agilemodeling.com/essays/agileSoftwareDevelopment.htm

ABOUT THE AUTHORS

John Frederick is the manager of the Verification and Validation Strategies and Practices Branch at the Federal Aviation Administration's (FAA) William J. Hughes Technical Center, where he is responsible for establishing quality verification and validation methods and standards in the FAA. He has more than 36 years of Test and Evaluation (T&E) experience with Federal Aviation Administration systems. Since starting the annual Verification and Validation (V&V) Summit in 2006, Mr. Frederick has gathered speakers and participants from across the FAA, other government organizations, industry, and academia to address innovative methods for complex problems and to promote a quality V&V culture. Mr. Frederick serves as the Test Standards Board Chairman to establish test standards in the FAA and provide quality T&E oversight for the agency. He is also the International Test and Evaluation Association (ITEA) South Jersey Chapter President and serves as the T&E representative for the FAA on the Acquisition System Advisory Group and Joint Resources Council. Mr. Frederick is a graduate of Drexel University (Philadelphia) with a Bachelor of Science in Computer Systems Management. He is also a graduate of the Federal Executives Institute with a Certificate of Mastery in Leadership for a Democratic Society.

Columb Higgins is a technical writer/editor and Document Control Administrator for the Federal Aviation Administration's (FAA) Verification and Validation Strategies and Practices Branch at the William J. Hughes Technical Center. He has worked as a documentation specialist for the FAA for the past 6 years. Prior to that he was a reporter and editor for The Cape May County Gazettes and The Press of Atlantic City, where he helped develop award-winning series on opioid addiction and affordable housing. He is a graduate of Villanova University and lives in southern New Jersey with his wife and two children.

Angela Harris-Moore is a process quality engineer with a 30-year career supporting the Federal Aviation Administration (FAA). Starting as an editor, she sought training in Digital Systems Acquisition/Development; quality standards (TQM, ISO, CMM™); and disciplines such as Systems Engineering, Cybersecurity, and Safety. She holds certifications from Villanova University (international Lean/Six Sigma Master Black Belt), USDA (Government Auditing Professional), and CMTF (Configuration Management Professional). Ms. Moore's current work includes lifecycle analysis and reengineering of processes and policies affecting the National Airspace System (NAS). She supports the V&V Strategies and Practices Branch's mission to strategically promote and implement robust V&V and Test and Evaluation practices that protect engineering integrity and agency investments. She trained artificial intelligence on genre recognition as an analyst for the Music Genome Project — the technology that enables today's music streaming. A native of southern New Jersey, she is a proud Duke University alumna who volunteers as an admissions interviewer and a youth advisor. However, she is most proud of her teen twins, Alexander and Jacqueline.

Embracing Digital Engineering? We Have the Science for That.

Leaders pursuing the technical frontier team with Caltech for transformational executive and professional education. We customize unique learning experiences for organizations and their people, working one-on-one with leadership to design and deliver practical learning programs and workshops that create impact and energize teams.

Customizable Programs for Organizations

Advanced Systems Engineering
Advanced Model-Based Systems Engineering (MBSE)
Technical Leadership Development Forums
Agile Project Management / Enterprise Agility
Software-Defined Futures Transformation
Machine Learning / Software Engineering
Industrial Dev*Ops for Systems Engineering

Caltech | Center for Technology & Management Education

Get started: ctme.caltech.edu
Connect with us: execed@caltech.edu

Determining Reliability Requirements and Testing Costs in the Early Stages of Single Use Medical Product Design

Fritz Eubanks
Copyright ©2010 Battelle Memorial Institute. Published and used by INCOSE with permission.

■ **ABSTRACT**
The production of single use medical devices, particularly for home use by patients, continues to grow, and the reliability of these devices is a primary concern for manufacturers and end-users. The systems engineer tasked with the device development needs methods and tools to establish reliability requirements and provide cost estimates for the testing necessary to show compliance with those requirements. This paper examines methods for determining reliability requirements, the cost of reliability testing for single use medical devices in the design input phase of product development, and how the costs of testing and potential errors can be used to perform trade-off analysis between reliability tolerance and confidence level.

INTRODUCTION

Reliability is a product performance parameter, and consequently shares in the three-way balance between product performance, cost, and time to market. Design for high reliability requires varying combinations of high reliability components, functional redundancy, and periodic overhaul/ maintenance, all of which make the product more expensive to design, build, and test. On the other hand, disregard for reliability makes products more expensive to operate and maintain and leads to customer dissatisfaction and loss of sales.

Medical products can range from simple, single use devices, like tongue depressors and syringes, to large, complex systems like MRI systems and multi-assay in vitro diagnostic devices. Likewise, the complexity of the establishing and meeting the device reliability requirements will vary with device complexity.

Product requirements for medical devices are established during the design inputs phase of product development. (QSR 2009). The product requirements define the performance characteristics, safety and reliability requirements, regulatory requirements, applicable product standards, physical characteristics, and packaging and labeling requirements, among other things. (Trautman 1997). At the same time, project managers and systems engineers begin establishing the design and development plan, including major schedule milestones and overall program costs. Chief among design and development costs is product testing to verify compliance with product requirements. In the case of reliability testing, these costs can be substantial due to the large number of items and/ or amount of time required to obtain statistically sound data that serve to verify product reliability requirements. Methods to estimate testing lot sizes during the design inputs phase can prove valuable for both cost and schedule planning, as well as performing trade-off analyses for refining reliability requirements.

ASSESSING THE RELIABILITY REQUIREMENTS

The Structure of Reliability Requirements

Reliability is defined as "the probability that an item will perform its intended function under stated conditions over a specified interval." Therefore, the reliability goal must include specifications for the following items:

- Measure of success/failure
  - A probability between 0 and 1, or a percentage between 0% and 100%
  - A mean time to failure (MTTF) or mean time between failure (MTBF)
  - A system availability between 0 and 1, or a percentage between 0% and 100%
- Definition of success/failure
  - Success: No downtime, performance parameters within specification, no lost data
  - Failure: no test result, false positive, insufficient output, complete system failure
- Range of normal operating conditions
  - Temperature, humidity, pressure,

vibration, dust/pollution, liquid, power levels
- Interval over which probability of success/failure will be measured
  - Time, cycles, miles
  - Note: this interval is not the same as product life

Examples of good reliability requirements are as follows:

The system shall have mean time before failure of 1000 hours over a one-year period when operating under laboratory conditions where failure is defined as a false positive indication.

The power subsystem shall have a 95% probability of performing in accordance with specifications over 1000 hours in arctic conditions.

The vessel shall remain pressurized at 100±5 psig without operator intervention for 150 hours at 120°F with 99.5% reliability.

Collecting Basic Information

The key reliability issues for any product or system are (RiAC 1996):
- What measures of reliability are important to the end-user?
- What levels of reliability are necessary to meet the end-user's needs?
- How will the manufacturer determine if the required levels of reliability have been achieved?

To answer these questions, it may be necessary to engage in a fact-finding effort that may involve a voice of the customer (VOC) study, benchmarking, and/or market surveys. Through these activities, the manufacturer should to come to alignment with the end user's needs on the following key reliability questions:
- How often will the product be used?
- How many failures per 1000 attempted uses can be tolerated?
- How much operating time per use is expected?
- Who will be the regular user of the product?
- Where and under what conditions will the product be used?
- How is success/failure of the product defined?
- What is the expected life of the product?
  - For single use products, how long will the product be stored before use and under what conditions?
- Will users be compensated for failed items and, if so, how much?

For repairable systems, additional reliability issues must be considered:
- How many product failures can be tolerated over a 3-, 6-, or 12-month period?
- How much product downtime for service/repair of failures can be tolerated?
- Who will be tasked with performing service/repair?
- Will there be a warranty period and for how long?
- How much product downtime for routine maintenance can be tolerated?
- Who will be tasked with performing routine maintenance?
- How much will routine maintenance parts cost and who will pay for it?

Some of the answers to these questions may not be available in the concept/feasibility stage, but need to be considered and, if possible, estimated for the manufacturer to decide on how to position the product from reliability and cost perspectives.

Reliability Requirement Testing Costs

The end user needs to know that reliability goes hand in hand with product cost. "Four nines" reliability is great but may increase the cost of the product to an unacceptable level. Suppose that the end user of the single use syringe demanded 1 failure for every 1000 attempts. Achieving this reliability will likely increase the cost of the device substantially.

The manufacturer needs to define the importance that reliability will have as a performance parameter relative to product cost and time to market. The importance aids in establishing the level of confidence required by the manufacturer when assessing how well the product has met the reliability requirements. This, in turn, allows the systems engineer to make a rough order of magnitude estimate of testing costs, because sample size and test time are driven by the combination of reliability and confidence interval. While testing approaches for more complex reparable and non-reparable system are well studied, simpler single use devices have not received a lot of attention. O'Connor (2002, 357) recommends that statistical acceptance sampling methods can be used for such devices. The success/failure nature of single use devices suggests that statistics of population proportions can be applied. (Devore 2008, 306)

Establishing and testing requirements for large reparable systems is well studied and documented. The remainder of this paper will focus on requirements and testing for single-use devices.

Notation

It is important to note that the reliability values expressed in the following development are not the same as are used for time-based reliability calculations. For this work, the device reliability is the ratio of the number of successes to the number of trials, commonly expressed as R = x/n, where x and n are discrete integer values. The notation is as follows:

Symbol	Definition
R	Required reliability as a population proportion
n	Test sample size
α	Level of significance; probability of type I error
β	Probability of type II error
R′	Potential reliability due to type II error
β(R′)	Probability of type II error when R = R′

SINGLE USE DEVICE RELIABILITY VERIFICATION

Verification Testing Requirements

Verification testing provides the objective evidence that the product meets performance requirements, and that the product is ready for release to production. Devices used for product verification testing need to be equivalent to the device that will be produced for sale and distribution.

In the language of statistical hypothesis testing, the null hypothesis is that the product performance meets the requirement being tested, while the alternative hypothesis is that the performance falls outside the limits of the requirement. Reliability requirements are generally stated as a minimum, for example, at least 95% with 95% confidence. Therefore, requirement verification will take the form of a one-tailed hypothesis test with a null hypothesis that the reliability is greater than or equal to 95%. The concern early in the design phase is to plan for enough tests to provide the required confidence in the validity of the verification test results.

Testing Errors and Sample Size

The two hypothesis test errors are defined as follows (Devore 2008, 288):

A type I error consists of rejecting the null hypothesis when it is true.

A type II error consists of not rejecting the null hypothesis when it is false.

In terms of requirements verification testing, these definitions can be re-written as:

A type I error consists of concluding that the requirement has not be met when it has.

A type II error consists of concluding that the requirement has been met when it hasn't.

While a type I error could result in schedule delays and additional testing cost, a type II error could result in the release of a product that does not meet the reliability requirement. In the context of verification testing, a type II error means that the level of reliability realized in production will be below the level of reliability measured during verification testing.

Statistical confidence is (1 - α), where α is the probability of a type I error. When nR ≥ 10 and n(1 - R) ≥ 10, p has approximately a normal distribution, and the lower confidence limit (LCL) for a one-sided, lower bound test of a population proportion can be computed. Therefore, the minimum sample size needed to establish the confidence interval for 95% reliability using the normal approximation is $n = \frac{10}{1 - 0.95} = 200$. Under the presumption that the reliability of the device will be 95%, and that the desired confidence is 95%, the LCL can be computed (Devore 2008, 266). See equation 1.

Increasing the sample size to 400 would make the LCL around 93.2%. Here is the first point where the manufacturer must define the importance of reliability:

Q1: What lower confidence limit of reliability is acceptable at the desired level of statistical confidence?

If the manufacturer desires 95% reliability with 95% confidence and LCL of 93%, then the necessary sample size can be estimated as (Devore 2008, 267). See equation 2.

Statistical power is (1 – β), where beta is the probability of a type II error. Unlike α, there is not a single value for β. There will be a different β for each value of p contained within the bounds of the alternative hypothesis. For example, if a test of 400 units shows that the reliability is 95% with 95% confidence, there is a 19% probability that the actual population reliability is 92%. In other words, there is an almost 1 in 5 chance that the actual device reliability in production will be below the 95% one-sided lower confidence limit. Here is the second point where the manufacturer must define the importance of reliability:

Q2: What tolerance for type II error (combination of actual reliability in production and probability of realizing that reliability) is acceptable at the desired level of statistical confidence?

The calculation of sample size necessary to properly control both type I and type II errors in reliability verification testing contains 4 variables: the required reliability (R_0), the confidence level (1 - α), the probability of a type II error (β), and the lower bound at which β applies (R′). Continuing our example, the manufacturer desires at least 95% reliability with 95% one-sided confidence (α = 0.05). In addition, the manufacturer feels they can only tolerate a 10% chance (β = 0.10) that the actual reliability in production is as low as 93%. The sample size can be estimated using (Devore 2008 308). See equation 3.

Note that the increased sample size also reduces the confidence interval, resulting in a LCL of 94%. In this case, the desire for a low probability of type II error has driven the sample size to a level that provides 95% confidence that the LCL will be within 1% of the required reliability.

Sample Size Tradeoff Analysis

Considering that pre-production samples for testing can cost anywhere from $50 to $500 each, the cost of parts alone for a sample size of 1176 starts at $58,800 and goes up from there, not to mention the time and effort required to manufacture the preproduction parts for testing. The manufacturer may want to examine options for possibly reducing the lot size for testing. Using the previous development, sample size can be calculated for combinations of acceptable limits of type I and type II errors. However, working with probabilities and sample sizes alone can be a little too abstract for making tradeoff decisions. What the manufacturer really wants at this stage is a rough order of magnitude estimate of the total cost of the reliability testing.

Using experience from previous programs and judicious estimation, the systems engineer can collect some basic parameters used to estimate the costs of conducting verification testing. These values can be used to calculate a simple estimate of testing costs for various sample sizes. See equation 4.

Testing costs are driven by sample size, and in this context, lower is better. However, lower sample size results in a higher probability of type II error, and thus a better chance that the production reliability will be lower than anticipated. The impact of lower reliability in production will be felt as a loss to the manufacturer due to warranty returns, customer dissatisfaction, and potential claims for property damage or personal injury. If the losses can be roughly estimated for each incremental shortfall in reliability, it can provide the basis for a tradeoff against testing costs.

In some cases, the cost of device failures may have been computed as part of the business case used to justify the decision to proceed with design. Otherwise, a rough estimate can be obtained by summing up the estimated probability and severity of each potential outcome of a device failure. Potential outcomes and estimates of severity and probability can be generated from

$$LCL = R - z_\alpha \sqrt{\frac{R(1-R)}{n}} = 0.95 - 1.645 \sqrt{\frac{0.95(1-0.95)}{200}} = 0.925 \quad (1)$$

$$n = R(1-R)\left(\frac{z_\alpha}{(R-LCL)}\right)^2 = 0.95(0.05)\left(\frac{1.645}{0.02}\right)^2 = 322 \quad (2)$$

$$n = \left[\frac{z_\alpha\sqrt{R_0(1-R_0)} + z_\beta\sqrt{R'(1-R')}}{R'-R_0}\right]^2 = \left[\frac{1.645\sqrt{0.95(1-0.95)} + 1.282\sqrt{0.93(1-0.93)}}{0.93-0.95}\right]^2 = 1176 \quad (3)$$

$$Testing\ Cost = [(Sample\ Size) * (Part\ Cost)] + \left[\frac{(Sample\ Size)}{(Testing\ Rate)} * (Labor\ Rate + Facility\ Rate)\right] + (Fixture\ Cost) \quad (4)$$

$$C(failure) = \sum_{i=1}^{m} P(outcome)_i \times C(outcome)_i \quad (5)$$

$$\beta(R') = 1 - \Phi\left[\frac{R - R' - z_\alpha\sqrt{\frac{R(1-R)}{n}}}{\sqrt{\frac{R'(1-R')}{n}}}\right] \quad (6)$$

previous experience with similar devices, or from high level risk assessments. For outcomes involving injury or property loss, Ayyub (2003) and Wilson and Crouch (2001) can be used to estimate costs. Expressing the severity in terms of cost to the manufacturer, the general expression would be equation 5.

where:

C(failure) = cost of a device failure
P(outcome)i = probability of potential outcome i occurring
C(outcome)i = cost of potential outcome i to manufacturer
m = number of potential outcomes identified.

Recall that there will be a different probability of type II error for each value of $R' < R$. For one-sided hypotheses, the probability is calculated as equation 6.

Therefore, the cost of potential type II errors can be expressed as the sum over potential values of R' of the probability of type II error multiplied by the cost associated with products having reliability R' instead of R_0. This calculation is not as intractable as it seems. For moderate values of sample size ($n \geq 400$) with $R = 95\%$, $\beta(90\%)$ is less than 1%.

The cost estimation and trade-off process is best illustrated through the following example.

SINGLE USE MEDICAL DEVICE EXAMPLE

A pharmaceutical manufacturer developed a drug for treating a chronic pain condition. The drug requires intramuscular injection daily, and the manufacturer wanted to develop a one-button, home use solution for making the injection. Answers to the salient questions are as follows:

- How often will the product be used? Once
- How many failures per 1000 attempted uses can be tolerated? 50
- How much operating time per use is expected? No more than 5 seconds
- Who will be the regular user of the product? Adults, 25-80 years old, no physical disabilities
- Where and under what conditions will the product be used? Home use, weekly or monthly, US, Canada, EU
- How is success/failure of the product defined? Success = proper dose delivered to patient's thigh muscle within 5 seconds of activation
- For single use products, how long will the product be stored before use and under what conditions? 2 years at 5°C
- Will users be compensated for failed items and, if so, how much? The cost of the device plus shipping.

Using the information above, the systems engineer can establish the following product reliability requirement:

> The product shall deliver the proper dose to the patient within 5 seconds of actuation with a probability of at least 95% when used in an environmentally controlled interior space with temperature of 15-35°C, humidity of 10-95% RH, and atmospheric pressure of 14.7-10.3 psia following storage at 5°C for no more than 2 years.

The manufacturer believes that a single use device with 95% reliability provides a good balance between performance and cost. Production volumes are estimated at 50,000 devices per year. The trade-off process starts by determining required sample size based on lower confidence limit and the level of confidence in achieving that limit in accordance with Equation (2). Table 1 provides the trade-offs between confidence, LCL, and sample size for R=95%.

Note that using sample sizes below 200 will require different treatment due to the restriction that $n(1-R) \geq 10$ for the normal distribution assumption of R to apply. The estimated costs for reliability testing as a function of sample size calculated using Equation 4 are shown in Table 2.

Assume a preliminary selection of the 90% confidence level. Based on the confidence level, the additional cost of development must be weighed against the possible additional cost of operation due to a higher-than-expected failure rate, as measured by the probability of a Type II error. Calculated probabilities for Type II error for each level of combination of R' and LCL in accordance with Equation 6 are shown in Table 3.

Assume that the manufacturer has performed a rough cost assessment of potential failure outcomes as follows:

Potential outcome	Probability	Cost
Serious injury	0.0001	$500,000
Moderate injury	0.005	$45,000
Minor injury	0.05	$6000
No injury – returned item	0.94	$500

Table 1. Lot Size for Testing, $R_0 = 95\%$

	LCL				
Confidence	94%	93%	92%	91%	90%
95%	1286	322	143	81	52
90%	781	196	87	49	32
85%	511	128	57	32	21
80%	337	85	38	22	14

Table 2. Estimated Cost of Reliability Testing

Part Cost =	$200 each
Testing Rate =	$6/hour
Labor Rate =	$100/hour
Fixture Cost =	$5,000
Facility Rate =	$75/hour

	LCL				
Confidence	94%	93%	92%	91%	90%
95%	$299,708	$78,792	$37,771	$23,563	$16,917
90%	$183,979	$49,917	$24,938	$16,229	$12,333
85%	$122,104	$34,333	$18,063	$12,333	$9,813
80%	$82,229	$24,479	$13,708	$10,042	$8,208

Table 3. Probability of Type II Error

R =	95%				
Confidence =	90%				
	LCL				
R′	94%	93%	92%	91%	90%
0.94	0.18	0.55	0.69	0.74	0.78
0.93	0.00	0.20	0.42	0.55	0.63
0.92	0.00	0.04	0.21	0.37	0.47
0.91	0.00	0.01	0.09	0.23	0.34
0.90	0.00	0.00	0.03	0.13	0.24

Table 4. Potential Loss due to Type II Error

R =	95%				
Confidence =	90%				
	LCL				
R′	94%	93%	92%	91%	90%
0.94	$9,593	$28,971	$35,813	$38,866	$40,554
0.93	$290	$20,956	$44,143	$57,452	$65,315
0.92	$1	$6,995	$33,625	$57,618	$74,303
0.91	$0	$1,399	$19,352	$47,183	$71,426
0.90	$0	$191	$9,129	$33,734	$61,682

Using Equation 5, the cost per device failure is estimated to be $1,045. For a population of 500,000 devices, an additional failure rate of 1% represent 5000 devices, for a potential annual loss of $5,225,000. Calculated potential losses for each level of combination of probability and magnitude of Type II error and their totals are shown in Table 4.

Our rough calculations indicate that a reliability test program that exhibits 90% confidence in a lower reliability bound of 93% is a reasonable trade-off of testing cost versus potential loss due to reliability uncertainty. A sample size for verification testing of 1176 is calculated using Equation 3. Note that the increased sample size brings the total estimated testing costs to around $94,000, but still represents a good trade when compared to the potential cost of lowering the acceptable LCL to 92%.

CONCLUSIONS AND FUTURE WORK

The production of single use medical devices, particularly for home use by patients, continues to grow, and reliability of these devices is a primary concern. The systems engineer tasked with the device development needs methods and tools to establish reliability requirements and provide cost estimates for the testing necessary to show compliance with those requirements. This paper presented a set of basic questions for determining reliability requirements during the design input stage. We also demonstrated that the cost of reliability testing for single use medical devices can be estimated during the design input stage, and the results used to perform trade-off analysis of between required tolerance, confidence level, and cost. We will continue to develop and refine the questions we ask to determine the proper reliability requirements, and the cost models for providing rough order of magnitude cost estimates as we apply them to future product development projects. ■

REFERENCES

- Ayyub, B. 2003. *Risk Analysis in Engineering and Economics*. Boca Raton, US-FL: Chapman & Hall/CRC Press LLC.
- Devore, J. 2008. *Probability and Statistics for Engineers and Scientists, 7th Edition*. Belmont, US-CA: Duxbury.
- ISO 2007. ANSI/AAMI/ISO 14971:2007 – Medical devices – Application of risk management to medical devices. Arlington, VA: Association for the Advancement of Medical Instrumentation.
- O'Connor, P. 2002. *Practical Reliability Engineering, Fourth Edition*. West Sussex, UK: John Wiley and Sons, Ltd.
- QSR 2009. Code of Federal Regulations Title 21 – Food and Drugs, Chapter I – Food and Drug Administration, Department of Health and Human Services, Subchapter H – Medical Devices, Part 820 – Quality System Regulation. Revised April 1, 2009.
- RiAC. 1996. Blueprints for Product Reliability Part 1 – Defining Reliability Programs. Reliability Information Analysis Center. Published May 15. http://theriac.org/DeskReference/viewDocument.php?id=280&Scope=reg.
- Trautman, K. 1997. *The FDA and Worldwide Quality System Requirements Guidebook for Medical Devices*. Milwaukee, US-WI: ASQ.
- Wilson, R., and E. Crouch. 2001. *Risk Benefit Analysis*. Cambridge, US-MA: Harvard University Press.

ABOUT THE AUTHOR

Fritz Eubanks has served as a systems engineer with Battelle's Health and Life Science's Medical Device Solutions group, where he has performed system-level design and analysis of both medical and commercial products and safety risk management lead engineer. His experience includes civil service in quality engineering with the Air Force Logistics Command, and medical and commercial product development at Battelle. He received a BS in mechanical engineering from Kansas State University in 1982, and MS and PhD from Ohio State University in 1992 and 1996, respectively. He has been a member of INCOSE and an ASQ certified reliability engineer.

MARCH 2023
VOLUME 26/ ISSUE 1

A Concept for Set-based Design of Verification Strategies

Pen Xu, xupeng@vt.edu; and **Alejandro Salado,** alejandrosalado@arizona.edu
Copyright ©2019 by Pen Xu and Alejandro Salado. Permission granted to INCOSE to publish and use.

■ **ABSTRACT**
In current practice, a verification strategy is defined at the beginning of an acquisition program and is agreed upon by customer and contractor at contract signature. Hence, the resources necessary to execute verification activities at various stages of the system development are allocated and committed at the beginning, when a small amount of knowledge about the system is available. However, contractually committing to a fixed verification strategy at the beginning of an acquisition program fundamentally leads to suboptimal acquisition performance. Essentially, the uncertain nature of system development will make verification activities that were not previously planned necessary and will make some of the planned ones unnecessary. To cope with these challenges, this paper presents an approach to apply set-based design to the design of verification activities to enable the execution of dynamic contracts for verification strategies, ultimately resulting in more valuable verification strategies than current practice.

INTRODUCTION

Verification activities, which usually take the form of a combination of analyses, inspections, and tests, consume a significant part, if not the biggest part, of the development costs of large-scale engineered systems (Engel 2010). Verification occurs at various levels of a system's decomposition and at different times during its life cycle (Engel 2010). Under a common master plan, low level verification activities are executed as risk mitigation activities, such as early identification of problems, or because some of them are not possible at higher levels of integration (Engel 2010). Therefore, a verification strategy is defined "aiming at maximizing confidence on verification coverage, which facilitates convincing a customer that contractual obligations have been met; minimizing risk of undetected problems, which is important for a manufacturer's reputation and to ensure customer satisfaction once the system is operational; and minimizing invested effort, which is related to manufacturer's profit" (Salado 2015). Essentially, verification activities are the vehicle by which contractors can collect evidence of contractual fulfillment in acquisition programs.

In current practice, a verification strategy is defined at the beginning of an acquisition program and is agreed upon by customer and contractor at contract signature. Hence, the resources necessary to execute verification activities at various stages of the system development are allocated and committed at the beginning, when a small amount of knowledge about the system is available (Engel 2010). However, the necessity and value of a verification activity cannot be measured independently of the overall verification strategy (Salado and Kannan 2018b, Salado et al. 2018). Instead, the necessity to perform a given verification activity depends on the results of all verification activities that have been previously performed. For example, testing the mass of a component is considered more necessary if a previous analysis has shown low margin with respect to the success criterion than if the analysis has shown ample margin. Thus, contractually committing to a fixed verification strategy at the beginning of an acquisition program fundamentally leads to suboptimal acquisition performance. Essentially, the uncertain nature of system development will make verification activities that were not previously planned necessary and will make some of the planned ones unnecessary. The former can be handled through change requests (CR), but they require unplanned financial investments. The latter can be recovered in a few cases through negative change requests, but, in general, they imply a financial waste because the investment has been committed to the contractor. Hence, we contend that dynamic contracting of verification activities is necessary to guarantee optimality of acquisition programs in this area.

Informed by the benefits of set-based design in conceptual design (Singer et al. 2009), this paper presents a concept to apply set-based design to design verification strategies. Like its application in the conceptual design phase, an initial set of possible verification strategies is reduced as the system development progresses by evaluating the knowledge built by the results of verification activities and the available investment opportunities. In this way, verification activities can be contracted on each epoch in which the set is reduced by leveraging the knowledge generated while executing the verification strategy dynamically.

This paper is organized as follows. First, background material is provided on

set-based design and its use in systems engineering, as well as on the basic verification definitions used in this paper. Second, the concept proposed in this paper for set-based designing verification activities is presented. Third, an application example of the proposed approach is shown. Finally, a summary of the conclusions is given.

BACKGROUND

Set-Based Design (SBD)

As previously described, verification strategies are defined in current practice at the beginning of an acquisition program and are agreed upon by customer and contractor at contract signature, when a small amount of knowledge about the system is available (Engel 2010). Such lack of knowledge in early design activities motivated the emergence of set-based design (Bernstein, 1998). Set-based design is built on the principle of working simultaneously with a plethora of design alternatives, instead of converging quickly to a single option (Bernstein 1998). As the knowledge about the system increases, suboptimal alternatives are discarded until a preferred one remains (Bernstein 1998). A key aspect is that discarding is not an activity at a given point of time, like a traditional trade-off, but a time-continuous activity that occurs as new knowledge is available (Bernstein 1998). A formal formulation of set-based design and how it makes product development resilient against changes in external factors is given in (Rapp et al. 2018).

Set-based design has been successfully applied in the conceptual stages of naval systems (Singer et al. 2009), graphic industry products (Raudberget 2010), automotive products (Raudberget 2010), and aeronautic systems (Bernstein 1998), among others. Historical analysis of the use of set-based design has shown that it inherently eliminates root causes of rework in system development (Kennedy et al. 2014). Researchers have integrated set-based design with tradespace exploration to further strengthen its value by leveraging the numerous solutions that tradespace exploration provides to generate the initial set (Small et al. 2018). However, empirical research about the implementation of set-based design in an industrial setting showed that there are some discrepancies as to how to operationalize the approach (Hansen and Muller 2012). It remains to explore if this was an anecdotal episode or if it happens in general.

Verification

In this paper, a *verification strategy* is understood to be a set of verification activities organized as an acyclic directed graph where the verification activities are modeled as nodes and the edges represent their information influence (Salado and Kannan 2018a). A *verification activity* is understood to be the collection of information about a specific aspect of the system under development and *verification evidence* refers to such information (Salado and Kannan 2019).

A verification strategy can be modeled as a Bayesian network to capture the way engineers build confidence on the state of the system as verification evidence becomes available (Salado and Kannan 2019). The basic structure of such Bayesian model is given by three subgraphs (Salado and Kannan 2019):

1. A graph that captures the temporal sequence and information dependencies between the different verification activities within the verification strategy.
2. A graph that captures the properties of the system architecture, that is, how the different parameters of the system and its building components relate to each other.
3. A graph that captures the ability of the verification activities to provide information about one or more system parameters.

This modeling approach forms the basis for the mathematical model underlying the application of set-based design to the design of verification strategies presented in this paper. The basic notation is represented in Figure 1. System parameters are denoted by θ_i and verification activities by V_i. Arrows represent information dependencies.

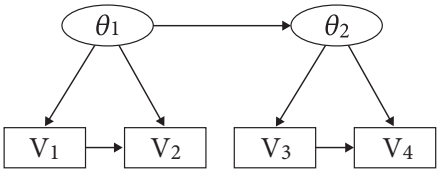


Figure 1. Example of modeling notation

In the example in Figure 1, θ_1 could represent, for example, the performance of a prototype, which is verified through an analysis V_1 and a test V_2 (such that the result of the analysis shapes the confidence on the expected result of the test). Such prototype performance shapes the confidence on the performance of the actual system θ_2, which is verified through verification activities V_3 and V_4.

CONCEPT: SET-BASED DESIGN OF VERIFICATION STRATEGIES

The approach presented in this paper is graphically compared against the current paradigm for contracting verification activities in Figure 2. In the current paradigm (top part of the figure), a contract for a verification strategy is fixed at the beginning of the system development program. The strategy is defined by the black dots connected by the orange line, which represent the verification activities that will be executed throughout the system development.

Without loss of generality, it is possible to assume that such verification strategy was determined optimal at the beginning of the program, that is, with the knowledge available at that point in time. Consider now that the verification activity V_1 at t_1 shows a tight margin with respect to the expected result of the activity. This may lead to a lower-than-expected confidence on the system being absent of errors that triggers the need for an additional, unplanned verification activity V_2 at t_1. Because the contract was fixed, such an activity needs to be contractually introduced through a change request.

Consider on the contrary, that the verification activity V_1 at t_3 showed much better results than previously expected. This may yield a higher-than-expected confidence on the system being absent of errors, potentially making verification activity V_2 at t_3 unnecessary or of little value, because of how confidence builds up on prior information (Salado and Kannan 2018b, Salado et al. 2018).

Consider now the proposed set-based design approach, depicted on the bottom side of Figure 2. In this case, an optimal strategy is also determined at t_1. However, because the value of verification activities may change as results become available (Salado and Kannan 2018b), a set (represented by the dotted lines connecting the dots) is considered instead of just one strategy, and only the first verification activity V_1 at t_1 is contracted at this point. This set is the set of all possible verification strategies that are consistent with the optimal verification strategy (that is, formed by all verification strategies that have the first activity in common).

Assume then that verification activity V_1 at t_1 provides low margin with respect to the expected results, as was the case before. With the updated confidence level, a new optimal strategy is selected within the remaining set. Then, the set is reduced to include only those verification activities that are consistent with the new optimal strategy. In this way, verification activity V_2 at t_1 is contracted as well. The process of identifying new optimal strategies based on updated confidence and reducing the set of remaining verification activities to those consistent with the new optimal strategy, continues at each *t*.

Assume later in the system development that, as was the case when describing the

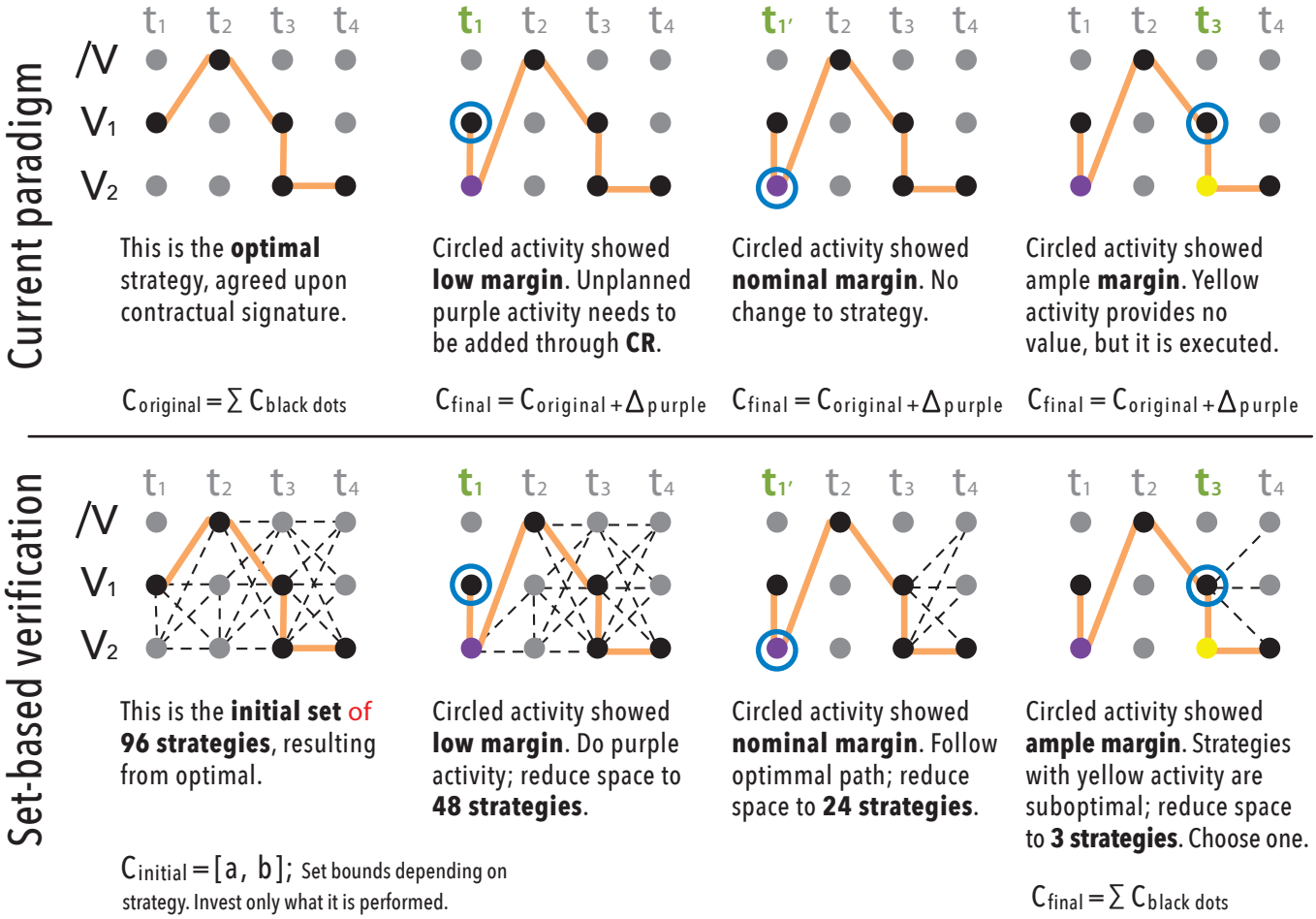


Figure 2. Current versus set-based approaches for designing verification strategies (C: cost of executing verification; t_i: verification events; /V: no verification; V_i: verification activity)

current paradigm, verification activity V_1 at t_3 shows ample margin with respect to the expected result. The next assessment of the remaining optimal path yields a set of verification strategies that do not include verification activity V_2 at t_3. Based on this result, V_2 is not contracted at t_3. Consequently, this approach does not waste resources in activities that become no longer needed as verification evidence becomes available.

APPLICATION EXAMPLE

In this section, we provide a notional example of how the proposed set-based design approach to design verification strategies operates and compares against the current static paradigm.

Case Description

The notional verification strategy in Figure 1 is used for this example. All nodes are assigned binary values for computational simplicity. This simplification does not affect the purpose of this paper. System parameter nodes may take the values of *no error* or *error*, which will be denoted by $\neg e$ and e, respectively. Verification activity nodes may take the values of *pass* or *fail*. A time vector $(T_1,...,T_n)$ is defined, where the element T_i precedes temporally the element T_i+1 for each $i = 0,..., n-1$. No specific time unit is employed, because only temporal order is relevant to the example. Each element in the vector will be referred to as time event.

It is assumed that at most one verification activity is performed at each time event and that any given verification activity is performed at most once during the entire verification strategy. Furthermore, restrictions on the feasibility to perform a given verification activity at a given time event have been defined and are listed in Table 1. The restrictions are intended to capture realistic constraints that may exist on the feasibility to perform a given verification activity at some point in the system development. For example, it is likely that tests on prototypes can happen since an earlier time event than tests on the final product.

Table 1. Activity contraint table

Time event	Feasible verification activities
T_1	$L(T_1) = \{V_3, V_5\}$
T_2	$L(T_1) = \{V_3, V_4, V_5\}$
T_3	$L(T_1) = \{V_3, V_4, V_5\}$
T_4	$L(T_1) = \{V_5, V_6\}$
T_5	$L(T_1) = \{V_5, V_6\}$
T_6	$L(T_1) = \{V_5, V_6\}$

*$L(T_i)$={all feasible verification activities at T_i}

The goodness or preference of a verification strategy will be determined by three main factors: (1) its cost of execution, which is given by the fixed cost to execute each of its verification activities; (2) the expected cost to repair/rework the system when deemed necessary to do so as a function of the available verification evidence; and (3) the expected impact cost of the system exhibiting an error once deployed. Mathematically, the expected cost of a verification strategy S has modeled as:

Verification strategy S

$$E\left[C_T(S)\right] = \sum_{V\in\mathbf{V}} C_V(\mathrm{V}) + \sum_{k=1}^{o}\sum_{j=1}^{n}\sum_{V\in L(T_j)} P(v)\,P(\theta_{\mathrm{jk}}\,|\,v)\,\delta(\theta_{\mathrm{jk}}\,|v)\,C_R(\theta_{jk}) + \sum_{k=1}^{o}\sum_{V\in\mathbf{V}^\star} P(v)\,P(\theta_k = e\,|\,v)\,C_I(\theta_k = e) \quad (1)$$

where:

$C_V(V)$ is the fixed cost to execute verification activity V,

V Is the set of verification activities included in the verification strategy S,

v is a specific vector of verification results,

$P(\theta_{\mathrm{jk}}\,|v)$ is the confidence level of the kth system parameter node at T_j given the verification results v,

$\delta(\theta_{\mathrm{jk}}\,|v)$ is the indicator function that equals 1 if $P(\theta_{\mathrm{jk}}\,|\mathrm{v}) \le H_l$, where H_l is a decision threshold, as will be explained in the next paragraph; otherwise its value is 0,

$C_R(\theta_{\mathrm{jk}})$ is the rework cost necessary to recover a failure detected during verification at T_j,

$\mathbf{V}^\star$ is the set of verification results and rework efforts possible as per the previous rework decisions given the set of verification activities **V**,

$\mathrm{P}(\theta = \mathrm{e}\,|\,\mathrm{v})$ is the probability that the system exhibits an error, given the specific verification results v, and

$C_I(\theta = \mathrm{e})$ is the financial impact of the system exhibiting an error once it is operational.

The treatment of rework costs deserves additional discussion. A failed verification activity does not necessarily lead to rework; since rework is only necessary if worth doing. An automated rework decision process, caricaturized in Figure 3, is used in this paper. Two confidence thresholds $\{H_l, H_u\} = \{0.4, 0.95\}$ distinguished between three decision zones, which are defined such that:

1. Zone 1 reflects a confidence state that is considered not acceptable. Therefore, if the confidence on the system being absent of errors drops to Zone 1, then a rework activity is executed. The rework activity results in the confidence increasing to the level it would be, had the verification activity yielded *pass* results. This is meaningful because the purpose of the verification activity that failed was to achieve certain confidence level.
2. Zone 2 reflects a confidence state that is in line with the confidence expected as the execution of the verification strategy progresses. Therefore, if (i) the confidence on the system being absent of errors falls in Zone 2 and (ii) the confidence level expected at completion of the verification strategy — assuming all remaining activities pass — falls in Zone 3, then the execution of the verification strategy continues as planned. If this condition is not met, then a rework activity is planned until such an objective is reached.
3. Zone 3 reflects a confidence state that does not require the collection of additional knowledge; the engineer is *convinced* about the correct function of the system. Therefore, if the confidence on the system being absent of errors falls in Zone 3, rework activities are not executed. In addition, reaching Zone 3 implies for the set-based approach presented in this paper (with the corresponding dynamic contracting structure) that no other verification activity will be executed, and the system can be deployed. However, for the benchmark (with static contracting), it is assumed that remaining pre-contracted verification activities will still be executed.

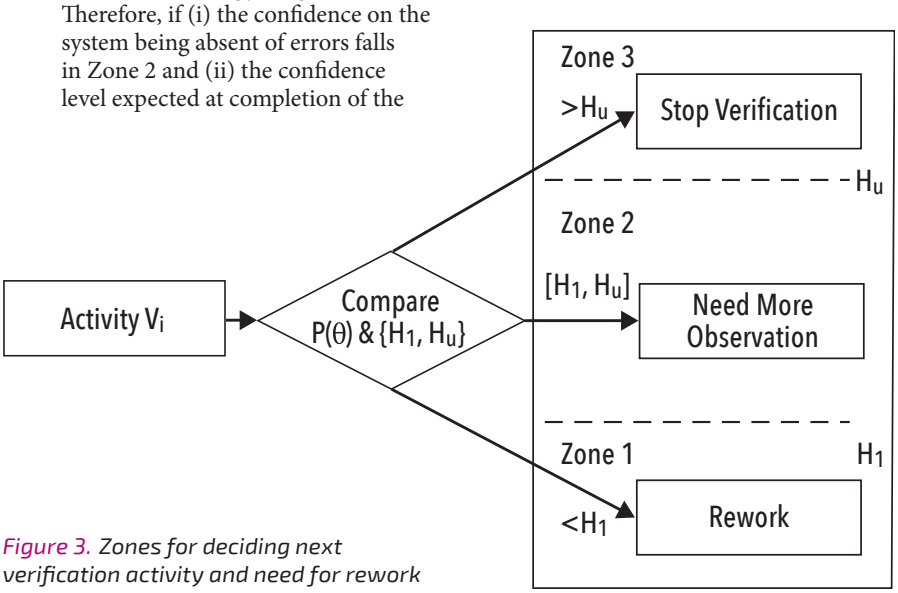


Figure 3. Zones for deciding next verification activity and need for rework

Probability assignments use synthetic data and are given in the Appendix. Following the modeling approach presented in (Salado and Kannan 2019), prior beliefs are assigned to system parameter nodes, which capture the initial belief on the state of the system (that is, being absent of errors), and conditional probability tables are created for the verification activity nodes. Posterior beliefs are calculated for system parameters through Bayesian update of the outcomes of the verification activity nodes. Probability update was conducted in this study using the Bayesian network toolbox for MATLAB®, which estimates the posterior probabilities of all nodes by the variable elimination method.

Cost values employed in this example, given in Table 2, are also synthetic, but reasonable. The following assumptions have been made: (1) rework cost increases with time, (2) the impact cost during deployment is much larger than the rework cost and the verification cost; (3) rework cost is in general higher than verification execution cost; and (4) verification execution cost is positively related to the information it yields.

Generation of Strategies

The verification strategy employed to model the benchmark design/contracting approach is set to be the optimal strategy before executing any verification activity; that is, before the first-time event. Then, the strategy may evolve to include additional verification activities in line with the description given in the previous section.

The verification strategy employed to model the proposed set-based and dynamic contracting approach is adaptively defined at each time interval. Adaptation is performed by choosing the optimal path of remaining activities right after each verification result is obtained, following also the guidelines described in

the previous section. Hence, at each time interval, a set of potential future verification strategies is kept. The corresponding algorithm is shown in Table 3.

The goodness of the proposed set-based approach to design verification strategies with respect to the benchmark approach is assessed based on the expected cost of using each approach. This cost is calculated by considering the expected cost resulting from every possible verification strategy that could be executed when using the benchmark and the proposed set-based approach. As has been described, the resulting verification strategy at completion of all time intervals may differ in both approaches from the optimal one selected before the first-time interval. In the case of the proposed set-based approach, this difference is inherent to the approach. In the case of the benchmark, the difference may only result from the need to incorporate additional activities that were not previously planned.

Results

Given the constraints in Table 1, an initial set of 198 verification strategies could be enumerated before the first-time interval. Among them, the optimal one is $S_1 = (V_1, V_2, NoV, V_3, V_4, NoV)$, where *NoV* indicates that no verification activity is executed at that time interval. This strategy has an expected total cost of $3,226k and an initial confidence on the system being absent of errors of 0.76. As discussed, S_1 is used as the baseline verification strategy for the benchmark.

As an example, the evolution of one of the paths for the proposed set-based approach is described here. V_1 is executed in the first-time interval because it is part of the optimal strategy identified before initiating the execution of the verification strategy. If the verification activity passes, the number of verification strategies remaining in the set reduces to 55 (all strategies that begin with V_1) and the confidence on the system being absent of error increases to 0.84 (as determined through Bayesian update of Figure 1). The optimal verification strategy out of the remaining set becomes $S_2 = \{V_1, V_2\ V_3, V_4, NoV, NoV\}$, with a lower expected cost of $2,994k. On the other hand, if the activity fails, the set of remaining verification activities would contain 115 elements and the confidence on the system being absent of error would drop to 0.57. Since this level is still larger than 0.40, the rework activity would not be entertained yet. The process repeats again by identifying a new optimal strategy and reducing the set accordingly until the verification activity on the last time interval is executed.

Table 2. Cost values

V_i	$C_V(V_i)$($k)	T_j	C_R at T_j($k)	System error	$C_i(\theta_k)$($k)
V_1	50	1	100	θ_2	60,000
V_2	100	2	200		
V_3	75	3	300		
V_4	200	4	400		
		5	500		
		6	600		

Table 3. Algorithm to generate verification strategies

Dynamic Contracting Strategy (N, ACT, C)

Input: N —— Bayesian Net; ACT —— Activity Constraint Table
C —— Cost Table

Output: $S_{opt} = \{V_{opt}(T_1), V_{opt}(T_2), \ldots V_{opt}(T_n)\}$

1: **For** t in $T_1 : T_n$

2: Generate all feasible paths $VS_{ti} = \{S_1, S_2, \ldots S_m\}$ at time point t;

3: Evaluate the expected cost of all verification paths;

4: Select the minimum one $S = \{V_{opt}(T_1), V_{opt}(T_2), \ldots V_{opt}(T_t)\ V(T_{t+1}) \ldots V(T_n)\}$ and update the optimal path $S_{opt} = \{V_{opt}(T_1), V_{opt}(T_2), \ldots V_{opt}(T_t)\}$;

5: Collect the results of $V(T_t)$ and set the evidence adaptively;

6: Update the Bayesian network and ACT

7: **End**

8: **Return** the optimal path $S_{opt} = \{V_{opt}(T_1), V_{opt}(T_2), \ldots V_{opt}(T_n)\}$

Table 4. Table of all cost items at T_7

Path Number	Path Probability (*PP*)	$P(\theta_2 = \neg error)$	$E[C_I]$	C_R	C_V	Path Cost ($CP=E[C_I]+ C_R+ C_V$)
1	0.0295	0.9077	5538	700	425	6663
2	0.0116	0.9077	5538	200	425	6163
3	0.1169	0.9657	2058	200	350	2608
4	0.0265	0.9077	5538	500	425	6463
5	0.0104	0.9077	5538	0	425	5963
6	0.1051	0.9657	2058	0	350	2408
7	0.0446	0.9364	3816	300	225	4341
8	0.0554	0.9364	3816	0	225	4041
9	0.0936	0.9316	4104	500	425	5029
10	0.0449	0.9316	4104	0	425	4529
11	0.4615	0.9750	1500	0	350	1850

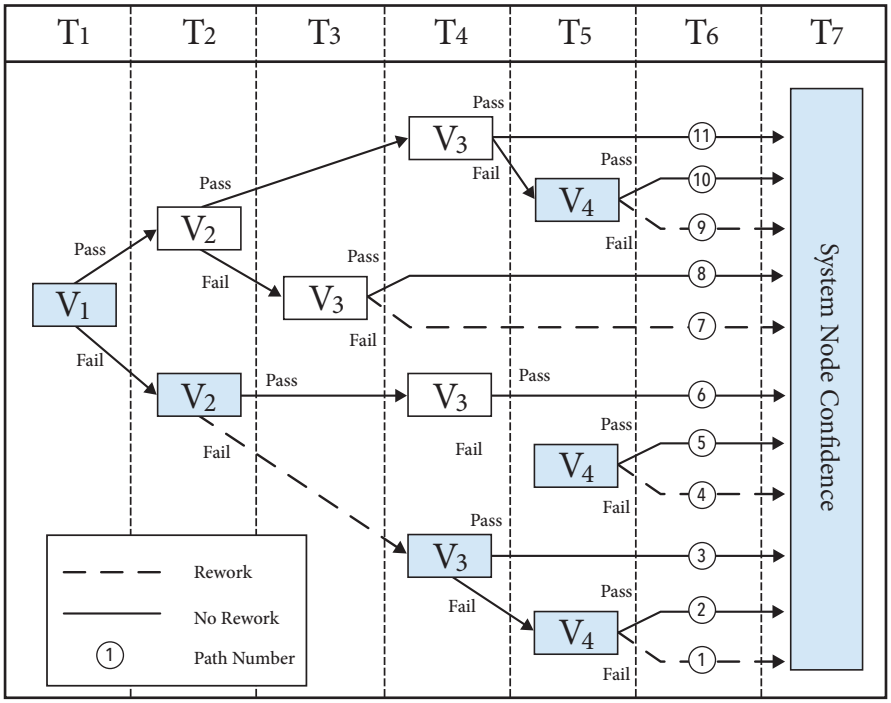


Figure 4. Verification path tree

The possible set reductions lead to 11 feasible paths for the proposed set-based approach. As illustrated in Figure 4, the set of all possible paths could be represented as a tree plot. The expected cost of each approach to design verification strategies is calculated as the sum of the cost of each path weighted by its resulting probability of occurrence. The probability of occurrence for each path is computed as the product of all the probabilities of all activities along the branch. Similarly, the benchmark could yield 16 possible paths. All paths are shown in Figure 5 (dotted, red lines represent benchmark paths; solid, blue lines represent set-based paths). The vertical axis represents the total expected cost of the verification strategy on each time interval. The resulting cost is given therefore after completion of the last time interval (to the right extreme in the plot). The total expected cost of the set-based approach is

$$\sum_{i=1}^{11} PP_i \times PC_i = \$3{,}004\text{k},$$

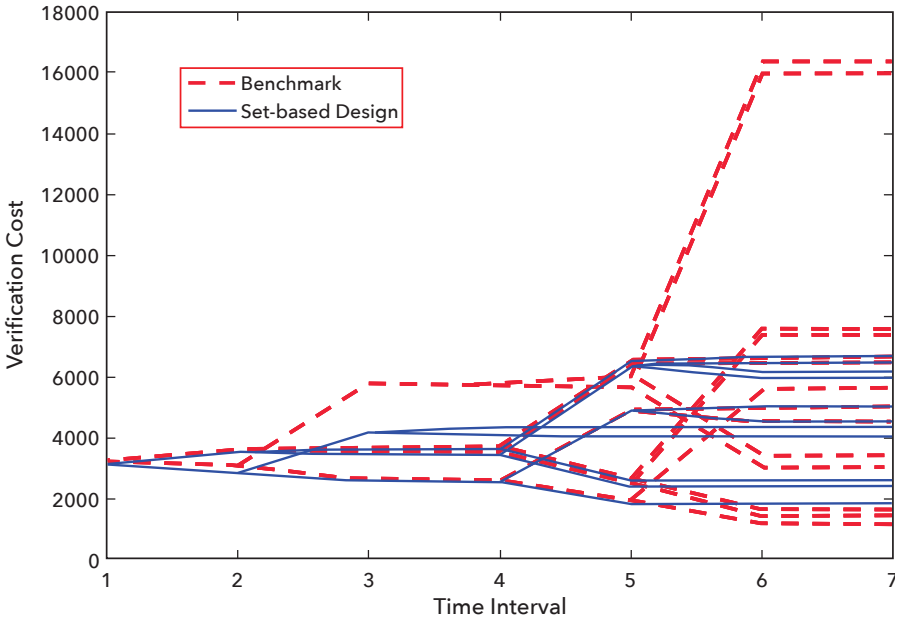


Figure 5. Comparison between set-based design and traditional strategy

which is smaller than that of the benchmark, $3,214k. This result provides an indication that the proposed approach yields indeed more valuable verification strategies than the benchmark, although additional cases need to be run to confirm this result.

Figure 5 provides in addition an interest insight about the properties of the proposed set-based approach to design verification strategies and contract verification activities. As can be seen, the amplitude of the tree corresponding to the benchmark approach (red dotted line) is larger than that of the set-based design method (blue solid line). This indicates that the benchmark approach responds more slowly to adjusting its parameters than the set-based design approach when receiving information from verification evidence. In cost control terms, this indicates that the benchmark approach is inefficient when compared against the proposed set-based approach.

CONCLUSIONS

This paper has presented and demonstrated the capability to use a set-based approach to design verification strategies and contract dynamically verification activities. The study employs Bayesian networks to model, through Bayesian inference, how the evidence provided by verification activities is used to shape the confidence on the system being absent of errors. The value of a verification strategy is determined as a function of its expected total cost, given by the fixed cost to execute its verification strategies, the expected cost of rework in case it is necessary, and the financial impact generated by the system exhibiting an error while operational. A notional example with synthetic data has been used to assess the performance of the proposed approach against a benchmark that represents current approaches in industrial and government settings to design and contract verification strategies; based on point design methods and static contracts that only vary if gaining additional confidence is needed.

The study presents certain limitations that need to be addressed in future research. Primarily, additional scenarios, as well as sensitivity analyses, are needed to increase the robustness of and confidence on the findings indicated in this paper. In fact, it is not evident at this point how the source data, both in terms of prior probabilities, resulting Bayesian networks, and cost values influence the behavior of the benchmark and of the proposed approach. Furthermore, some aspects from real-life applications have been abstracted out in this study and may need to be incorporated into future studies. For example, verification activities may be executed at the same time intervals. In

addition, there may be certain contractual and pragmatic restrictions associated to activate or plan for potential activities, as well as from a contractor's perspective to commit to an open contract and risks to the customer from the contractor not necessarily committing to future activities that need to be accounted for when implementing the resulting dynamic contracting. Finally, the effects of scale on the feasibility of the proposed approach need to be studied. ■

APPENDIX Conditional Probability Tables:

Conditional Probability Table – a. $P(\theta_1)$

θ_1	Probability assignment
Error	0.8
No Error	0.2

Conditional Probability Table – b. $P(\theta_2 \mid \theta_1)$

θ_1	θ_2	Probability assignment
Error	Error	0.8
Error	No Error	0.2
No Error	Error	0.1
No Error	No Error	0.9

Conditional Probability Table – c. $P(V_1 \mid \theta_1)$

θ_1	V_1	Probability assignment
Error	Fail	0.7
Error	Pass	0.3
No Error	Fail	0.2
No Error	Pass	0.8

Conditional Probability Table – d. $P(V_2 \mid V_1, \theta_1)$

θ_1	V_1	V_2	Probability assignment
Error	Fail	Fail	0.9
Error	Fail	Pass	0.1
Error	Pass	Fail	0.6
Error	Pass	Pass	0.4
No Error	Fail	Fail	0.2
No Error	Fail	Pass	0.8
No Error	Pass	Fail	0.1
No Error	Pass	Pass	0.9

Conditional Probability Table – e. $P(V_3 \mid \theta_2)$

θ_2	V_3	Probability assignment
Error	Fail	0.9
Error	Pass	0.1
No Error	Fail	0.4
No Error	Pass	0.6

Conditional Probability Table – f. $P(V_4 \mid V_3, \theta_2)$

θ_1	V_1	V_2	Probability assignment
Error	Fail	Fail	0.9
Error	Fail	Pass	0.1
Error	Pass	Fail	0.4
Error	Pass	Pass	0.6
No Error	Fail	Fail	0.3
No Error	Fail	Pass	0.7
No Error	Pass	Fail	0.1
No Error	Pass	Pass	0.9

REFERENCES

- Bernstein, J. I. 1998. *Design methods in the aerospace industry: looking for evidence of set-based practices*. MSc, Massachusetts Institute of Technology.
- Engel, A. 2010. *Verification, Validation, and Testing of Engineered Systems*, Hoboken, US-NJ: John Wiley & Sons, Inc.
- Hansen, E. and G. Muller. 2012. 11.3.1 "Set-based design – the lean tool that eludes us; Pitfalls in implementing set-based design in Kongsberg Automotive." *INCOSE International Symposium*, 22: 1603-1618.
- Kennedy, B. M., D. K. Sobek and M. N. Kennedy. 2014. "Reducing Rework by Applying Set-Based Practices Early in the Systems Engineering Process." *Systems Engineering* 17: 278-296.

- Rapp, S., R. Chinnam, N. Doerry, A. Murat, and G. Witus. 2018. "Product development resilience through set-based design." *Systems Engineering*, 21: 490-500.
- Raudberget, D. 2010. "Practical Applications of Set-Based Concurrent Engineering in Industry." *Journal of Mechanical Engineering*, 56: 685.
- Salado, A. 2015. "Defining Better Test Strategies with Tradespace Exploration Techniques and Pareto Fronts: Application in an Industrial Project." *Systems Engineering*, 18: 639-658.
- Salado, A., and H. Kannan. 2018a. "A mathematical model of verification strategies." *Systems Engineering*, 21: 583-608.
- Salado, A., and H. Kannan. 2018b. "Properties of the Utility of Verification." *IEEE International Symposium in Systems Engineering*. Rome, IT, 1-3 October.
- Salado, A., and H. Kannan. 2019. "Elemental Patterns of Verification Strategies." *Systems Engineering*, 22:370–388.
- Salado, A., H. Kannan, and F. Farkhondehmaal. 2018. "Capturing the Information Dependencies of Verification Activities with Bayesian Networks." Conference on Systems Engineering Research (CSER). Charlottesville, US-VA, 8-9 May.
- Singer, D. J., N. Doerry, and M. E. Buckley. 2009. "What Is Set-Based Design?" *Naval Engineers Journal*, 121: 31-43.
- Small, C., R. Buchanan, E. Pohl, G. S. Parnell, M. Cilli, S. Goerger, and Z. Wade. 2018. "A UAV Case Study with Set-based Design." INCOSE International Symposium, 28: 1578-1591.

ABOUT THE AUTHORS

Peng Xu has a PhD from the Grado Department of Industrial and Systems Engineering at Virginia Tech. He received his MS in mechanical engineering from National Cheng Kung University in 2015 and his BS in mechanical engineering from Shandong University in 2013. His research interests include complex system diagnosis, dynamic decision making, and knowledge elicitation.

Dr. Alejandro Salado has over 15 years of experience as a systems engineer, consultant, researcher, and instructor. He is currently an associate professor of systems engineering with the Department of Systems and Industrial Engineering at the University of Arizona. In addition, he provides part-time consulting in areas related to enterprise transformation, cultural change of technical teams, systems engineering, and engineering strategy. Alejandro conducts research in problem formulation, design of verification and validation strategies, model-based systems engineering, and engineering education. Before joining academia, he held positions as systems engineer, chief architect, and chief systems engineer in manned and unmanned space systems of up to $1B in development cost. He has published over 100 technical papers, and his research has received federal funding from the National Science Foundation (NSF), the Naval Surface Warfare Command (NSWC), the Naval Air System Command (NAVAIR), and the Office of Naval Research (ONR), among others. He is a recipient of the NSF CAREER Award, the International Fulbright Science and Technology Award, the Omega Alpha Association's Exemplary Dissertation Award, and several best paper awards. Dr. Salado holds a BS/MS in electrical and computer engineering from the Polytechnic University of Valencia, a MS in project management and a MS in electronics engineering from the Polytechnic University of Catalonia, the SpaceTech MEng in space systems engineering from the Technical University of Delft, and a PhD in systems engineering from the Stevens Institute of Technology. Alejandro is a member of INCOSE and a senior member of IEEE and AIAA.

INCOSE

INCOSE Certification

See why the top companies are **seeking out INCOSE Certified Systems Engineering Professionals.**

Are you ready to advance your career in systems engineering? Then look into INCOSE certification and set yourself apart. We offer three levels of certification for professionals who are ready to take charge of their career success.

Apply for INCOSE Certification Today!

Visit www.incose.org or call 800.366.1164

ASSOCIATE INCOSE ASEP SYSTEMS ENGINEERING PROFESSIONAL
CERTIFIED INCOSE CSEP SYSTEMS ENGINEERING PROFESSIONAL
EXPERT INCOSE ESEP SYSTEMS ENGINEERING PROFESSIONAL

MARCH 2023
VOLUME 26/ ISSUE 1

Formalizing the Representativeness of Verification Models using Morphisms

Paul Wach, paulw86@vt.edu; **Peter Beling,** beling@vt.edu; and **Alejandro Salado,** alejandrosalado@arizona.edu
Copyright ©2022 by Paul Wach, Peter Beling, and Alejandro Salado. Permission granted to INCOSE to publish and use.

■ **ABSTRACT**
With the increasing complexity that is being introduced to engineered systems, the literature suggests that verification may benefit from theoretical foundations. In practice and in teaching of system engineering (SE), we typically define a verification model (simulation, test article, etc.) under the assumption that the model is a valid representation of the system design. Is this assumption always true? In this article, we explore the use of system theoretic morphisms to mathematically characterize the validity of representativeness between verification models and corresponding system design.

INTRODUCTION

The last decade has seen an increase in calls for theoretical foundations of systems enginering (Triantis and Collopy 2014; Collopy 2015b; Collopy 2015a; INCOSE ; Rousseau and Calvo-Amodio 2019; Rousseau 2020, 2019; Hammami and Edmonson 2015; Schindel 2019). One major perspective is that a "unifying" theory of systems engineering may not be feasible; however, a need for theoretical foundations remains unresolved (Collopy 2015b). The International Council on Systems Engineering (INCOSE) has defined a desire for systems enginering to be grounded in rigorous mathematics (INCOSE 2014). These rigorous foundations are being explored by various INCOSE sponsored initiatives and groups such as the future of systems engineering (FuSE) initiative (INCOSE 2020). In other research, while some have called for renewed interest in general systems theory (Schindel 2019; Rousseau 2020), others have suggested that mathematical approaches can be used to "disambiguate systems engineering" (Hammami and Edmonson 2015). In this context, verification has been deemed fundamentally broken for addressing the increasing complexity of modern systems (Collopy 2015a). In the article, the author described several areas of systems engineering in need of theoretical foundations, such as the need for systems engineering theory to characterize abstraction and elaboration, which is an aspect of the proposed path related to verification that we present in this article.

To account for resource constraints and reduction of technical risks, systems engineering relies on verification models that are perceived as representative of the design. Examples of these verification models include mass mock-ups, development models, breadboards, integration models, structural models, thermal models, engineering models, qualification or certification models, and operational models, among others (Larson et al. 2009). A common belief in systems engineering is that the verification models do not need to fully represent the design; and instead, only need to represent the design "as far as required for test purposes" (Larson et al. 2009). For example, a breadboard may only account for selected functions of the design.

From experience of the authors in practice, a subject matter expert determines the validity of a model used for verification regarding the system design. As an example, we generally assume that a verification model in the form of a mass mock-up may be a valid representation of a system design with respect to its mass. On the other end of the spectrum, we generally assume that a fully functional verification model physically produced on a precise basis of a system design to be a valid representation of the system design. While the assumption of validity of representation of a verification model at these two levels of abstraction is likely to hold, the assumption may not hold for every verification model. This becomes a particular concern when considering that the validity (that is, representativity) of verification models that partially represent the system design is left to the qualitative assessment of one or more engineers.

To establish the characterization of the validity of a verification model to its corresponding system design, we propose the use of systems theoretic morphisms. In exploration of this research thrust, we have found many morphisms that may have applicability within the context of verification. This article serves to provide insight into those morphisms and how they may be used to provide theoretical underpinning to SE.

The remainder of this article is as follows. We provide a literature review on research

that may provide some comparison to our method. This is followed by background to the system theoretic context to which we will provide in detail later in the article. Then we provide a characterization of the morphisms and the inference that the morphisms are expected to enable toward enhancement of verification. This is followed by a discussion and conclusion.

LITERATURE REVIEW

The use of morphisms in the context of systems engineering tasks is scarce; in fact, many of the references to morphisms in the literature provide only vague sense of meaning. For example, references to "self-morphing systems" (Ring 2007) and "morphable architecture" (Ring 2001) are discussed with minimal insights into meaning. The concept of homomorphism seems to have been applied to categorically map processes of requirements definition used throughout the literature (White, Lacy, and O'Hair 1996). In another article, there are brief and abstract mentions of homomorphism to transform observation to possible representations of nature (Ferris 2009).

In Martin (2004), there is reference to "graph morphisms", "concept morphisms", homomorphisms, and "infomorphism" within a figure of the article; however, there is minimal elaboration as to the meaning of the morphisms and no use within the context of representativeness of a verification models to corresponding system design such as we present in this article.

The term homeomorphism has been used in reference to topological mapping (Carl and Hofmeister 2004); and although we do not expressly use homeomorphism, there is indication of its relevance to our research. Specifically, George Friedman, the creator of constraint theory, has referred to homomorphism, seemingly in the same context as the use of homeomorphism, as a means to map the topologies of math models to their corresponding computer aids (Friedman 2007). Indeed, we view the relation to constraint theory and topology to be a complementary factor to our research and evidence that morphisms provide a means to characterizing representativeness of verification models to corresponding system design.

Some articles reference to or build upon the research of A. Wayne Wymore, whose research leverages morphisms as a core concept. In one article, the word "homomorphic" is used in reference to understand categories of systems (Ring 2007). The article is an output from an INCOSE Intelligent Enterprises Working Group to which Wymore was a contributor. Another article contains a reference to Wymore's research and the use of homomorphism to characterize the relationship between a functional architecture and a corresponding physical architecture (Lykins 1997). Indeed, in Shell (1999 and 2001), the article makes the case for leveraging Wymore's research to enable a science-based approach to engineer complex systems, a statement which our research agrees with.

Furthermore, Wymore used homomorphism to mathematically characterize the preservation of equivalence between a software system design, its corresponding hardware system design, and corresponding functional system design (Wymore 1993). Each elaboration and abstraction of the overall system design are mathematically characterized relative to one-another. To expand on the art of the possible, the implication here is that homomorphism is applicable to aiding in our understanding of modern complex cyber-physical systems (CPS). Although we do not address CPS in this article, our research agenda is slated for study of CPS with the means provided in this article.

Our research is based on Wymore's theoretical contributions; however, it should not be considered one-and-the-same. Wymore largely limited his research to homomorphism between the functional and buildable/physical designs/architectures. Based on the homomorphic mapping between the two, an implementable design is formed and verified. An assumption here is that testing (verification) is only conducted on a complete design, whereas in actual practice, models that may not be a complete system design are used for verification. Rather, a verification model may be created to partially represent the system design. Our research builds on Wymore's use of homomorphism and suggests that all verification models (for example, mass mock-up, partial-functional, full functional, simulation, physical, etc.) should be characterized based on a morphic relationship to the corresponding system design for which confidence as to adherence to requirements is intended to be inferred.

We aim to leverage the concept of a homomorphism and add other morphisms to the toolbox from which systems engineering can draw upon. The use of homomorphism has been recently applied in a verification context from the literature to define a theory for capturing verification strategies (Salado and Kannan 2018). As an example, the authors suggest that a simulation [verification] model is homomorphic to the system design and can thus be used to establish confidence that the system design adheres to its system requirements. We agree that a simulation [verification] model should be morphically characterized as to representativeness to the system design. However, the morphic characterization may not be limited to homomorphism and should not be heuristically assumed.

THEORETICAL FRAMEWORK

At the core for the research presented in this article is *General Systems Theory* (GST) (von Bertalanffy 1969). The concepts of GST were characterized in the context of systems engineering by A. Wayne Wymore originally in his *Mathematical Theory of Systems Engineering* (Wymore 1967). Wymore refined his systems theory of systems engineering to define a mathematical approach to (and coining of the term (Bjorkman, Sarkani, and Mazzuchi 2013)) MBSE, which Wymore referred to as the Tricotyledon Theory of System Design (T3SD) (Wymore 1993). Despite its existence for several decades, the T3SD remains largely unexplored by the systems engineering community with some referring to Wymore's research as mathematically dense such that only a

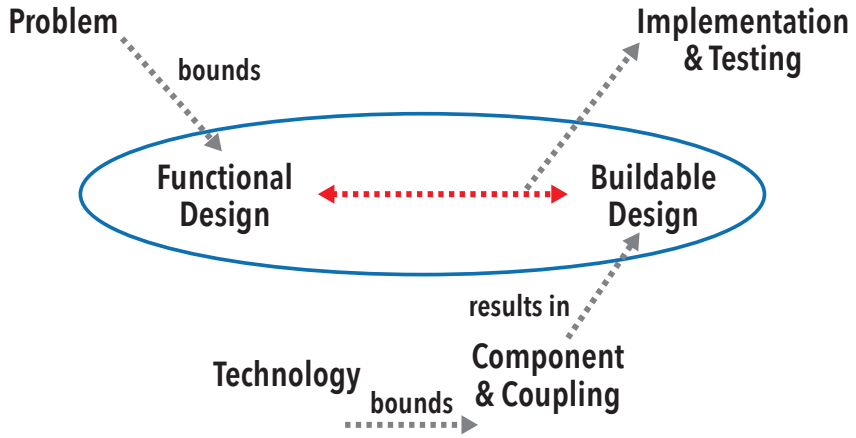


Figure 1. Abstract representation of Wymore's T3SD and use of a morphism (red) to mathematically characterize the preservation of equivalence between the more abstract functional system design and more elaborate buildable system design

mathematician may comprehend (Mabrok and Ryan 2017). Our research thrust has extracted elements of Wymorian systems theory that we believe to be relevant to modern systems engineering.

The concept of homomorphism is a key to establishing understanding of Wymore's mathematically dense T3SD. To abstract the importance of homomorphism in the context of T3SD and the research body presented in this article, refer to Figure 1. The figure can be read as follows. There are three spaces of system designs. The first one, the space of functional system designs results from establishing a problem space (that is, system requirements). The second one, the space of buildable system designs, results from setting constraints on the available technology to build the system. Given that the buildable system design is established such that the functional system design is a homomorphic image of the buildable system design, the system is said to be an implementable system design and is then used as a basis for verification.

We adopt the definition of morphism from systems theory as defined in (Zeigler, Muzy, and Kofman 2019), which defines a morphism as a claim relating the equivalence of a pair of artifacts. In the case of this article, the pair of artifacts are a system design and a verification model. Essentially, the morphism mathematically characterizes the preservation of equivalence between a system design and a verification model.

Note, we intentionally do not provide the mathematical notation as to not distract from the concepts. Our future articles will include mathematical context, to include metrics as to the validity of the representativeness of the verification models to a corresponding system design.

CONCEPTUAL POWER OF MORPHIC VERIFICATION MODELS

In this section, we have selected morphisms that we have discovered from the literature and provide insights into our conceptual understanding of use toward inference for verification.

Homomorphism. The concept of a homomorphism is used to characterize the preservation of structure and behavior between a potentially more abstract model and a potentially more elaborate model. A typical homomorphism may characterize a many-to-one relationship; however, given a relationship that is one-to-one, the characterization is said to be a special kind of homomorphism, referred to as isomorphism. We use Figure 2 to demonstrate the concepts of homomorphism.

We use Figure 2, from which we characterize a homomorphism between Z1, Z2, and Z3. First, we must clarify that Z2 is

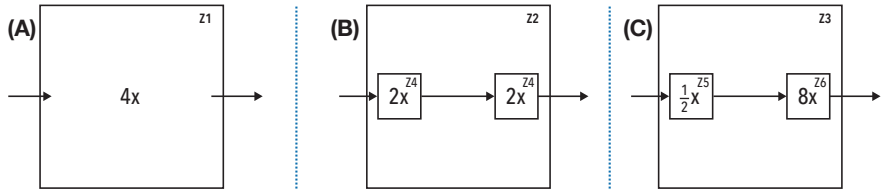


Figure 2. Functional examples used to discuss the concepts of homomorphism

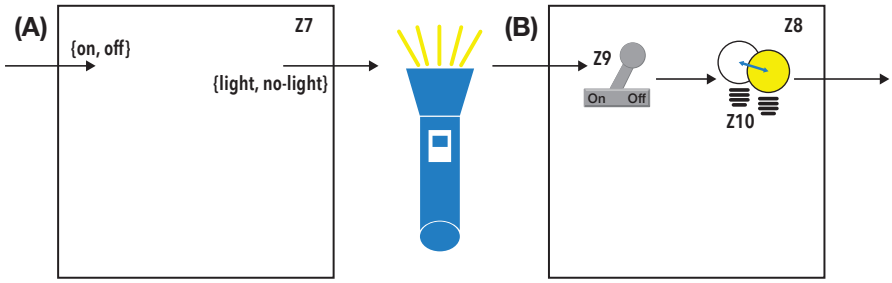


Figure 3. Practical examples used to discuss the concepts of homomorphism

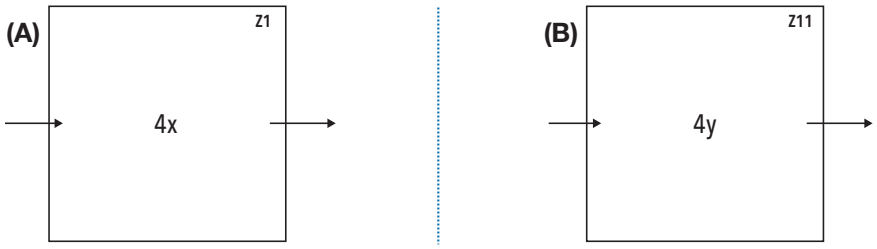


Figure 4. Functional examples used to discuss the concepts isomorphism

the resultant of the coupling of Z4 with Z4 and that Z3 is the resultant of the coupling of Z5 and Z6. In this case, imagine that Z1 is the high-level functional representation to which detailed architecture and system design must adhere to. Z2 and Z3 present alternative functional system designs that are equivalent to Z1. We can prove this by using a homomorphism to map Z2 to Z1 and another homomorphism to map Z3 to Z1.

An alternative perspective is to view Z2 and Z3 as detailed designs and Z1 as an abstract verification model. Similar to discussion in the previous paragraph, in Z2, two of the same components (Z4) are coupled to form the resultant system. We compare Z2 to Z1 to determine proof that, at the system-level, each has equivalent structure and behavior. This proof is provided by homomorphism, which, when established, suggests that we have quantitively determined the existence of a valid abstraction from Z2 to Z1, and understand the limitations to use of the abstract verification model represented in Z1.

We now shift to the practical example of the flashlight represented in Figure 3. Like our discussion in the previous paragraphs, the two components Z9 and Z10 are coupled to form the resultant system model Z8. Homomorphism can be used to characterize the relationship of Z8 to the more abstract system Z7. The indication here may be in that Z8 is a verification model that is more elaborate than the current design to which we wish to test our assumptions for potential future design. As discussed previously, we could also view Z7 as the abstract verification model representing the more elaborate system design of Z8.

Isomorphism. While a typical homomorphism may characterize a many-to-one relationship, given a relationship that is one-to-one, the characterization is said to be a special kind of homomorphism, referred to as isomorphism. We use Figure 4 to demonstrate the concepts of isomorphism.

In the example above, the two functions have the same structure and behavior. Therefore, we can say that they are mathematically equivalent even though the variables may have changed. In the higher complexity of engineering practice, we can leverage well known isomorphisms such as the one between a mechanical mass-spring compared to an electrical circuit (Takahashi 2021), which behave proportionally. In this way, one can use a certain mass-spring system to verify properties of an isomorphic electrical circuit and vice versa.

Coupling Morphisms. There are many aspects of systems engineering to which

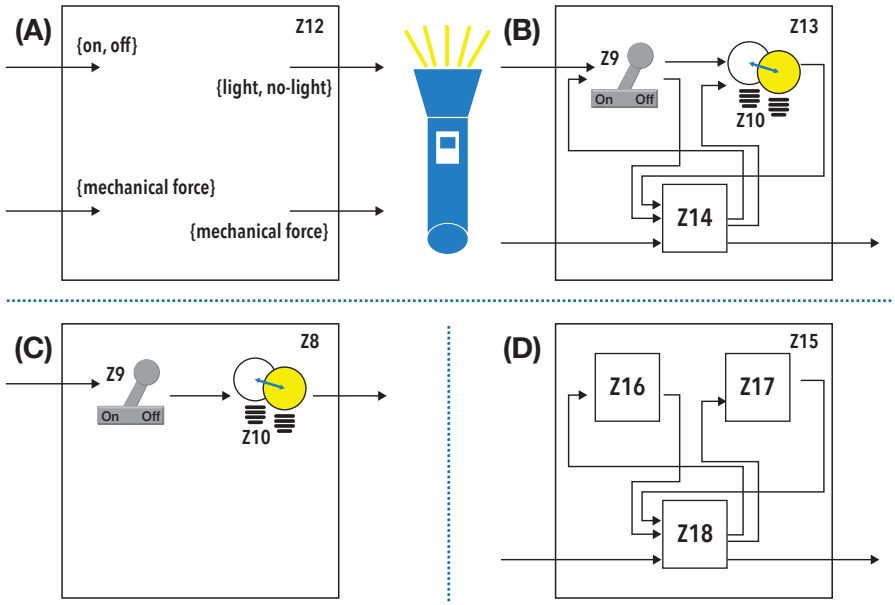


Figure 5. Practical examples used to discuss the concepts of coupling morphism

an understanding of system coupling is important. We can leverage coupling morphisms to determine abstraction of coupling structure and its impact on resulting behavior. To provide insights into coupling morphisms, we show Figure 5.

Consider that Z12 provides a black-box system model showing that the system transforms on/off inputs into the presence of light or no light, as well as a transformation of mechanical forces. Consider that Z13 captures the system design, which consists of three components: the on/off switch (Z9), a lightbulb (Z10), and case (Z15) through which all mechanical forces are transferred. The connections indicate the exchanges between the different components and between some components and the exterior of the system. All functionality is captured in Z13.

Now consider system models Z8 and Z15, which are simplifications of Z13. Particularly, Z8 uses a coupling morphism to remove detail from Z13 (i.e., abstract Z13) to only account for the on/off to light functionality. System model Z15 also uses a coupling morphism, but in this case to capture only the transformation of mechanical forces of the system without considering the on/off to light functionality.

Parameter Morphisms. Here we discuss the use of parameter morphism, which is an approximate homomorphism. The concept of parameter morphism enables explicit accounting for and bounding of acceptable error while maintaining a degree of preservation of equivalence. We use the examples of a simple mathematical function and flashlight provided in Figure 6 and Figure 7 to discuss parameter morphisms.

We first consider the functional example provided by Figure 6. In this case, we assume Z1 to be the system design and Z4 to be the verification model. We show Z1 in Figure 6A with inputs of {1, 2, 4}, which the system function transforms into outputs of {4, 8, 16}. We show Z4 in Figure 6A with inputs of {1, 2, 4}, which the system function transforms into outputs of {2, 4, 8}. Using the parameter morphism, we enable explicit mapping between the inputs of Z1, inputs of Z4, and the resulting distinctions in behavior. Furthermore, we can explicitly note that the system function of the verification model Z4 produces an error (deviation) associated with the behavior in comparison to that which is expected by the system function of the system design Z1.

For practical context of parameter morphism, we use the flashlight example in Figure 7. Here we assume Z8 to be the system design, which has component Z9 that accepts on/off input in the form of a toggle switch mechanism and component Z10 that provides an output of yellow light. We assume Z21 to be the verification model, which has component Z19 that accepts on/off input in the form of a rotation mechanism and component Z20 that provides an output of blue light. Using parameter morphism, we can explicitly characterize the preservation of equivalence between the toggle on/off input of Z8 and the rotation on/off input of Z21 as well as between the yellow light of Z8 and the blue light of Z21. In other words, the parameter morphism allows to define equivalence on part of the behavior of the system. In practice, such surrogate verification models may be necessary to reduce risk from the lack of access to the system design. By explicitly characterizing the differences in parameters between the verification model Z21 and the system design Z8, we account for potential deviations in behavior and, therefore, we should adjust our confidence, in the system design adherence to system requirements, accordingly.

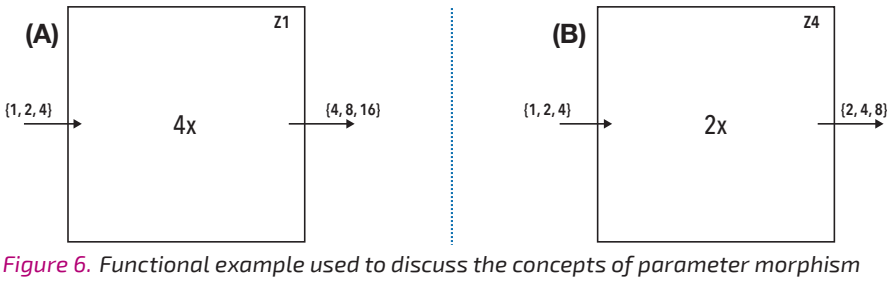


Figure 6. Functional example used to discuss the concepts of parameter morphism

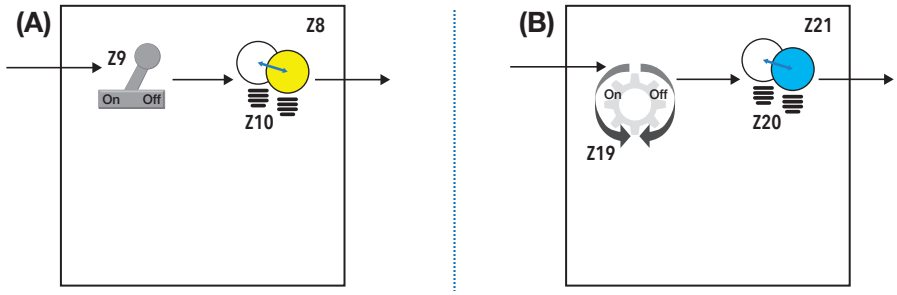


Figure 7. Practical example used to discuss the concepts of parameter morphism

DISCUSSION

Our search of the literature suggests that minimal use of morphisms exist within systems engioneeirng nor do the software tools exist for implementation, especially within the context we presented here of characterization of verification models. This

suggests a research gap and novelty to the approach that we present in this article.

The basic idea of a morphism within the context of systems theory, and therefore systems engineering, is a mathematical characterization of the preservations of equivalence of structural and behavioral similarities (or dissimilarities) (Zeigler, Muzy, and Kofman 2019). The use of the word structure is important to the future implementation of morphisms in practice. Essentially, a morphism is heavily dependent on the mathematical construct used to model the system being engineered. In the case of Wymore's T3SD (Wymore 1993), he used a mathematical construct based on the Moore state machine. This enabled homomorphic comparison between the more abstract functional system design and the more elaborate buildable system design.

We used morphisms in a different article to create a metamodel of verification artifacts (Wach, Beling, and Salado 2022). Specifically, morphisms were leveraged to characterize the preservation of equivalence between system requirements and verification requirements as well as between a system design and a verification model. In doing so, we provide a frame in which we characterize the argument for confidence of system design adherence to system requirements through the morphic preservation provided in the verification model and as observed through the morphic preservation as provided through the verification requirements.

Our research thrust is to bridge the gap between theory and practice. Based on personal experience, we believe that the detailed mathematical descriptions inherent in the theory of morphisms may distract from the necessary conversation we facilitate through our abstract representations provided in this article. As such, both in this article and in Wach, Beling, and Salado (2022), we have intentionally not provided the mathematical descriptions of the morphisms. Future research articles are expected to provide depth on the mathematical descriptions necessary to provide a rigorous basis for morphisms.

Model-based systems engineering (MBSE) has emerged over the last decade as a paradigm to the modern study and practice of systems engineering. From review of the literature and experience of the authors, MBSE is serving as a core mechanism to enable the digital transformation. Among the desired outcomes of the digital transformation is improved verification (Zimmerman, Gilbert, and Salvatore 2019), which MBSE may be able to enable. However, MBSE currently lacks theoretical foundations, which may hinder its ability to deliver the desired outcomes of the digital transformation. The lack of theoretical foundations of MBSE comes despite the INCOSE stated desire to have systems engineering "grounded in a more rigorous foundation of mathematics" (INCOSE 2014) and "based on [codified and] accepted theoretical foundations" (INCOSE 2022). In engineering domains outside of systems engineering, theoretical definition of the practice came first after which software was defined to implement the theoretical algorithms for practical modeling. MBSE currently remains largely qualitatively descriptive, rather than quantitively analytical such as is seen with finite element analysis and computational fluid dynamics. Therefore, the lack of theoretical foundations puts the ability of MBSE to deliver improved verification for the digital transformation into question, which we believe will be resolved in part due to contributions from our research thrust.

CONCLUSION

We have presented an approach to mathematically characterize verification models (for example, simulation, physical articles, etc.) relative to the system design to which models correspond. In the practice of systems engineering, the characterization of a verification model relative to design is something we may take for granted under an assumption of heuristic validity. In doing so, we are creating a level of confidence in adherence of the system design to system requirements, which may not hold given agreement based on the mathematical characterization that we suggest for employment in systems engineering practice. This research advances the theoretical foundations of systems engineering, and subsequently the theoretical foundations of our modeling practice (that is, MBSE). Furthermore, the research we present in this article suggests that systems theoretic morphisms can be leveraged to mathematically characterize verification model relative to the system design, which we believe will correspond to enhanced verification. ■

REFERENCES

- Bjorkman, Eileen A., Shahram Sarkani, and Thomas A. Mazzuchi. 2013. 'Using model-based systems engineering as a framework for improving test and evaluation activities', Systems Engineering, 16: 346-62.
- Carl, Joseph, and Janet Hofmeister. 2004. "7.6. 2 Object-Oriented and Structured Analyses Are Homeomorphic." In INCOSE International Symposium, 1581-90. Wiley Online Library.
- Collopy, P. D. 2015a. "Systems engineering theory: What needs to be done." In 2015 Annual IEEE Systems Conference (SysCon) Proceedings, 536-41.
- Collopy, Paul D. 2015b. "Report on the science of systems engineering workshop." In 53rd AIAA aerospace sciences meeting, 1865.
- Ferris, Timothy LJ. 2009. "9.1. 3 development of a framework of research topics in systems engineering." In INCOSE International Symposium, 1378-90. Wiley Online Library.
- Friedman, George. 2007. "Biologically Inspired Systems Concepts–A personal history." In INCOSE International Symposium, 1909-15. Wiley Online Library.
- Hammami, O., and W. Edmonson. 2015. "THEFOSE – Theoretical Foundations of System Engineering: A first feedback." In 2015 IEEE International Symposium on Systems Engineering (ISSE), 370-74.
- INCOSE. 'Future of Systems Engineering (FuSE)', Accessed June 9. https://www.incose.org/about-systems-engineering/fuse.
- ———. 2014. 'INCOSE System Egineering Vision 2025 July, 2014', Accessed November 12. https://www.incose.org/docs/default-source/aboutse/se-vision2025.pdf?sfvrsn=4&sfvrsn=4.
- ———. 2020. 'Systems Processes & Pathologies', Accessed 7 July.
- ———. 2022. 'Systems Engineering Vision 2035', Accessed Mar 16. https://www.incose.org/about-systems-engineering/se-vision-2035.
- Larson, W J, D. Kirkpatrick, J. J. Sellers, D Thomas, and D. Verma. 2009. Applied Space Systems Engineering (The McGraw-Hill Companies, Inc.).
- Lykins, Howard. 1997. "A Framework for Research Into Model-Driven System Design." In INCOSE International Symposium, 220-27. Wiley Online Library.
- Mabrok, M, and M Ryan. 2017. 'Category Theory as a Formal Mathematical Foundation for Model-Based Systems Engineering', Appl. Math. Inf. Sci., 11: 43-51.
- Martin, Richard A. 2004. 'Enabling Intelligence with Ontology,' *INSIGHT*, 6: 12-15.
- Ring, J. 2007. 'About Intelligent Enterprises: A Collection of Knowledge Claims', INCOSE International Symposium, 17: 1964-2079.

- Ring, Jack. 2001. "2.6. 3 The Next Venue for Systems Engineering." In INCOSE International Symposium, 900-07. Wiley Online Library.
- Rousseau, David. 2019. 'A vision for advancing systems science as a foundation for the systems engineering and systems practice of the future,' Systems Research and Behavioral Science, 36: 621-34.
- ———. 2020. 'The Theoretical Foundation(s) for Systems Engineering? Response to Yearworth,' Systems Research and Behavioral Science, 37: 188-91.
- Rousseau, David, and Javier Calvo-Amodio. 2019. 'Systems Principles, Systems Science, and the Future of Systems Engineering', *INSIGHT*, 22: 13-15.
- Salado, Alejandro, and Hanumanthrao Kannan. 2018. 'A mathematical model of verification strategies', Systems Engineering, 21: 593-608.
- Schindel, William D. 2019. 'Identifying phenomenological foundations of systems engineering and systems science', Systems Research and Behavioral Science, 36: 635-47.
- Shell, AD. 1999. "1 Function Based Design Of Complex, Complicated Systems." In INCOSE International Symposium, 224-31. Wiley Online Library.
- Shell, T. 2001. 'System function implementation and behavioral modeling: A systems theoretic approach,' Systems Engineering, 4: 58-75.
- Takahashi, Shingo. 2021. 'Systems Modeling.' in Gary S. Metcalf, Kyoichi Kijima and Hiroshi Deguchi (eds.), Handbook of Systems Sciences (Springer Singapore: Singapore).
- Triantis, Konstantinos P, and Paul D Collopy. 2014. "A comprehensive basis for systems engineering theory." In 2014 IEEE International Systems Conference Proceedings, 97102. IEEE.
- von Bertalanffy, L. 1969. General Systems Theory—Foundations, Development, Applications (George Braziller, Inc.: New York, NY, USA).
- Wach, P, P Beling, and A Salado. 2022. "Systems Theoretic Metamodel of Verification Artifacts." In CSER. Norwegian Univeristy of Science and Technology, Norway.
- White, Michelle M, James A Lacy, and Edgar A O'Hair. 1996. "Refinement of the Requirements Definition (RD) Concept in System Development: Development of the RD Areas." In INCOSE International Symposium, 762-70. Wiley Online Library.
- Wymore, A. Wayne. 1967. A Mathematical Theory of Systems Engineering - the Elements (John Wiley and Sons Inc.: New York, NY, USA).
- ———. 1993. Model-Based Systems Engineering (CRC Press LLC: 2000 NW Corporate Blvd., Boca Raton, FL, USA 33431).
- Zeigler, Bernard P., A Muzy, and E Kofman. 2019. Theory of Modeling and Simulation: Discrete Event & Interative System Computational Foundations (Elsevier Inc: London Wall, London, UK).
- Zimmerman, Phil, Tracee Gilbert, and Frank Salvatore. 2019. 'Digital engineering transformation across the Department of Defense', The Journal of Defense Modeling and Simulation, 16: 325-38.

ABOUT THE AUTHORS

Paul Wach received a BS in biomedical engineering from Georgia Tech, MS in mechanical engineering from the University of South Carolina, and PhD in systems engineering at Virginia Tech. His research interests include theoretical foundations of systems engineering, digital transformation, and artificial intelligence. Paul is a member of the Intelligent Systems Laboratory with the Virginia Tech National Security Institute. He is the president and founder of the Virginia Tech student division of INCOSE. Paul has part-time association with The Aerospace Corporation and is leading enterprise digital engineering transformation. His prior work experience is with the Department of Energy, two National Laboratories, and the medical industry.

Peter Beling is a professor in the Grado Department of Industrial and Systems Engineering and associate director of the Intelligent Systems Laboratory at the Virginia Tech National Security Institute. Dr. Beling's research interests lie at the intersections of systems engineering and artificial intelligence (AI) and include AI adoption, reinforcement learning, transfer learning, and digital engineering. His research has found application in a variety of domains, including mission engineering, cyber resilience of cyber-physical systems, prognostics and health management, and smart manufacturing. He received his PhD in operations research from the University of California at Berkeley.

Alejandro Salado has over 15 years of experience as a systems engineer, consultant, researcher, and instructor. He is currently an associate professor of systems engineering with the Department of Systems and Industrial Engineering at the University of Arizona. In addition, he provides part-time consulting in areas related to enterprise transformation, cultural change of technical teams, systems engineering, and engineering strategy. Alejandro conducts research in problem formulation, design of verification and validation strategies, model-based systems engineering, and engineering education. Before joining academia, he held positions as systems engineer, chief architect, and chief systems engineer in manned and unmanned space systems of up to $1B in development cost. He has published over 100 technical papers, and his research has received federal funding from the National Science Foundation (NSF), the Naval Surface Warfare Command (NSWC), the Naval Air System Command (NAVAIR), and the Office of Naval Research (ONR), among others. He is a recipient of the NSF CAREER Award, the International Fulbright Science and Technology Award, the Omega Alpha Association's Exemplary Dissertation Award, and several best paper awards. Dr. Salado holds a BS/MS in electrical and computer engineering from the Polytechnic University of Valencia, a MS in project management and a MS in electronics engineering from the Polytechnic University of Catalonia, the SpaceTech MEng in space systems engineering from the Technical University of Delft, and a PhD in systems engineering from the Stevens Institute of Technology. Alejandro is a member of INCOSE and a senior member of IEEE and AIAA.

MARCH 2023
VOLUME 26/ ISSUE 1

Verification and Validation of SysML Models

Myron Hecht, myron.hecht@aero.org; and **Jaron Chen,** jaron.chen@aero.org
Copyright ©2021 by Myron Hecht and Jaron Chen. Permission granted to INCOSE to publish and use.

■ **ABSTRACT**
Model-based systems engineering depends on correct models. However, thus far, relatively little attention has been paid to ensuring their correctness. This paper describes a methodology for performing verification and validation on models written in SysML. The methodology relies on a catalog of candidate requirements that can be tailored for a specific project. Both manual and automated methods are used to verify and validate these requirements. Manual methods are necessary where knowledge of the domain and other extrinsic characteristics are necessary. Automated methods can be used where the requirements cover the use of SysML. Examples from a public domain SysML model of a satellite are presented to demonstrate application of automated requirements verification.

INTRODUCTION

Model-based systems engineering (MBSE) will not succeed without correct and complete system engineering models. Incorrect models can cause wrong design decisions or integration failures because of errors in representation of the architecture, design, or behavior; and many other reasons. Therefore, verification and validation (V&V) of models is essential to successful MBSE projects. However, system engineering model V&V is not widely practiced. In a survey conducted by the System Engineering Research Consortium (SERC), more than 60% of government respondents disagreed with the statement "Our organization has defined processes and tools for V&V of models at appropriate levels and program phases" (McDermott, et al. 2020). Less than 3% strongly agreed with it. The difficulties of verifying and validating system engineering models include:

1. *No definition of what a "correct" model is*: Without a set of requirements that define correctness, it is impossible to verify a model.
2. *Failure to define model user needs*: Without a definition of user needs, it is not possible to validate a model.
3. *Difficulty of finding syntax errors*: Most syntactic, value property, and data typing errors are difficult to find through manual inspection.
4. *Inadequate documentation*: Internal and external documentation of models is necessary to define the intent of the model and to minimize spurious assessments of model validity.

This paper describes an approach that addresses these difficulties. The next two subsections de- scribe previous work and the conceptual framework of our approach. The next section describes model requirements formulation, and the final section describes of verification and validation methods for SysML models.

There are many definitions of "verifi–cation" and "validation." IEC/ISO/IEEE 24765 "Software and System Engineering Vocabulary" lists six definitions for verification and five for validation. There is a separate vocabulary entry for "verification and validation" which combines them. The meaning of both terms, either separately or collectively, is a set of activities intended to assess a model for correctness and suitability for the user. In this paper, "verification" means assessment of conformance to requirements and "validation" means assessment of suitability to user needs. For the methodology described in this paper, user needs are stated as requirements.

PREVIOUS WORK

A recent literature survey of 579 SysML publications between 2005 and 2017 identified few that addressed verification and validation of SysML models (Wolny, et al. 2020). What has been published can be placed in two major categories: model transformation and model inspection.

Model transformation: Model transformation involves converting SysML models into other formalisms that could then subsequently be analyzed for conformance to specialized properties that could be analyzed with those formalisms. Ahmad, Dragomir, et al. proved invariant properties by transforming an XMI export of a SysML to an executable UML/SysML profile model (OMEGA2) of a diabetes patient and her refrigerator (Ahmad, et al. Jul 2013). The model consisted of two internal block diagrams and a state machine diagram consisting of two 2 states. Jarraya and Debbabi demonstrated a method of probabilistic verification of SysML activity diagrams by transforming them into the PRISM

(probabilistic symbolic model checker) on a banking operation consisting of 10 actions (Jarraya and Debbabi 2012). Rahim, Hammad, and Boukala-Ioualalen described an approach to verification by transforming activity diagrams into the Petri net markup language (PNML) and demonstrated it on a vending machine activity diagram consisting of 9 actions (Rahim, et al. 2015). Our work does not use model transformations because they have not been successfully applied to large system models.

Model inspection: Model inspection uses checklists or rules to assess conformance. The inspections can be manual, automated, or both. Inspection cannot formally prove correctness but provides a "good enough" basis for acceptance. Pettit formulated a list of rules for inspection of UML models (Pettit 2003). Douglas (2017) defined a rule set for use with UML and SysML specifically tailored for the Rhapsody tool set. Baduel, Chami, Burel, and Ober described a corporate-specific approaching combining the use of domain experts to check compliance with external "real-world" attributes against a set of criteria and of OCL to verify conformance with syntax rules and for a product line of train control systems (Baduel, et al. 2018). Vinarcik and Jukovic have developed a digital engineering profile containing hundreds of specific automated verification rules that check for violations of their own standards and have placed them in the public domain (Vinarcik and Jugovich 2020). The automated verification rules described in this paper adapted approximately 60 of these rules. SysML modeling tools such as the one used in this work (3DS/Nomagic 2021) have validation rules based on the SysML language specification that check for basic violations (for example, connections made to incompatible ports). The limitation of these inspection-based approaches is that they are generic. Thus, application of these rule sets will not ensure that a model meets the needs and objectives of a specific project. To verify and validate a particular model, it is necessary to evaluate against the specific objectives and requirements of a particular modeling project including its sponsors, users, and developing organizations.

Conceptual Framework

Figure 1 shows a metamodel that defines the context and our approach to SysML model verification and validation. A program *sponsor* is responsible for a project has *needs for a model*. As a result, it sponsors funds a *SysML Model*. The model represents a *system*, which is specified by its own *system requirements*. The sponsor needs are translated into *model requirements*. It is important to note that *system requirements* and *model requirements* are different and are satisfied by different items, that is, the model and the system respectively.

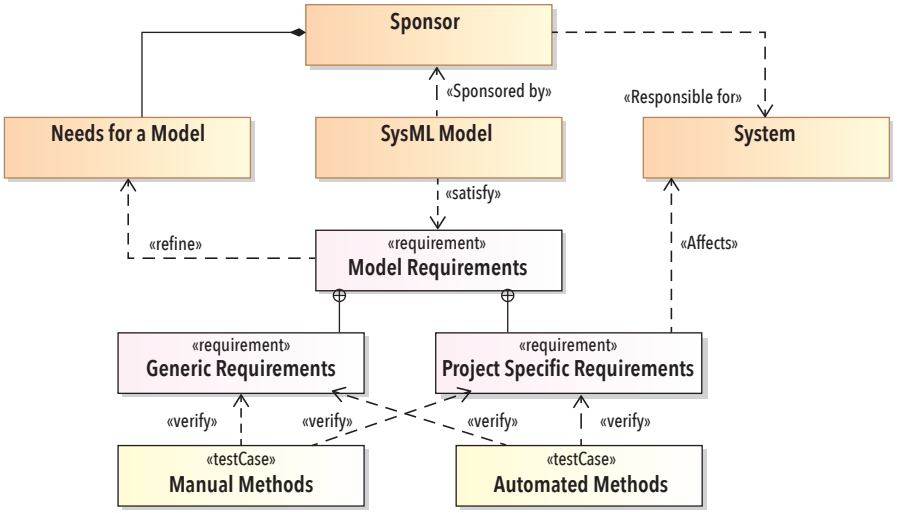


Figure 1. Verification and validation (V&V) metamodel

Model requirements are either generic (that is, project independent) or project specific. Examples of generic requirements include specification of modeling language, tools, and profiles; sponsoring organization unique requirements (for example, security and protection of intellectual property); and organizational modeling conventions and rules. Project specific requirements include the scope and level of detail of what is to be modeled, specific artifacts to support program management and design reviews, naming conventions, and rules related to use of model elements that are specific to the project. Both manual and automated methods can be used for V&V of both generic and system specific requirements.

MODEL REQUIREMENTS

Formulation of SysML model requirements is a difficult task, but a catalog of sample requirements, or checklist can simplify the work and facilitate a more complete set of requirements. This section describes such a catalog whose organization is shown in Figure 2 (Hecht, et al. 2021). The major categories are overarching general requirements, requirements, structure (including parametric diagrams), behavior, model data, and non-functional attributes (that cut across the other categories). More than 300 sample requirements have been defined within this framework. These model requirements represent lessons learned from a substantial amount of modeling and model review experience, but a metric of completeness is not yet possible. This section provides an overview. A full listing is available from the authors.

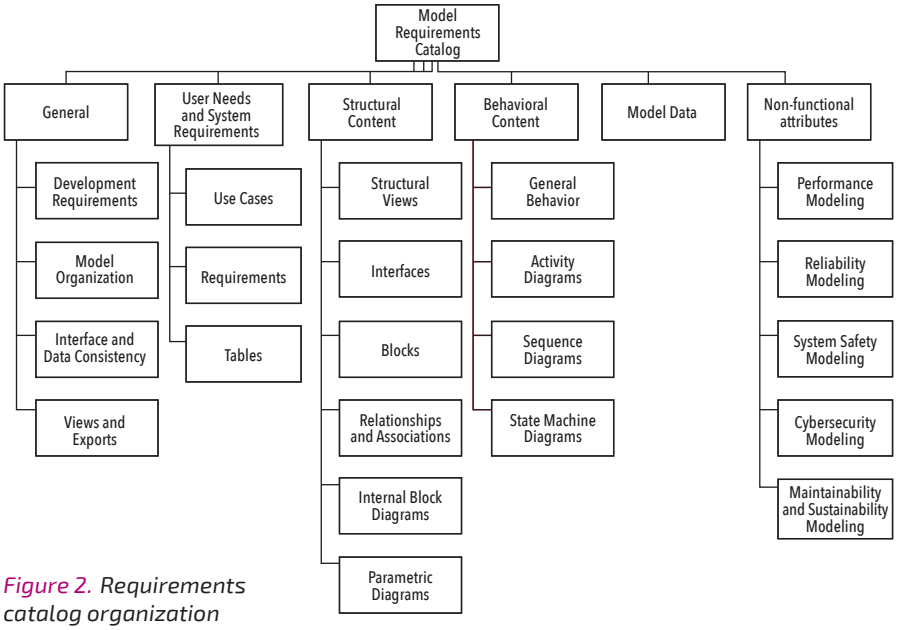


Figure 2. Requirements catalog organization

General: General Requirements cover the general user and model sponsor's needs rather than a specific project model. Lower-level headings include:

- *Development Requirements:* related to model language, stereotypes, tool selection, documentation of model elements, conformance to style guidelines, and use of prescribed modeling patterns.
- *Model Organization:* requirements on consistent and logical model structure and organization, use of model navigation aids, and labeling of all packages.
- *Interface and Data Consistency:* requirements for interfacing and provision or ingesting of data to at specified interfaces to other components of the digital engineering ecosystem and consistency with model libraries of the sponsoring organization.
- *Views and Exports:* requirements specifying the information that the model provides to non-modeling users and include views into the model and exports. Such exports can include tables, "dashboards", and even complete documents such as system and subsystem design documents, verification and validation plans, test procedures, test reports, and technical orders.

User needs and system requirements: These requirements concern modeling the needs of the organization and users of the system under development (not the model) as would be stated in the Concept Definition Document and the System Requirements Document. Topics covered include:

- *Use Cases:* SysML model requirements that Use Cases capture mission objectives and system operations, trace use cases to requirements and to verification methods, standards for description and documentation for use cases and actors, and coverage of exception conditions.
- *System Requirements:* requirements on system requirements, for example that the model shall include of all system requirements in the technical baseline, that all requirements shall include a rationale, and that all requirements be related either to other requirements through contain, derive or define relationships or must themselves have satisfy and verify relationships.
- *System Requirements Diagrams and Tables:* requirements on the model to show requirements relationships – especially satisfy and verify relationships.

Structural content requirements: These requirements affect SysML blocks, block definition, internal block, and parametric diagrams. Subheadings include:

- *Structural Views:* Requirements to capture context, logical architecture, functional architecture, and physical architecture, and to allocate model elements among all three architectures to enable traceability.
- *Interfaces:* Requirements that all block connections must be made through defined interfaces, that interfaces on each side of connection must be compatible, that conventional interface control documents (ICDs) be linked to interface blocks that represent them, that all connections have defined flows, and that operations and receptions be used when behaviors are part of the interface.
- *Blocks:* Requirements on traceability between SysML blocks and system requirements, block naming conventions, documentation including description of pre- and post-conditions on operations and receptions, allocation of activities or actions to blocks, depiction of allocation of software to hardware, data typing, and others.
- *Relationships and associations:* Requirements for roles and multiplicity on part ends of composition relationships, prohibition against both composition and inheritance to a single block, prohibition against cyclic inheritance, justification for any block with no relationship to other elements, and others.
- *Internal block diagrams:* Requirements for all blocks to be connected through ports, flows to be shown on all connections, multiplicities on both ends of the connectors, typing of flow properties, and others.
- *Parametric diagrams:* Requirements for limits on the number of blocks and connections to retain readability, verification of numerical constraint blocks, visibility of binding connectors, and populated constraint specifications, and others.

Behavioral Content: Requirements in this area cover activity, sequence, and state machine diagrams. Topics covered include:

- *General behavior:* Requirements to model all dynamic behavior in the scope of the model and that all diagrams document the assumptions, design decisions, preconditions, and postconditions (all requiring manual verification).
- *Activity diagrams:* Requirements on the correctness of allocation of actions to blocks (swim lanes), and the inclusion of error handling. Automatically verifiable requirements include: Input pins must have an incoming object flow, output pins must have an outgoing object flow, all control and object flows exiting a decision node must have defined guards, forks and decisions having at least two outgoing flows, joins and merges having at least two incoming flows, and others.
- *Sequence diagrams:* Requirements that sequence diagrams capture behaviors on connections, model operations and receptions between diagrams, and that they capture error handling and recovery; messages on sequence diagrams have signatures assigned (signal or operation); and others.
- *State machine diagrams:* Requirements for state machine such as that that all blocks that change state shall be modeled using state machine diagrams, naming conventions on states and transitions, capturing of signals, events, message flows as transition triggers, representation of error handling and recovery, and others.

Model Data: Requirements for model data including documentation, data types for all attributes and properties, consistent naming in accordance with naming conventions, and a glossary.

Non-functional Requirements: Requirements covering performance, reliability/availability, safety, cybersecurity, and maintainability and sustainment modeling. Architectural design decisions can have a large impact these attributes (particularly if they are overlooked). This section is responsive to IEEE 15288 (ISO/IEC/IEEE, 2015), Appendix F "Architecture modeling", par. F.3.8 "Other model considerations".

- *Performance Modeling:* Requirements for data necessary to analyze and calculate response time and capacity at any architectural level; data necessary to analyze and calculate response time, capacity, ability to simulate at any architectural level; and others.
- *Reliability Modeling:* Requirements for data necessary to analyze and calculate quantitative reliability attributes using analytical techniques such as reliability block diagrams, fault tree analysis, and Markov modeling; data necessary to analyze and reliability and availability at any architectural level; and others.
- *System Safety Modeling:* Requirements that the model allow for the input, storage, and export of data necessary to support all deliverables required under the program contract, data for all safety review and certification authorities whose approval is required, data input, storage, and export for all tasks related to explosive safety standards, production of reports

Table 1. Manual acceptance criteria for model requirements

Category	Acceptance Criteria Examples
Inspection	
General	• Reasonableness of model structure • Effectiveness and correctness of model navigation and dashboards • Correctness of access control assignment to packages and marking of elements • Logical and reasonable naming conventions • Correctness in the application of profiles and modeling patterns • Correctness, completeness, organization, and appearance of model generated documents
Modeling of User Needs and System Requirements	• Clarity, specificity, feasibility, verifiability, and correctness of requirements in the model • Correctness of links between use cases and requirements • Correctness of derive and refine links between requirements and satisfy, refine, and trace links from requirements to other model elements • Correctness of requirements documentation • Completeness and feasibility of use case exception handling use cases, preconditions, postconditions, and assumptions • Correctness and completeness of requirements rationale explanations
Structural Content	• Correctness of requirements documentation • Completeness of the structural models • Adequate level of detail in the structural model • Correctness and completeness of block documentation • Correctness and completeness of allocations between logical, functional, and physical model elements • Correctness and completeness of translation of manually written ICDs into interface blocks
Behavioral Content	• Completeness of behavior modeling • Completeness and correctness of documentation of assumptions, design decisions, pre-conditions, and post-conditions • Correctness of the allocation of activities and actions to blocks • Correctness and completeness of message sequences
Model Data	• All value properties are defined and typed
Non-functional Requirements Modeling	• Correctness and completeness of internal parametric diagrams for quantitative modeling of non-functional requirements • Correctness in the selection of model elements exported to external quantitative modeling tools of non-functional attributes • Correctness and completeness of documents created for approval and certification authorities
Analysis	
Structural Content	• Calculation using an alternative tool of the content of constraint blocks within parametric diagrams. Acceptance criteria are the matching of results within allowed tolerances.
Behavioral Content	• Simulation ("animation") of activity diagrams, sequence diagrams, and state machines. Acceptance criteria are the matching of model results with expected results determined by another method
Non-functional requirements	• For internal models, comparing results of parametric model constraints against alternative calculations. Acceptance criteria are the matching of results within allowed tolerances
Demonstration	
General	• Generation of required model reports that meet requirements (reports themselves would be validated by inspection as indicated in Table 1) • Application of model profiles to provide expected results (for example, reports, tables, views) • Demonstration of model views and navigation aids (model views and navigation aid correctness would be validated by inspection as indicated in Table 1) • Demonstration of acceptable response times
Behavioral Content	• Simulation ("animation") of activity diagrams, sequence diagrams, and state machines and observing that execution occurs and that the model is capable of continuing operation. Acceptance criteria are the matching of model results with expected results validated by analysis

Table 1. Manual acceptance criteria for model requirements (continued)

Category	Acceptance Criteria Examples
Demonstration	
Non-functional requirements	• Demonstration that model data can be exported and executed by external tools and that results of external tools can be imported into the model. • Acceptance criteria are that data interchange results are expected by analysis • Demonstration that internal tools are capable of execution. Acceptance criteria are the matching of model results with expected results validated by analysis
Test	
General	• Constraint blocks in reliability and availability, or performance models provide the same results as independent calculations

acceptably formatted, and display the status of activities required for safety certification approval at any architectural level.

- *Cybersecurity Modeling:* Requirements for the input, storage, and export of data necessary for deliverables for all cybersecurity standards or instructions required under the program contract, all cybersecurity accreditation and certification authorities, reports acceptably formatted for all tasks in all cybersecurity standards or instructions.
- *Maintainability and Sustainability Modeling:* Requirements for the input, storage, and export of data necessary to support all deliverables for all maintainability, sustainability, storage, repair, packaging, shipping, and handling and other activities required for all maintainability, sustainability, storage, repair, packaging, shipping, and handling standards.
- *Cybersecurity Modeling:* Requirements for the input, storage, and export of data necessary for deliverables for all cybersecurity standards or instructions required under the program contract, all cybersecurity accreditation and certification authorities, reports acceptably formatted for all tasks in all cybersecurity standards or instructions.
- *Maintainability and Sustainability Modeling:* Requirements for the input, storage, and export of data necessary to support all deliverables for all maintainability, sustainability, storage, repair, packaging, shipping, and handling and other activities required for all maintainability, sustainability, storage, repair, packaging, shipping, and handling standards.

VERIFICATION AND VALIDATION

This section describes the verification and validation of SysML model requirements. The first subsection summarizes manual verification methods and the second covers automated methods.

Manual Methods

Table 1 summarizes manually evaluated model requirement acceptance criteria grouped by standard verification method (inspection, analysis, demonstration, and test) and the model requirement topic areas shown in Figure 2.

Our experience using manual inspection in SysML models of multiple large projects is that use of checklists derived from the requirements catalog has resulted in significantly faster and more thorough evaluations than with ad hoc inspection. Manual inspections also detect program specific issues (as expressed in the content of the model elements) that are not detectable using automated methods (discussed in the next section). However, there are two disadvantages: (1) manual methods only inspect what is visible in the model, and (2) manual inspection coverage of large models is difficult to assess.

Automated Methods

Automated methods rely on the capabilities of SysML modeling tools to run scripts that query models to check SysML models for conformance with model element, relation, and connection requirements (sometimes called "validation rules"). A total of 127 requirements in the requirements catalog described earlier can be automatically validated. Benefits of automated rules are that (1) they can identify subtle syntax errors far more rapidly and thoroughly than manual inspection, (2) while they cover a minority of the requirements, these scripts cover many more elements and diagrams than manually verified requirements, and (3) verification can be repeated at low cost.

Implementation of automated verification differs across SysML modeling tools. In Cameo Systems Modeler (3DS/Nomagic 2021), the tool used in this work, every rule violation needs to be associated with a model element. The output of the script associated with the rule must be Boolean. If the output is false, the error message will be displayed with the element that violates the associated. An output of true would mean that there is no violation. The scripts are written using the application programming interface in that tool to access model properties and navigate to other elements.

Figure 3 shows a script to check conformance a requirement that in a state machine diagram, there should be only one transition between any two states. As shown in the figure, the scripting language is Jython (Python implemented in the Java Runtime Environment).

The following examples show the results of application of these automated verification rules for use case, requirements, block definition, internal block, and activity diagrams. The SysML model used to demonstrate their application is a hypothetical satellite (Friedenthal and Oster 2017). The model authors created it as public domain example for identifying and evaluating alternative spacecraft architectures, and not as a full industrial

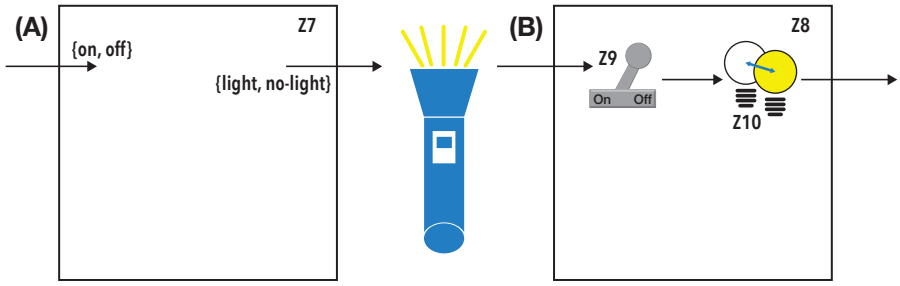


Figure 3. Script for automated verification of a state machine diagram requirement

grade model for representing a system. The model was not intended to be comprehensive, complete, or rigorous. However, it is model is representative of medium sized projects. Because the authors have contributed it to the public domain under a BSD-2 license, it is also a valuable test and demonstration article. We gratefully acknowledge the effort and expertise that were invested in its creation.

Use case diagram. Figure 4 shows how the automated verification found a nonconformance with the requirement actors must have documentation" (actors outlined in red). The other red-outlined use cases violate requirements on use case documentation.

System requirements diagram. Figure 5 shows automated verification detecting non-conformances in a SysML requirements diagram. The model requirement being violated is "Requirements model elements must have one of (a) satisfied by relationship with another model element, or (b) a derived or refined relationship with another requirement that is satisfied by another model element." None of these requirements in this diagram had these relationships as shown by the red outline.

Internal block diagram: Figure 6 shows the result of running the validation rules against on an Internal Block Diagram. Among the requirements non-conformances that were found by the automated scripts are: (1) attributes and properties were not typed, (2) absence of signals on connectors, (3) not all blocks were connected with ports, and (4) absence of flows on connectors.

Parametric diagrams: Figure 7 shows a parametric diagram of a mass analysis. The automated verification rules detected (1) lack of value property typing and (2) a parameter without a binding connector.

Activity diagrams: Figure 8 shows an activity in which the output pin outlined in red does not have an outgoing object flow detected by an automated validation script. Figure 9 shows a similar requirements violation for a pin that does not have an output flow, but also has another violation: Decision nodes must have a name.

CONCLUSION

For projects using model-based systems engineering (MBSE), verification and validation of the models on which they depend is essential. This paper has described a method for SysML model verification and validation using both manual and automated methods. The overall approach is based on model requirements, which state both general user needs and specific model functions or characteristics. Requirements addressing the extrinsic properties of the model, its purpose, and interaction with other software are verified manually. Other requirements addressing model element use, syntax, and consistency can be verified using automated methods based on the scripting capabilities of SysML modeling tools. Together, the use of manually and automatically performed verification can lead to more correct and higher quality models. ■

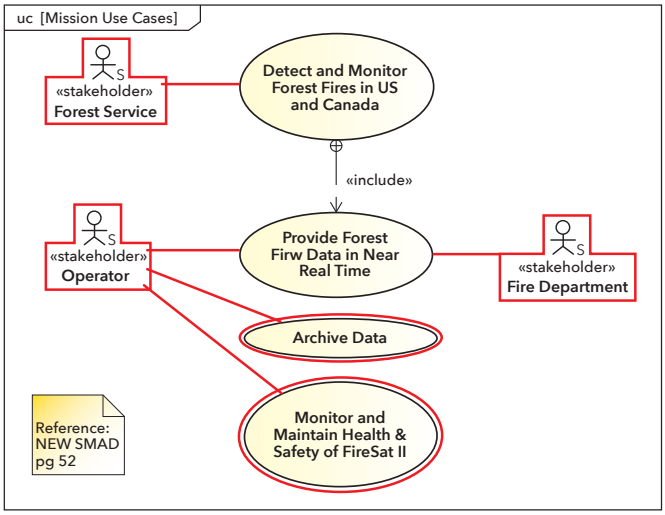


Figure 4. Output of automated verification rule checking for actor documentation

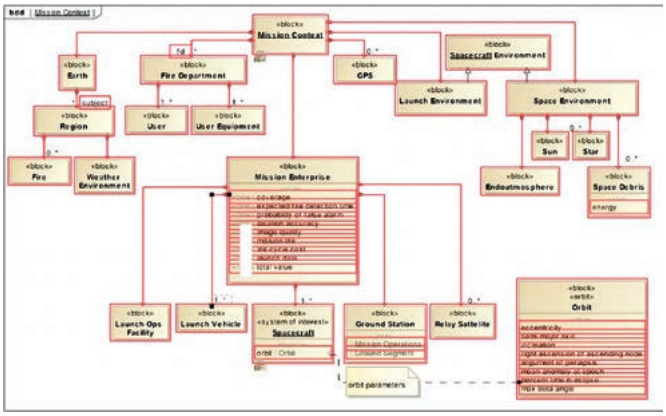


Figure 5. Output of automated verification rules on block definition diagrams

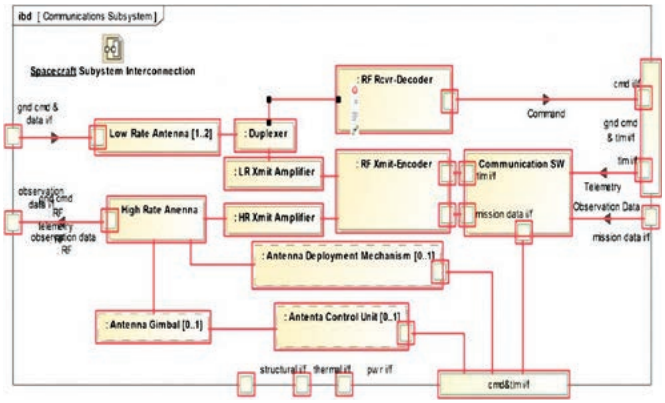


Figure 6. Output of automated verification rules on Internal Block Diagrams

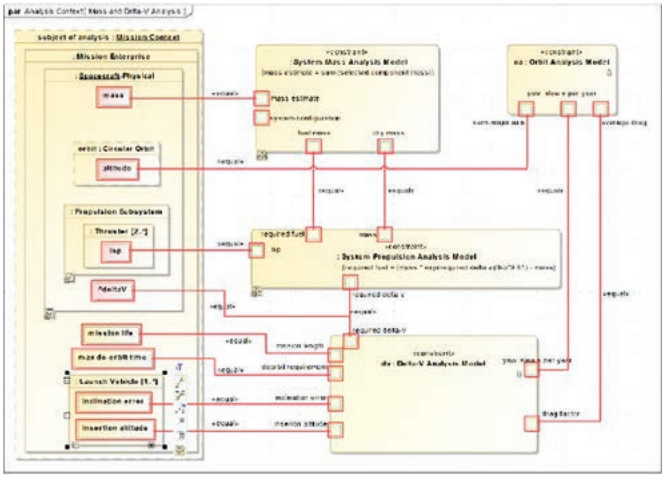


Figure 7. Automated verification rules output in parametric diagrams

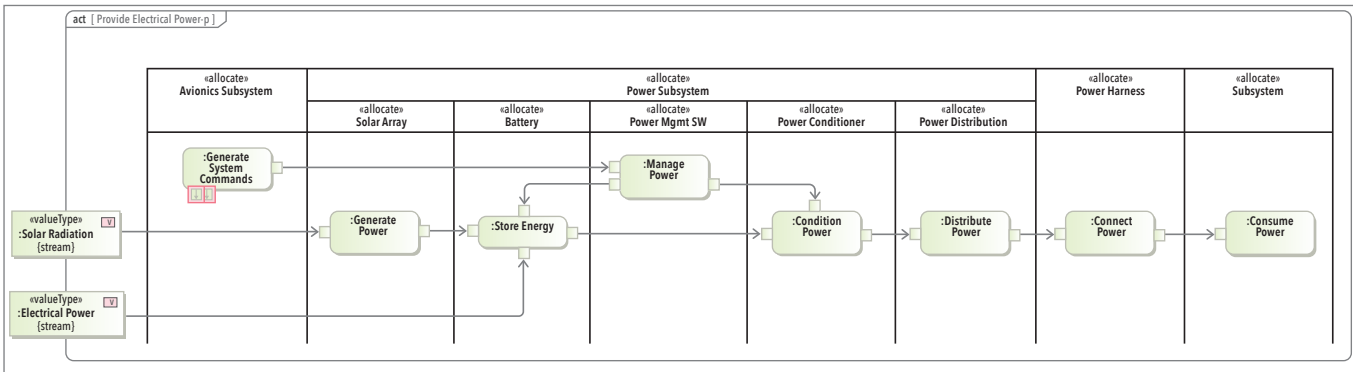


Figure 8. Activity diagram requirements violations detected by automated verification rules

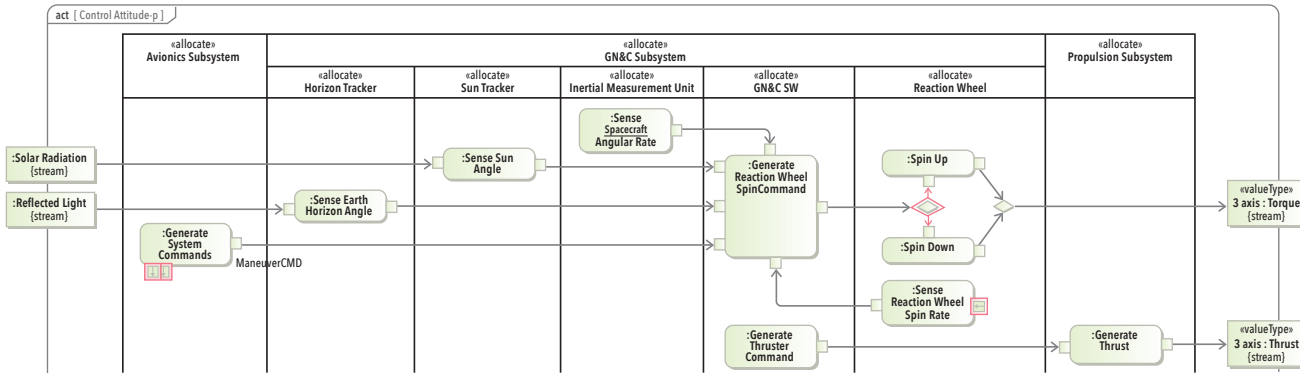


Figure 9. Requirements violations detected by automated verification rules displayed in the control attitude activity

REFERENCES

- 3DS/Nomagic. 2021. *Cameo Systems Modeler.* https://www.nomagic.com/products/cameo-systems-modeler.
- Ahmad, M. et al. 2013. *Early Analysis of Ambient Systems SysML Properties using OMEGA2IFx*. Reykjavik, IS. https://hal.archives-ouvertes.fr/hal-01085410. [Accessed 9 July 2020.]
- Baduel, R., M. Chami, I. Bruel, and I. Ober. 2018. SysML Models Verification and Validation in an Industrial Context: Challenges and Experimentation. *Lecture Notes in Computer Science*, 10890(ECMFA 2018: Modeling Foundation and applications), pp. 132-146.
- Douglas, B. 2017. *Harmony MBSE Modeling Guidelines*. http://merlinscave.info/Merlins_Cave/Tutorials/Entries/2017/5/26_Harmony_Modeling_ Guidelines_files/harmony%20mbse%20modeling%20guidelines%202.0.pdf: IBM Rational.
- Friedenthal, S. and C. Oster. 2017. *Architecting Spacecraft with SysML Web Site github site with BSD-2 clause simplified license (permissions for commercial use, modification, distribution, and private use).* http://sysmlmodels.com/spacecraft/models.html [Accessed 20 September 2020].
- Friedenthal, S. and C. Oster. 2017. *Architecting Spacecraft with SysML*. s.l.:CreateSpace Independent Publishing Platform.
- Friedenthal, S., A. Moore, and R. Steiner. 2015. *A Practical Guide to SysML, 3rd Edition*.
- Hecht, M., J. Chen, and G. Pugliese-Rosillo. 2021. *Verification and Validation of SysML Models*, El Segundo: Aerospace Corporation.
- ISO/IEC/IEEE, 2015. *Systems and Software Engineering – System Life Cycle Processes*. Geneva: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.
- Jarraya, Y. and M. Debbabi. 2012. "Formal Specification and Probabilistic Verification of SysML Activity Diagrams." IEEE Sixth International Symposium on Theoretical Aspects of Software Engineering, Beijing, CN, 4-6 July.
- McDermott, T. et al. 2020. *Benchmarking the Benefits and Current Maturity of Digital Engineering/Model-Based SE*. https://sercuarc.org/wpcontent/uploads/2020/03/Briefing_Benchmarking-the-Benefits-and-Current-Maturity-ofMBSE-3-2020.pdf. Systems Engineering Research Center.
- Object Management Group. 2017. *OMG SysML Version 1.5 Specification*. http://www.omg.org/spec/SysML/1.5/.: s.n.
- Pettit, R. 2003. *UML Inspection Criteria*, s.l.: The Aerospace Corporation, TOR 2003(1465)2412e.
- Rahim, M., A. Hammad, and M. Boukala-Iuoaleln. 2015. *Towards the Formal Verification of SysML Specifications: Translation of Activity Diagrams into Modular Petri Nets*. s.l., s.n.
- Selic, B. 2012. "What will it take? A view on adoption of model-based methods in practice." *Software and Systems Modeling* 11: 513-526.
- Vinarcik, M. and H. Jugovich. 2020. *Digital Engineering Validation Tool Enables Efficiency Gains*. https://www.saic.com/blogs/digital-engineering-validation-tool-enables-efficiency-gains: SAIC Corporation.
- Wolny, S., A. Mazak, C. Carpella, et al. 2020. *Thirteen years of SysML: a systematic mapping study. Software and Systems Modeling* 19: 111–169. https://doi.org/10.1007/s10270-01900735-y.

> continued on page 50

From Model-based to Model and Simulation-based Systems Architectures—Achieving Quality Engineering through Descriptive and Analytical Models

Pierre Nowodzienski, pierre.nowodzienski@thalesgroup.com; and **Juan Navas,** juan.navas@thalesgroup.com
Copyright ©2022 by Pierre Nowodzienski and Juan Navas. Permission granted to INCOSE to publish and use.

ABSTRACT
Systems architecture design is a key activity that affects the overall systems engineering cost. Hence it is fundamental to ensure that the system architecture reaches a proper quality. In this paper, we leverage model-based systems engineering (MBSE) approaches and complement them with simulation techniques, as a promising way to improve the quality of the system architecture definition, and to come up with innovative solutions while securing the systems engineering process.

INTRODUCTION

System architecture is a key engineering artefact in systems engineering. It permits engineers to reach a common comprehension of the expectations of their customers, to orchestrate the design of the subsystems and components to reach the system purpose, to compare alternative orchestrations, to propose solutions that are fitted to stakeholders' expectations, and to ensure that the engineering outcomes are compliant to these solutions.

The information contained in system architecture deliverables directly impacts (that is, an input for) a large part of other engineering activities and their related outcomes; and in complex systems engineering practice the system architecture is often considered as the backbone of a large part of the engineering process and beyond, covering the whole system life cycle. Hence, ensuring a good architecture quality is strongly contributes to securing the further engineering activities.

Model-based systems engineering (MBSE) approaches have proven their value on improving the quality of system architectures. When properly set up, MBSE practices and tools can support the architects and guide them through systematic workflows that reduce the risk of inaccuracy, inconsistency, and incompleteness of the design. However, our experience shows that in some contexts the current MBSE capabilities do not suffice to reduce these risks as required. This is due to both the increasing complexity of the systems under design, and to the increasing complexity of the engineering workflows and organizations put in place to develop our systems.

As systems engineering practitioners, we acknowledge the existence of extensions of MBSE approaches that make use of simulation to address these issues. However, we have identified a lack of formal, concrete, and effective proposals regarding the extension of MBSE methodologies to embrace analytical models.

In this paper, we present how, and under which conditions, simulation techniques can be articulated with MBSE methodologies so to fill the gaps of current MBSE approaches in ensuring proper quality architecture designs. We start by providing a necessary background on architecture design, MBSE for architecture design, and simulation of architecture design. Then we describe the objectives and the limitations of current MBSE approaches for architecture design, and present how simulation can contribute to overcome these limitations. A set of good practices for simulation is presented, along with an analysis of a case study that illustrates the proposals of this paper.

BACKGROUND

Systems architecture design

A system architecture is defined as the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution (ISO/IEC/IEEE 42010). The standard specifies the way system architectures are organized and expressed.

An architecture is defined with regards to a set of *stakeholders*, which are individuals, teams, organizations. or any other kind of entities having an interest in the system, for example. if the system of interest is a residential building, examples of stakeholders

	Perspective	Objective
NEED & CONTEXT PERSPECTIVES	Operational Analysis	What the stakeholders need to accomplish
	System Needs Analysis	What the system has to accomplish for the stakeholders
SOLUTION PERSPECTIVES	Conceptual Architecture	How the system will work to fulfill expectations
	Finalized Architecture	How the system will be developed and built

Figure 1. Arcadia engineering perspectives

are the promoter, the future homeowners, the future administrator, and the architect herself. A *concern* is a matter of relevance or importance regarding the system of interest to a stakeholder; for example, one of the architect's concerns would be to design the best possible atmosphere in which to live.

A (stakeholder) *perspective* is a way of thinking about an entity, especially as it relates to concerns, for example, how future homeowners may circulate across the residential building. An architecture *aspect* is a unit of modularization of concerns within an architecture description, capturing characteristics or features of the entity of interest, for example, the organization of common areas in the building and in its green zones. An architecture view is an information item comprising part of the architecture description, for example, a plan of the green zones and its circulation paths.

Model-based systems architecture design

Model-based systems engineering (MBSE) is the formalized application of modelling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases (INCOSE 2014). In addition to providing an increased rigor in these engineering activities, one essential objective of a model-centric approach is to provide authoritative sources of truth that can be shared with all stakeholders.

Overview of the Arcadia method. Arcadia is a model-based method devoted to systems, software, and hardware architecture design (Voirin 2017). It describes the detailed reasoning to understand and capture the customer need, define, and share the product architecture among all engineering stakeholders, and validate early and justify the design. Arcadia can be applied to complex systems, equipment, software, or hardware architecture design, especially those dealing with strong constraints to be reconciled such as cost, performance, safety, security, reuse, consumption of resources, mass, and so forth. It is intended to be embraced by most stakeholders in system/product/software/hardware definition as their common engineering reference. It has been applied in a large variety of engineering contexts for more than 10 years.

Inspired on ISO/IEC/IEEE 42010, the Arcadia method defines a set of perspectives that the system architect can adopt when designing the architecture. The four main perspectives are summarized in Figure 1. Arcadia enforces a clear separation between the capture and analysis of the system context and needs (the operational analysis and system need analysis perspectives), also called here *need perspectives*, and the design of the solution (the conceptual and physical architectures) also called here *solution perspectives*.

When following the Arcadia method, the system architect is led to consider five main aspects during the process of defining the system architecture:

- The *purpose* – the reason to exist of the entities, including the system: the services it provides in different contexts of usage, so to fulfill stakeholders' expectations and provide them with valuable solutions.
- The *form* – the entities that are considered during the different contexts of use of the system, including the constituents of the system; here two sub-aspects may be considered: the *structure* of the architecture and the *interfaces* between its constituents.
- The *behavior* – the expected behavior of the entities in the context of usage, including the expected behavior of the system and of its constituents; here two sub-aspects can be considered: the *functions* that emerge from a functional analysis, and the *modes & states* of the constituents.

These aspects shall be considered for every perspective stated before. In addition to these aspects, Arcadia can be extended through viewpoints that address other aspects such as safety or cybersecurity, by defining new concepts, views and practices, and how those depend on or impact the Arcadia standard concepts, views and practices.

The maturity of a system architecture, or of a reference architecture (Cloutier 2010), especially in a product line context (Oster 2016), can be evaluated by considering the aspects that have been addressed in the architecture design matrix shown in Figure 2.

As the lack of properly tailored tools has proven to be a major obstacle to the implementation of MBSE in industrial organizations (Bonnet 2015), Arcadia is recommended to be implemented using the open-source modelling workbench Capella, whose diagrams are inspired from SysML and that has proven suitable for systems engineers with diverse backgrounds and skills (Capella 2021a).

Overview of Arcadia concepts. In this paper we will use a subset of the concepts defined by Arcadia. For a sake of clarity, this section provides a brief definition of them. A *capability* is an entity's ability to

		ASPECTS				
		Perspective	Function	Modes & States	Structure	Interfaces
NEED PERSPECTIVES	Operational Analysis What the stakeholders need to accomplish					
	System Needs Analysis What the system has to accomplish for the stakeholders					
SOLUTION PERSPECTIVES	Conceptual Architecture How the system will work to fulfill expectations					
	Finalized Architecture How the system will be developed and built					

Figure 2. Arcadia architecture design matrix

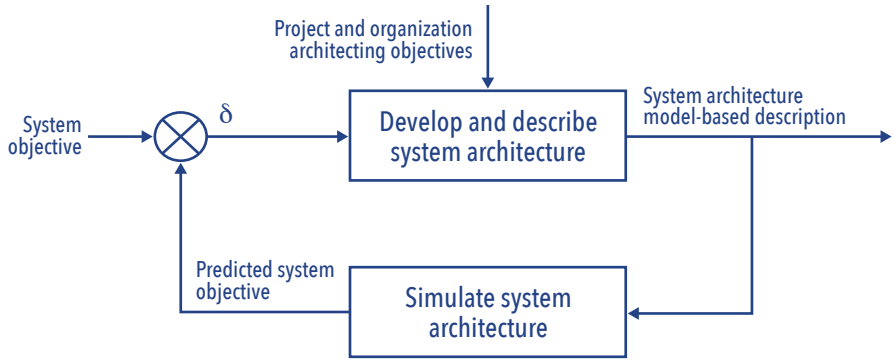


Figure 3. Regulation role of simulation in the architecture design process

supply a service. A system capability represents a usage context and is characterized by a set of *functional chains* and *scenarios* that describe the system behavior in a particular usage.

Both functional chains and scenarios reference *functions*, which are actions, operations or services performed by entities – the system, one of its components, or also by any other entity interacting with the system. They also reference *functional exchanges*, that express possible interactions between source and target functions, and which are further characterized by *exchange items*, which reference the elements routed together during an interaction.

Entity's behavior can also be represented by *modes*, which are behaviors expected under chosen conditions; *states*, which are behaviors undergone by entities in conditions imposed by the environment; and *transitions*, which are changes from one mode/state to another.

Simulation of an architecture design

Wippler (2018) provides a broad definition of simulation as a cognition process that predicts effects, including any form of deliberation based on a model, and not only the use of a model executable by computer means. This permits us to consider simulation as a component of a control mechanism and hence as a valuable means for reaching a system architecture design that satisfies the system objectives and the stakeholders' expectations. Figure 3 illustrates the role of the simulation in the architecture process in the control loop.

By considering simulation as part of the engineering control loop, we can identify the contribution of simulation in the architecture design process, both in terms of reaching the objective quality of the system, and of the time required to reach this objective quality.

Regarding reaching the objective quality of design, experience shows that 50% to 70% of the total of design errors made during system development lifecycle are introduced before implementation phase, mainly during the orientation and design phase. It means that we expect at least half of all the design errors already exists at the preliminary design review (PDR). We can expect the quantity of errors to increase with the increase of system complexity the industry is facing, according to the metrics the aerospace domain provides regarding the growth of embedded software complexity correlated with the number of source lines of code (SLOC) (Filho 2018).

Regarding the time required to reach the objective quality, simulation contributes to reach it sooner, reducing the design error costs. As presented in Stecklein (2004), design error fixing costs increase exponentially with project phase: if the cost of fixing a requirements error discovered during the requirements phase is defined to be 1 unit, the cost to fix that error if found during the design phase increases to 3 – 8 units; at the manufacturing/build phase, the cost to fix the error is 7 – 16 units; at the integration and test phase, the cost to fix the error becomes 21 – 78 units; and at the operations phase, the cost to fix the requirements error ranged from 29 to more than 1500 units.

Whereas the control loop is a useful pattern for identifying the role of simulation in architecture design, not all simulation mechanisms have the same effectiveness, and each mechanism requires different resources to be mobilized to ensure their effectiveness. Three existing mechanisms are:

- Workshops in which experts deliberate about the expected behavior of the system are relatively easy to perform, but its effectiveness strongly depend on factors that are difficult to assess, such as the skills of the experts and their ability to work together.
- Automatic validation rules, such as the ones provided by MBSE tools like Capella, are more reliable as they are based on the know-how of many experts and the return of experience of past projects. However, they only cover some quality aspects of the design, often its correctness and consistency.
- Computer-assisted simulation, in which executable models are created or generated from design models, is a powerful mechanism that may cover a large scope of quality aspects. However, the cost of putting in place and maintain simulation mechanisms is also higher than the previous ones.

In the rest of this paper, we will focus on this third mechanism. From now on, the term "simulation" alone refers to computer-assisted simulation.

ARCHITECTING OBJECTIVES, CURRENT STATE, AND LIMITATIONS

Architecture design objectives

As the architecture design strongly affects a large set of the engineering activities, both the quality of the system architecture artefact and the efficiency of the architecture design activity are key optimization levers for engineering organizations.

The usages of the architecture are manifold and depend heavily on endogenous parameters such as the engineering organization, the engineering process and practices, the engineering teams, the industrial domain constraints and regulations, the expectations of the customers in terms of engineering activities, among many others. However, our experience shows that the objectives of engineering organizations with regards to the architecture design activities can be categorized in:

- *Share* – improve communication, reduce ambiguities and reach, as fast as possible, a common understanding of the system's purpose, form and behavior, for example, use model-based architecture views to support discussions with customers.
- *Secure* – guarantee the quality of the engineering data, hopefully at any moment of the engineering process and beyond, for example, use architecture models as the backbone for textual requirements elicitation, analysis, and documentation (Bonnet 2019).
- Automate – automate the execution of the engineering processes to improve its efficiency, through (i) automatically generating engineering artefacts from architecture models, and (ii) automate architecture tasks, for example, generate document deliverables from models, generate software modules from architecture models, and automate the exploration of solution spaces.

The Share objective of architecture design deserves a few more words. This one is often despised and reduced to "having good-looking diagrams", ignoring that:

- Improving communications requires a common (architecture) language and to

use a common vocabulary, which is not a trivial task.
- Fluency in (technical) communication drastically reduces the time required to understand what the other says, enables active listening attitudes, and ultimately improves the efficiency of collaboration in engineering.
- Clear, concise, properly organized and well documented architecture design models and views, can have a positive impact on the perceived quality of the information contained in them, and on the perception of quality of the system itself (Iandoli 2018 and Bateman 2010).
- Experience shows that the share objective is often the first one that is set by engineering teams with few or no prior knowledge on MBSE practices and tools. This often means that the share objective is their first action of a wider change management initiative in their organization (for example, a digital engineering transformation). The results of these first change management actions often determine what happens next.

Current state and limitations

Architecture design descriptions, such as the ones that can be obtained using the Arcadia method and its associated MBSE tool Capella, allow engineers to address the objectives presented above. Indeed:
- The systematic use of Arcadia perspectives in Capella (cf. Fig. 2), from operational analysis to physical architectures and product breakdown structure definitions, permit engineers to build a complete and consistent digital thread from the expectations of their stakeholders to the detailed definition of system components and expected behavior. This is particularly important both for business; to ensure that the expectations of future customers are properly addressed, and for certification concerns in mission-critical systems; to ensure that high-level safety and security concerns are properly addressed by lower-level components and their implementation, and that they can be validated.
- The systematic use of Arcadia aspects (cf. Fig. 2) permit engineers to ensure that each perspective is analyzed consistently and exhaustively. Arcadia aspects complement each other, for example, by performing an analysis of components' modes and states, a previously developed functional analysis is completed through the addition of new functions or the clarification of the textual requirements attached to them.
- The kinds of model views proposed by Arcadia were defined in close collaboration with systems engineering practitioners. Several of them are based on or are the same as SysML ones — such as modes and states machines or sequence diagrams – while others correspond to diagrams that have been drawn by systems engineers for many years – such as views presenting multiple layers of functional decompositions, or views presenting the structure, functional and interfaces aspects together (Capella 2021b).

Nevertheless, experience on implementing MBSE in the field has identified a set of limitations of the current tooled-up practices:

Describing the expected behavior. One of the main goals of architecture design is to reach correct, complete, and error-free descriptions of the system behavior that is agreed with the customer, and of components behaviors as designed by architects and agreed by components' engineering teams.

Arcadia concepts and methods, the fact that these methods are embedded in tools, and tool-specific features such as validation rules, all contribute to reaching correctness and completeness to a certain extent. However:
- Ensuring a full consistency between Arcadia aspects, and particularly between the functional analysis and the modes and states analysis, require the execution of cross-checks that in many cases (i) are specific to the project's needs, and (ii) cannot be specified and automated easily.
- In contexts such as Systems of Systems (SoS), or systems made mainly of subsystems, architecture models alone do not suffice to understand and master the behaviors that emerge from the composition.
- In contexts on which the system integrates components that are capable of learning and of adjusting their behavior to improve their efficiency (for example, artificial intelligence (AI) powered components), current means for describing the system behavior need to be extended to consider different ranges of operation derived from the integration of these components.

Ensuring the quality of the integration, verification, and validation (IVV) strategy. In many of our engineering contexts, IVV strategies shall be defined as soon as possible, to secure the testability of the design, anticipate the means that will be required for IVV tests, and secure the execution of IVV campaigns. Therefore, architecture design is a major input of the definition of IVV strategies.

Ensuring the quality of the integration, verification, and validation (IVV) strategy. In many of our engineering contexts, IVV strategies shall be defined as soon as possible, to secure the testability of the design, anticipate the means that will be required for IVV tests, and secure the execution of IVV campaigns. Therefore, architecture design is a major input of the definition of IVV strategies.

FILLING THE GAPS WITH SIMULATION

One major root cause of the limitations identified above is the fact that architecture models today are mostly descriptive, and that prediction capabilities are limited. Descriptive models are different in nature from predictive ones, as they retain "gray areas" of ambiguity, and do not need to perform as accurately as the predictive models. Although allowing designers to define their architecture while keeping some gray areas has proven useful at introducing model-based engineering approaches in large organizations, these gray areas may induce negative consequences in the engineering contexts presented above.

To overcome all these limitations, simulation offers capabilities to enable engineers to access the behaviors of the solution being designed. Simulation provides to an engineer an executable virtual object that can be exploited to enhance communication between all stakeholders, to further evaluate and analyze the solution definition as well as to expand and automate the exploration of the solution space.

Improved communication through simulation

Descriptive models are often used to illustrate how the system behaves, either with modes & states machines or scenarios and functional chains. The overall behavior of a system is then captured through a collection of such views, each one focusing on describing the behavior in a unique situation. Additionally for these types of views there is a compromise between completeness of the behavior description (completeness = unambiguous + full coverage of the scenario) and the easiness of readability.

The spreading of the information across views and the lack of readability or unambiguity (due to the above compromise) makes it hard to ensure a common understanding of the system behavior among stakeholders. Furthermore, the reader performs a model execution mentally to figure out what is the behavior described. In this case, the execution engine (that is, the human brain) is different from one reader to another and thus leads to a variety of simulation results (that is, interpretation) and not repeatable that dramatically increases the risk of inter-

pretation divergence among stakeholders.

Simulation provides unambiguous means to figure out the system behavior such as time dependent curves, 2D and 3D rendering, and quantitative results based on post-processing of simulation outputs. The use of one or another depends on the purpose of the communication, the type of information to be shared, and the profile of the persons the information is shared with.

In addition to the simulation results themselves, the use of interactive simulations let the stakeholders (engineers, end-users, customers, managers ...) experience the future system for themselves, confirm or refine the expected behavior, or to get a better understanding of the system behavior for detailed design. The use of simulation to improve communication is mainly facilitated by "simulation as a service" capabilities as it provides easy access to simulation means (execution and visualization) while complying with security constraints.

For engineers in charge of the detailed design, an executable specification provides extra-benefits by securing the understanding of the expected behavior. The simulation environment provides similar capabilities to the one provided by software IDE such as breakpoints with pause conditions, step forward and backward, model animation with highlighting or coloring, toggling data and much more. This set of debugging features clarifies the internal mechanisms and interactions between functions from which the system behavior emerges.

Another observed benefit is the reinforcement of concurrent engineering and collaboration. For system engineers, being able to share very early an executable specification that reflects certain aspects of the system with other engineering specialties is a powerful mean to enable the other specialties to start activities earlier and concurrently, to quickly iterate on the system definition as early as possible. Without the use of simulation these activities would have been performed far later or in a less complete and detailed way. For instance, an IVVQ engineer can start experimenting test cases with a virtual test bench and virtual object that mimics the system to test. This way the test campaign gets more mature and is verified before the execution of the test campaign on the real system. Actually, the test campaign elaboration and verification activity performed early in a virtual environment also offers opportunity to not only verify the test cases but also to provide feedback to systems engineers on defects detected during the test's execution. In this case we can see the simulation as a concrete enabler for co-engineering. Another example is for safety engineering. Although the use of descriptive models is already a proven way to perform safety analysis (R. de Ferluc 2018) early in the design process based on the architectural descriptive model produced by the system engineering team and to feed the architectural model back with the analysis results, the use of simulation offers more advanced analysis capabilities for reliability, availability, maintainability, testability, and safety analysis (A. Garro 2012) that can be integrated in a feedback loop process on system architecture design activity (cf. Fig. 3). This feedback loop can occur at any stage of the system architecture design.

Secure the design through simulation

Leveraging simulation in the engineering process enables engineering project teams to early and continuously integrate, verify, and validate (IVV) the solution under design (that is, through execution of a subset of the system definition, called "item under design"). Having the capability to simulate the item under design at any time enables engineers to perform unit-test and integration-test to verify and validate the item under design after each small design step.

This brings two major benefits. The first benefit is that it drastically reduces the overall solution cost reducing the time between the introduction of a design error and the fix, as (i) the earlier an error happens the more the error might cost; (ii) the majority of errors occur before starting the implementation, which is relatively early in the development process; and (iii) the situation gets worse as the complexity of the systems increases.

Given this situation, it is critical for engineers to have means to test their design before the implementation, hence before it is physically accessible. Having access to virtual test means including a virtual item under test and virtual test bench at any time through the development process is an answer to minimize the cost of errors.

A concrete example is a system engineering team who captured the logical architecture of the system (structural and functional aspect) by following Arcadia methodology and language. The systems engineers wanted to be more precise and unambiguous regarding the dynamics of exchanges between functions, as there were many intricate feedback loops and parallel branches. One system engineer started to build this simulation and it appeared the intellectual process required to build the simulation raised a lot of questions challenging the content of the logical architecture. He found ambiguities in the behavior description and missing data in interfaces. These errors have been fixed very early, avoiding later discovery by other engineering teams who would have spent a lot more time dealing with these errors before reaching out to the systems engineering team.

The second benefit is that it enables wider solution space exploration and alternatives comparison. Due to project planning pressure, systems engineering teams rarely invest the time required to fully explore the solution and design space as well as comparing alternatives. These are activities that lead to producing designs that will not be retained, and simulation reduces the time required to evaluate designs and determine which one is better than others, hence making the design space exploration and alternatives exploration more efficient. To further speed up the exploration, the simulation model can be parametrized and integrated into an optimization loop to automate this exploration and to converge towards the best solution.

As described in the previous section ("Improve communication with simulation"), leveraging simulation not only helps to secure the design with early and continuous IVV but also permits IVVQ engineering teams to secure the test campaign concurrently with the design activity. This opens new methodologies that can be applied to complex systems engineering such as test or behavioral driven development, provided the IVVQ team can provide comprehensive virtual test means for the foreseen solution to the system engineering team.

Automate tasks with simulation models

Working with models, be they descriptive or executable, means working on digital data. Hence, access to and transformation of this data can be automated with programming languages. MBSE tools often provide tasks automation such as document generation, code generation (for software component interfaces mainly) or descriptive models generation – for system to sub-system transition or to initialize dysfunctional models for instance. We can extend the range of tasks and enrich them based on simulation models. Maybe the more valuable capabilities are the design space exploration automation, design optimization automation, early test execution automation and model coverage analysis automation (Camus 2016) to achieve high quality system definition.

Based on simulation models we can also enrich the document generation with parts of simulation models that would be non-ambiguous or with simulation results to better express a desired behavior. And finally, as simulation models are executable by nature, we can generate both the detailed software component interfaces along with the internal behavior of the component (Fleischer 2009).

GOOD PRACTICES PROPOSALS

As discussed above, there are many reasons why we would want to rely on simulation. Hence, adopting a simulation-based engineering process, that is, the use of simulation to support all engineering activities throughout the product lifecycle, leads to the creation of a multitude of simulations. As for any engineering activities, keeping consistency and coherency across all engineering artefacts including simulations is a challenge. To overcome this challenge there are good practices to apply.

Separation of concerns. An all-in-one model merging descriptive architectural models and simulation may be considered as a response to the need of coherency and consistency. We encourage separation of concerns to keep architecture definition and simulations as distinct, differentiable but still coherent objects. This is to avoid the overload of architectural models that support specific modeling objectives owned and defined by system architects. The system architect often requires a simple model to support the design thinking process. At some point, the overload of information in a single representation becomes counter-productive to achieve system architecture goals. This choice is like the one existing in hardware engineering where there is a clear separation between modeling and simulation respectively named computer-aided design (CAD) and computer-aided engineering (CAE).

Frame simulation design in your MBSE methodology. To help organize simulations and to ease the coherency and consistency of them with other engineering artefacts, simulation models should contribute to meet objectives defined in the applied MBSE methodology. For instance, if a simulation is built in the context of the system analysis, then this simulation must contribute to the system analysis objective, .that is, to define what the system must accomplish for the stakeholders. Furthermore, the simulation model should respect the same representation point of view as the one adopted in the Arcadia perspective. Again, if we consider a simulation built at the system analysis perspective, the way the system behavior is captured must be considered as a "black box" specification and not an implementation specification for the subsystem engineering teams.

One simulation per type of concern. Considering a single simulation to answer all the questions the engineers would have throughout the product lifecycle would mean this simulation is a perfect virtual representation of the system in all its aspects throughout all its history. This is neither realistic nor achievable. Hence, we must consider the worst case that we would have one simulation for each engineer's questions that will occur during the product lifecycle. To reduce the number of simulations one proposal is to define typology of concerns and to build one simulation per type of concern or at least reusable simulation components for each type of concern. To ease the management of these simulations a proper governance is needed.

Maximize reuse of simulation assets. There are several ways to reuse simulation assets: (i) reuse existing simulation components to build a new simulation, (ii) reuse an existing simulation to answer different questions from similar concerns, and (iii) reuse test harnesses and test cases throughout the development lifecycle. For the first type of reuse, we need to define simulation modeling rules that ensure reusability of components (for example, interoperability formats such as FMU, HLA..., shared interfaces repository, ...), we must also minimize the number of simulation tools to keep the co-simulation constraints as low as possible, and define a standard component-based simulation architecture. For the last type of reuse, we need to ensure that the simulation facilities offer seamless support for software-in-the-loop (SIL), processor-in-the-loop (PIL) and hardware-in-the-loop (HIL) to progressively transition from a full virtual testing to real system testing.

Automate the transformation of architectural model element to simulation model element. To avoid human mistakes in the design of a simulation model based on existing architectural model, we strongly encourage automating this model transformation. This provides confidence in the coherency and consistency between architectural models and simulation models. The automation can also integrate the traceability links to ease the impact analysis and justification activity. A subsequent advantage of such automation is the standardization of the approach to building simulation models in the engineering organization.

CASE STUDY, ANALYSIS, AND DISCUSSION

To test the proposed approach, we used a hypothetical, still realistic, drone-based system capable of fulfilling multiple missions including the evaluation of the health of crops. Its architecture has been formalized in Capella following the Arcadia method, exploiting the Arcadia perspectives. The following paragraph is an excerpt showing how an expectation of a stakeholder (evaluating crops in very large fields) drives to key capabilities (navigation) and key system and subsystem requirements (mass, integrity).

- Operational analysis – by identifying operational entities (OE) and the analysis of their perceived value (for example, pains & gains), operational activities and processes. For instance, farmers (an OE) are interested on evaluating crops in very large fields.
- System needs analysis – by defining (i) the system capabilities and related functional chains and scenarios, and how they provide value to the stakeholders (the OE); and (ii) the non-functional key system requirements. For instance, the drone-based system shall provide functional capabilities such as navigation, acquisition of data, or mission planning; key system requirements include integrity–related to the capability of navigating while avoiding obstacles.
- Logical architecture – by allocating functions to the conceptual logical components (LC) of the solution. For instance, both the drone and the ground station LC contribute to the navigation capability, including the obstacle avoidance integrity feature. Key requirement for the drone subsystem is mass – as an enabler to navigate across large fields during the time required to do an evaluation.
- Physical architecture – by performing a detailed functional analysis and allocation of functions to the concrete components of the architecture. For instance, detailed requirements regarding integrity (including those related to obstacle avoidance) would be allocated to the navigation processors, and mass requirements resulting from mass versus fuel capacity tradeoffs may be allocated to drone physical components.

In the case study we built simulation models to support architecture objectives of operational analysis, system analysis and physical architecture perspectives. On the operational analysis, simulations are used to verify and validate the operational processes and to optimize them. We have used a modeling language inspired from eFFBD to represent in a simulation tool the operational processes. The execution of the simulation is based on discrete event simulation which is a convenient type of simulation engine for processes that requires not much behavioral details to get ready to run. Hence it appears to be very well suited for early simulation of processes. Systems architecture design is a key activity that affects the overall systems engineering cost. Hence it is fundamental to ensure that the system architecture reaches a proper quality. In this paper, we leverage model-based systems engineering (MBSE) approaches and complement them with simulation techniques, as a promising way to improve the quality of the system ar-

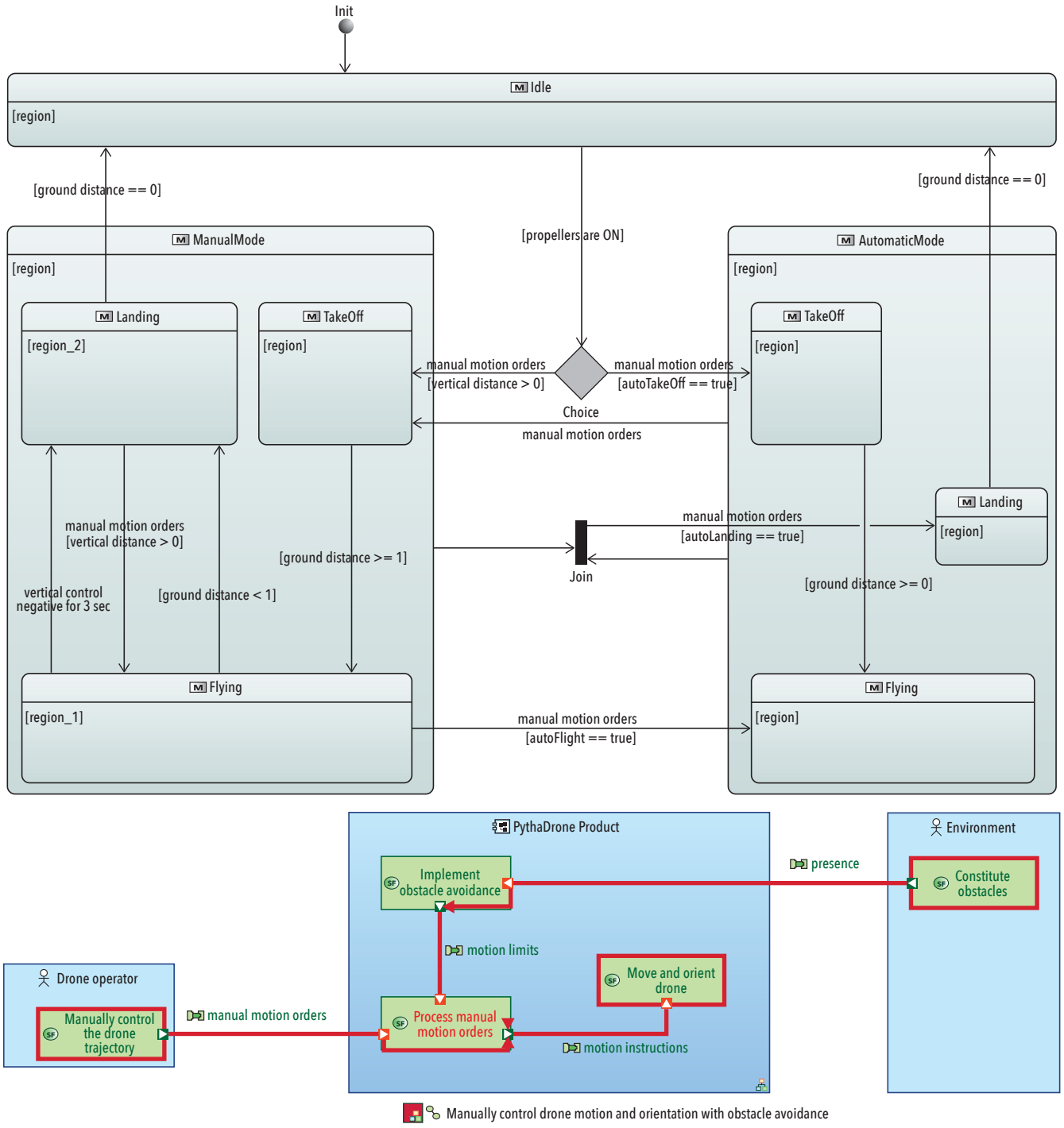


Figure 4. Drone-based system case study's mode machine (top) and functional chain (bottom)

chitecture definition, and to come up with innovative solutions while securing the systems engineering process. Systems architecture design is a key activity that affects the overall systems engineering cost. Hence it is fundamental to ensure that the system architecture reaches a proper quality.

On the physical architecture, we performed simulation to verify the correctness of the functional analysis especially regarding the definition of the functional exchanges and to identify potential ambiguities in the architecture definition. We also performed architecture optimization leveraging a parametrized simulation model. The purpose was to optimize several characteristics of the system to reduce the overall mass of the drone given a collection of operational missions to accomplish.

Focusing in the system needs analysis, we leveraged simulation to (i) verify and validate the modes machine supervising the system, (ii) specify the expected behavior of a subset of functions in a functional chain context, (iii) integrate these functions to build the functional chain and verify and validate this functional chain, (iv) ensure a full consistency between the functional chain and the system modes as they interact each other, and finally (v) validate the emerging behavior coming from these interactions.

Figure 4 shows the modes machine used for the case-study, which is the one supervising navigation modes of the drone. Figure 4 (bottom) shows the functional chain

used for the case-study, which involves the functions in charge of manually pilot the drone with an obstacle avoidance feature to preserve the integrity of the system. The transitions of the modes machine are triggered by functional exchanges involved in the functional chain and the modes update function parameters to adapt the overall behavior of the system given the active mode.

It shall be noted that for all perspectives, simulation results were injected back into the architecture descriptive model, ensuring the regulation role of simulation depicted in Figure 3. Two cases were identified, they are illustrated below with examples of a simulation of the physical architecture:

- The simulation results modified the value of a property of an existing architecture element – for instance, the mass ranges of the concrete physical components of the architecture, which are key requirements of the subsystems directly impacting the customer satisfaction (been able to evaluate large fields).
- The simulation results induced stronger modifications of the architecture, such as new functions, different allocations or new operating modes, that often require specific co-engineering actions – for instance, the early simulation of the obstacle avoidance feature led to these kinds of modifications, done under the responsibility of the system architect and involving several subsystems' architects.

Limitations of the descriptive architecture model

On this scope, the architecture model has limitations to completely capture the desired behavior and to ensure full consistency between modes analysis and functional analysis.

First, it is nearly impossible to specify with a descriptive model only the desired dynamic of the drone, that is, how the drone reacts on an operator order. Similarly, it is hard to explain how aggressive the response to an upcoming obstacle should be, or how the drone should behave when concurrent orders come from the operator and the obstacle avoidance functions.

Second, the content of modes and states machines must deal with a compromise between readability on one hand and formal expression and completeness on the other hand. By experience, the architecture model often favors the first aspect. Furthermore, even if the tool offers state machine simulation capability it is often at the price of detailed implementation effort. Hence, we must often deal with ambiguities and uncomplete specification.

Third, if the modes machine is growing in complexity, it becomes hard to verify it by mental simulation covering all the possible paths. Hence the potential is rapidly growing. Finally, it is hard to really capture the emerging behavior coming from the interactions between the functions, the supervising modes and the system states. It is also challenging to ensure full coherency across these three aspects of the system.

Simulation design workflow and benefits

In the context of the case-study, we use Simulink as the main simulation tool as this tool supports the simulation of discrete event systems, state charts, data flow (discrete or continuous time) and acausal systems (suited for multi-physics simulation). To support the simulation design, we have tooled-up the transition from Capella to Simulink. This tool (called Cap2SL) ensures the coherency, consistency, and traceability of shared elements. To maximize reuse across simulation models, Cap2SL implements modeling rules favoring componentization and modularity.

In this workflow we consider two roles: the system architect is responsible for the architecture definition and generates a simulation request; the simulation engineer is in charge of building and running the simulation and providing data required by the request.

Case 1: Verify and validate modes machines. To build the simulation model from the modes machine defined in Capella, first we get an initialized version generated with Cap2SL. This simulation model captures the information stored in the Capella model. A substantial amount of data can be transformed applying a semantic mapping rule with no ambiguities. However other mapping rules might be applicable depending on the modeling rules a company wants to apply and the meaning a company confers to modeling patterns. In case of potential ambiguities to map the semantics of the two tools, the information is transformed as textual information for the simulation engineer to have all existing information to build the simulation model. In our case, we must deal with:

- Transition trigger expressed in natural language such as "propellers are ON". For such trigger, additional data manipulated in the system needs to be defined.
- Ambiguous guard expression: the data used to express the condition is part of several exchange items and these exchange items are carried by multiple functional exchanges. Hence, we don't know if the data has a unique or several instance and if so, then which instance is to be tested in the condition expression.

For these two problems we already see value of simulation just in the process of building a simulation. It permits us to challenge the system definition in front of a sort of a reality. Thus, to identify incomplete or too ambiguous specifications very early in the development lifecycle. Indeed, these issues would have been raised either by the subsystem engineering team or by the software team and some of these issues are tightly connected with hardware because

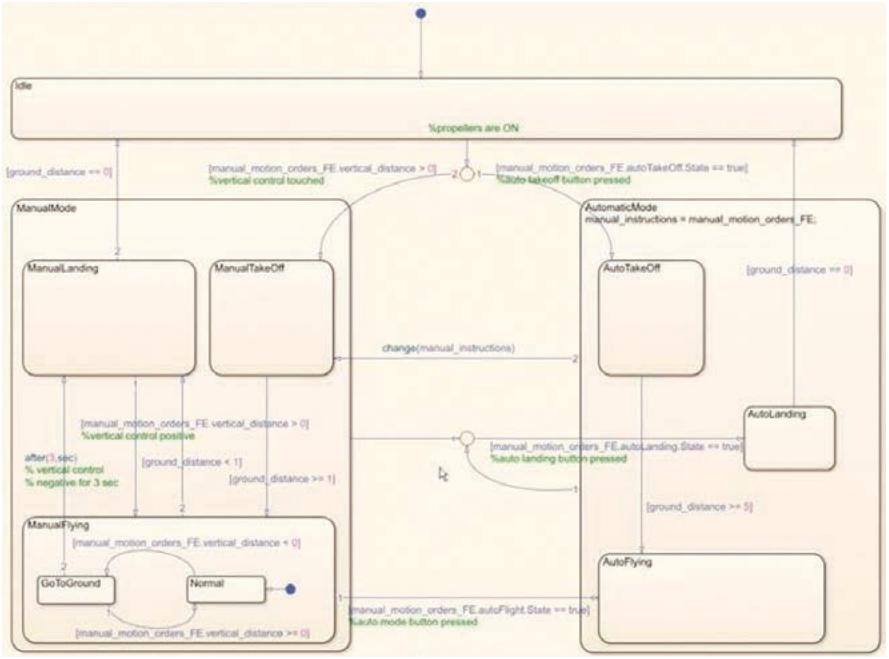


Figure 5. Drone-based system case study's mode machine executable model

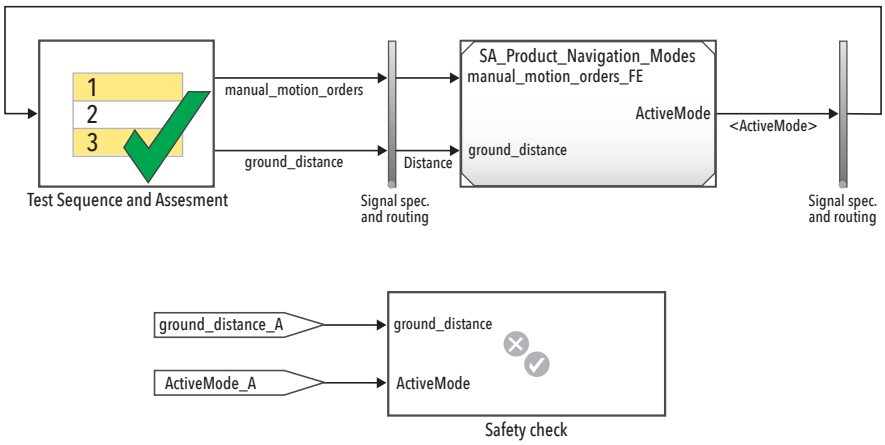


Figure 6. Test configuration

of the required data to capture by a sensor. Hence some of these errors would have led to costly rework and planning shift.

After some collaborative work sessions with system architect and simulation engineer the simulation engineer comes up with an executable model shown in Figure 5. Concurrently, the IVVQ engineer develops a test harness shown in Figure 6, reusing the interface definition generated by Cap2SL, and a test scenario based on the scenarios captured in Capella.

From this point, the early-testing activity can start. In the specific context of this case study, a lot more effort has been spent on fixing and improving the test scenario than on designing and fixing the item under test. We also used model coverage features offered by Simulink to identify missing scenarios. The early test raised a design error in the modes machine that was introduced in the Capella model and impacted the functional exchanges and exchange items definitions.

Case 2: Specify functions' behavior and verify functional chain. As for case 1, we first started by leveraging Cap2SL to initialize the functions models, the functional chain (FC) model and all the required data and interfaces. The FC model is made by aggregation of function models individually created and with their own lifecycle. At this point the FC model is compilation-ready. It means that the model does not contain design error that prevents to generate an executable software from the model. This already verifies the correctness of the Capella model on this scope.

From here we start work sessions with the system architect and simulation engineer to define the desired behavior of each function individually as the design model is not well suited to capturing this. As we are at the system analysis perspective, we can keep things with a high level of abstraction leading to interactive and collaborative work session iterating quickly on the simulation model. The main challenge is to get a robust set of test scenarios to ensure we don't miss corner cases. Concretely, it is easier to specify expected behavior and corresponding scenarios at the FC boundaries than specifying the details of each function individually. Hence, we started by defining the test scenario for the FC and then working function by function in the same order than the sequence defined in the FC. For a given function involved in the FC, the system architect defines at a coarse grain level the expected behavior of the function. The test scenario is generated by executing the FC scenario and recording the outputs of the upstream functions. The key point is to quickly get a first executable FC model to leverage the better-defined FC expected behavior to identify errors in the individual functions' definition.

In our case we decided to develop an interactive control dashboard to let the system architect and other stakeholders somehow related to this functional chain to experience by themselves the system within the FC scope to provide feedback.

Case 3: Integrate modes machine with functional chain. Once we get both the modes machine and the FC models verified, it becomes very easy to integrate them together, thanks to the componentization modeling rules we implemented in Cap2SL. The main concern we had was regarding how we should model the interactions between the modes machine and the functions since they do not concretely appear as interfaces in the design model. Either we decide to keep the Capella layout to make the system architect more comfortable to dig into the simulation model, or we make these interfaces visible in the diagram but by doing so we modify the interface definition of the functions. Since the beginning we favored keeping the layout of Capella for modeling in the simulation tool, so we continued with this approach for the last stage. However, this choice can be discussed. The

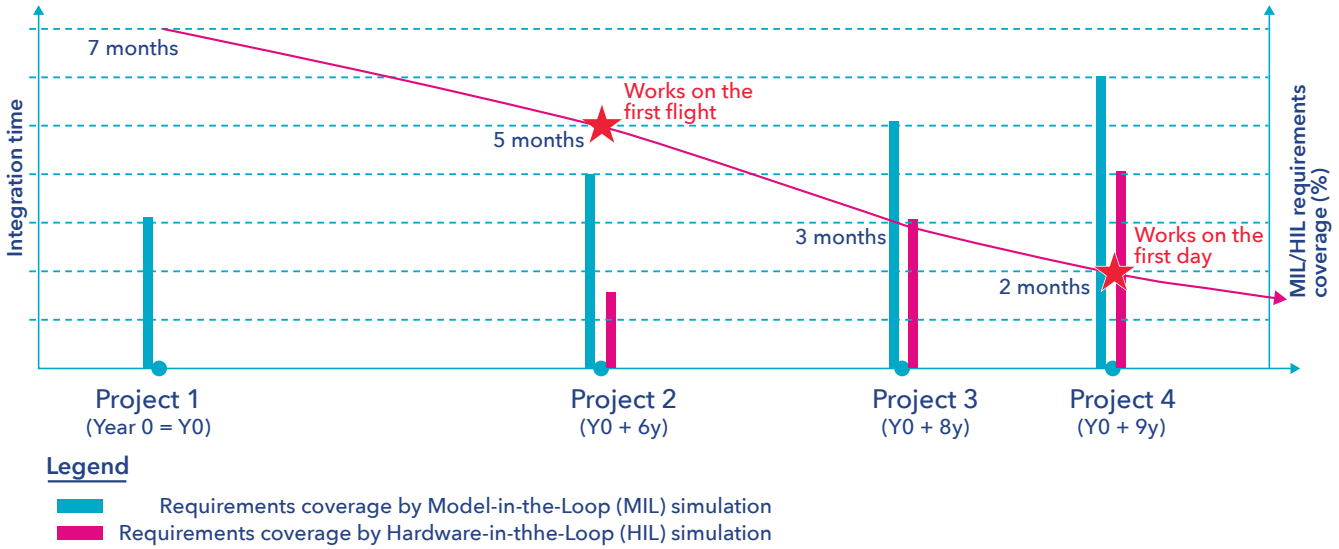


Figure 7. Evolution of integration and flight test qualification time over 4 projects in 9 years

simulation of this model did not raise any error; thus it contributes to increase our confidence in our system definition.

Feedback from the frontline

While this paper focuses on specific use-cases to illustrate the approach through concrete examples, the overall intention is to incorporate simulation activities in the daily system engineering work and to make it the new normal. According to return of experience from some company businesses already mature in this field, there is a measured project performance improvement when adopting simulation as the foundations for any design decision. Figure 7 shows the correlation between the amount of system requirements covered by analytical models and the integration and flight tests qualification effort.

Figure 7 also highlights an increase in the system quality with a drastically reduced technical debt. But this comes at the price of a multi-year journey transforming the organization, the culture, the skills, the processes, and the tools. Finally, it is worth to note higher benefits happen when the engineering process is transformed instead of optimizing locally. In this project example, a large part of the benefits come from the continuous reuse of test cases built at the early stage of the development lifecycle down to the ground field tests.

CONCLUSION AND PERSPECTIVES

The use of simulation to support architecture and detailed design activities is an efficient means to improve quality, cost and planning while reducing risks. It is rare enough to be noticed as most of the time these three key project objectives cannot be improved together. This often comes at the price of an increase of the project expense on the upstream engineering activities, which is often perceived as a gamble from the project management perspective. However, this is not always true and some other experiments on other projects have demonstrated that the effort spent to specify the system with a simulation model is lower than doing it with textual requirements as a lot of effort is spent trying to identify all the missing cases, all the incoherencies and to make the textual requirements traced, coherent and consistent with the design model. This case-study has demonstrated clear benefits using simulation on top of design model. However, to be efficient, an automation tool is highly recommended as well as clear modeling rules that favor modularity and reusability.

One point of attention is when simulation is used as specification as the simulation can lead to over constrained specifications. Indeed, it is common to see an executable specification describing one nominal or ideal operating point. If the specification model is used as a scoreboard for equivalence testing, the probability of having the real system exactly match the behavior specified that way tends to zero. Hence it is critical as a system engineer to define margins in the expected behavior and to identify parameters' tolerance. The same way, as a simulation engineer it is as important to quantify the uncertainty of simulation parameters and the credibility level granted to simulation results.

Regarding the case study we intend to extend it to demonstrate other use cases. Among others, the most prevalent are:

- To handle the round-tripping. As simulation is used, in the context of this study, as a supporting activity of the architecture definition there is a need to go back and forth between the architecture definition and the simulation. This round-trip can be either simple and straightforward or more complex. The simple case is when simulation is leveraged for architecture verification or for sizing and optimization. In these cases, the simulation request can clearly define what are the model elements of the architecture model to update (verification status, architecture parameters' value for instance). Hence, the update of these elements does not raise any issue. The complex case is when the simulation process raises some evolution requests on the architecture (ambiguities to resolve, missing data, incomplete definition...). This situation must be handled via a dedicated process to guarantee the integrity of the overall architecture definition. This process, when defined, can then be tooled-up to make easier the concrete update of the architecture model from simulation models.
- To perform simulation-based dependability analysis in accordance with both the architecture model and the Arcadia method.

Finally, this work shall be continued to confirm applicability and define good practices for use of simulation in different contexts, such as simulation for architecture optimization from the orientation phase down to the detailed design, simulation in product line engineering (PLE) contexts, and simulation in agile contexts. ■

REFERENCES

- Bateman, S., R. L. Mandryk, C. Gutwin, A. Genest, D. McDine, and C. Brooks. 2010. "Useful Junk? The Effects of Visual Embellishment on Comprehension and Memorability of Charts." In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, US-GA, 10-15 April.
- Bonnet S, J.-L. Voirin, V. Normand, and D. Exertier. 2015. "Implementing the MBSE Cultural Change: Organization, Coaching and Lessons Learned." Paper presented at the 25th Annual International Symposium of INCOSE, Seattle, US-WA, 13-161 July.
- Bonnet S., J.-L. Voirin, and J. Navas. 2019. "Augmenting Requirements with Models to Improve the Articulation Between System Engineering Levels and Optimize V&V Practices." Paper presented at the 29th Annual International Symposium of INCOSE, Orlando, US-FL, 20-25 July.
- Camus, J.L., C. Haudebourg, M. Schlickling, and J. Barrho. 2016. "Data Flow Model Coverage Analysis: Principles and Practice." Paper presented at the 8th European Congress on Embedded Real Time Software and Systems (ERTS), Toulouse, FR, 27-29 January. hal-01262411f.
- Capella. 2021b. Equivalences and Differences between SysML and Arcadia/Capella. https://www.eclipse.org/capella/arcadia_capella_sysml_tool.html.
- Capella. 2022a. https://www.eclipse.org/capella/.
- Cloutier R., G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone. 2010. "The Concept of Reference Architectures." *Systems Engineering* 13 (1): 14-27.
- de Ferluc, R., F. Capogna, G. Garcia, O. Rigaud, D. Demarquilly, and L. Bitetti. 2018. "Model Based Safety Assessment (MBSA) in the Space Domain with Capella Open-Source Tool." Paper presented at Congrès Lambda Mu 21, " Maîtrise des risques et transformation numérique : opportunités et menaces," Reims, FR. 16-18 October. hal-02064930.
- Filho, P. S. O. 2018. "The growing level of aircraft systems complexity and software investigation." Embraer Air Safety Department.
- Fleischer, D., M. Beine, and U. Eisemann. 2009. "Applying Model-Based Design and Automatic Production Code Generation to Safety-Critical System Development, " *SAE International Journal of Passenger Cars – Electronic and Electrical Systems* 2 (1): 240-248.

- Garro, A. S., and A. Tundis. 2012. "Modeling and Simulation for System Reliability Analysis: The RAMSAS Method." Paper presented at the IEEE 7th International Conference on System of Systems Engineering (SoSE), Genova, IT, 16-19 July.
- Iandoli, L., L. Piantedosi, A. Salado, and G. Zollo. 2018. "Elegance as Complexity Reduction in Systems Design." *Complexity* 2018: 5987249:1-5987249, 10 pages.
- INCOSE. 2014. "A World in Motion – Systems Engineering Vision 2025."
- ISO/IEC/IEEE 42010. 2011. "Systems and software engineering — Architecture description."
- ISO/IEC/IEEE 42010. 2011. "Systems and software engineering — Architecture description."
- Langlois B., and J. Barata. 2017. "Extensibility of Capella with Capella Studio." Eclipse Newsletter. https://www.eclipse.org/community/eclipse_newsletter/2017/december/article4.php.
- Lorenzo, B., R. Ferluc, D. Mailland, G. Gregoris, and F. Capogna. 2019. "Model Based Approach for RAMS Analyses in the Space Domain with Capella Open-Source Tool." Paper presented at the International Symposium on Model-Based Safety and Assessment, Thessaloniki, Greece, GR, 16-18 October. 10.1007/978-3-030-32872-6_2.
- Navas J., S. Bonnet, J.-L. Voirin, and G. Journaux. 2020. "Models as Enablers of Agility in Complex Systems Engineering." Paper presented at the 30th Annual International Symposium of INCOSE, Virtual, 20-22 July.
- Navas, J., S. Paul, and S. Bonnet. 2019. "Towards a Model-Based Approach to Systems and Cybersecurity Co-engineering." Paper presented at the 29th Annual International Symposium of INCOSE, Orlando, US-FL, 20-25 July.
- Oster, C., M. Kaiser, J. Kruse, J. Wade, and R. Cloutier. 2016. "Applying Composable Architectures to the Design and Development of a Product Line of Complex Systems." *Systems Engineering* 19 (6): 522-534.
- Pineda, C. S., and X. Wang. 2011. "A Study of the Characteristics of Behaviour Driven Development." Paper presented at the 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Oulu, FI, 30 August – 2 September, pp. 383-387, doi: 10.1109/SEAA.2011.76.
- Stecklein, J. M., J. B. Dabney, B. N. Dick, B. R. Haskins, R. Lovell, and G. Moroney. 2004. "Error Cost Escalation Through the Project Life Cycle." Paper presented at the 14th Annual International Symposium of INCOSE, Toulouse, FR, 20-24 June.
- Voirin J.-L. 2017. "Model-based System and Architecture Engineering with the Arcadia Method. " UK-London & Elsevier, Oxford: ISTE Press.
- Voirin J.-L., S. Bonnet, V. Normand, and D. Exertier. 2015. "From Initial Investigations up to Large-Scale Rollout of an MBSE Method and its Supporting Workbench: the Thales Experience." Paper presented at the 25th Annual International Symposium of INCOSE, Seattle, US-WA, 13-161 July.
- Wippler J. L. 2018. "Une approche paradigmatique de la conception architecturale des systèmes artificiels complexes." Université Paris-Saclay, FR-Paris.

ABOUT THE AUTHORS

Juan Navas is a systems architect with +12-years' experience on performing and implementing systems engineering practices in multiple organizations. He oversees the Thales corporate modelling, and simulation coaching team and dedicates most of his time to training and other consulting activities worldwide, for Thales and other organizations. He accompanies systems engineering managers and systems architects implement MBSE approaches on agile operational projects, helping them define their engineering schemes, objectives, and guidelines. He holds a PhD on embedded software engineering (Brest, France), a MSc degree on control and computer science from MINES ParisTech (Paris, France), and electronics and electrical engineering degrees from Universidad de Los Andes (Bogota, Colombia).

Pierre Nowodzienski is a systems architect with 10-years experience on performing and implementing systems engineering practices in multiple organizations. He is a Thales corporate MBSE coach and dedicates most of his time to training and other consulting activities worldwide, for Thales and other organizations. He accompanies systems engineering managers and systems architects implement MBSE and simulation-based engineering approaches on agile operational projects, helping them define their engineering schemes, objectives, and guidelines. He holds a MSc degree in mechatronics and complex systems from ENSEA (Paris, France).

Hecht and Chen continued from page 39

ABOUT THE AUTHORS

Myron Hecht is a senior project leader at The Aerospace Corporation where he specializes in model-based systems engineering (MBSE) and in reliability, for complex weapons systems. He also is a consultant to the Nuclear Regulatory Commission and a lecturer at the UCLA School of Engineering and Applied Sciences. His current research is on application of MBSE to reliability, availability, and safety analysis. He has served on standards committees for reliability (IEEE 1332 and GEIA STD 009), computers in nuclear power plants (IEEE 7-4.3.2), software in avionics systems (RTCA DO 178C and 278A), and model-based safety and reliability for the Object Management Group (OMG). He is an author of more than 100 refereed publications in reliability, safety, products liability, and systems engineering. Myron holds BS (chemistry), MS (nuclear engineering), MBA, and JD degrees all from UCLA.

Jaron Chen is a member of the technical staff at The Aerospace Corporation. He works in in the areas of model-based systems engineering (MBSE), machine learning, software tools development, discrete event simulation, and integration of modeling techniques in support of space and ground communications systems. He holds an MS from Carnegie Mellon University and a BS from the University of California Irvine.

System Verification and Validation Approach Using the MagicGrid Framework

Aurelijus Morkevicius, aurelijus.morkevicius@3ds.com; **Aiste Aleksandraviciene,** aiste.aleksandraviciene@3ds.com; and **Zilvinas Strolia,** zilvinas.strolia@3ds.com
Copyright ©2022 by Aurelijus Morkevicius, Aiste Aleksandraviciene, and Zilvinas Strolia. Permission granted to INCOSE to publish and use.

ABSTRACT
The ongoing transformation in the industry from a document-based systems engineering to a model-based systems engineering approach reveals a need for new methods of verifying and validating systems. Traditional methods of testing the actual system are getting more and more expensive. A model-based environment could significantly reduce testing and, most importantly, verification and validation processes costs. It allows testing on the system model by applying various techniques, such as simulation, analysis, review, mock-ups, etc. There are, however, very few approaches today detailing how verification and validation of the entire system (taking into count its components and subsystems) could be performed. This paper proposes an approach to perform verification and validation of a system using system models developed with Systems Modeling Language (SysML) and in accordance with the MagicGrid (formerly known as MBSE Grid) framework. The approach covers system testing activities beginning with verification of the lowest modeled system elements against system requirements and finishing with validation of the system as a whole, against stakeholder needs.

INTRODUCTION

Verification and validation (V&V) are independent processes. The purpose of verification is to provide objective evidence that a system or system element fulfills its specified requirements and characteristics. The purpose of validation is to provide objective evidence that the system, when in use, fulfills its business or mission and stakeholder requirements, achieving its intended use in its intended operating environment. Both are used together and are critical components of system testing (ISO, 2015).

Testing the actual system is expensive, not only where testing to realistic conditions cannot be achieved, but also when it is not cost-effective. Model-based systems engineering (MBSE) promises a more effective way to show theoretical compliance. It allows testing to be performed using simulation on models or mock-ups (instead of actual/physical system elements) under defined conditions (INCOSE, 2015). Although it may sound promising, having a system architecture in the model-based environment does not automatically test a system. It is a common misunderstanding in systems engineering, that Systems Modeling Language (SysML) is enough to fulfill the promise of MBSE. Silingas et al. (2009) states that the modeling language is just the language and must be combined with a methodology to be useful. In the MBSE environment, testing activities such as verification and validation depend heavily on the methodology used to develop system architecture. For this reason, finding the right approach for the specific environment is very difficult, or in the best case requires tailoring and customization.

In this paper, a new approach for model-based V&V is proposed. The approach covers system testing activities starting with verification of the lowest modeled system elements against system requirements and going up to the system as a whole validation against stakeholder needs. The approach is aligned with the SysML language and MagicGrid (formerly known as MBSE Grid) framework. The approach extends the framework by introducing the V&V pillar to existing ones and demonstrating interrelationships between them. A thorough case study on the vehicle model is presented to prove the usefulness of the proposed approach in the overall lifecycle of the system engineering.

This paper is structured as follows: in Section 2, related works are analyzed; in Section 3, the proposed approach is presented; in Section 4, application of the proposed approach is described; and in Section 5, the achieved results, conclusions, and future work directions are indicated.

RELATED WORKS

Hazle et al. (2020) analyze the literature on verification and validation for systems, software and requirements engineering.

			Pillar				
			Requirements	Structure	Behavior	Parameters	Safety & Reliability
Domain	Problem	Black Box	Stakeholder Needs	System Context	Use Cases	Measures of Effectiveness (MoEs)	Conceptual and Functional Failure Mode & Effects Analysis (FMEA)
		White Box		Conceptual Subsystems	Functional Analysis	MoEs for Subsystems	Conceptual Subsystems (FMEA)
	Solution		System Requirements	System Structure	System Behavior	System Parameters	System Safety & Reliability (S&R)
			Subsystem Requirements	Subsystem Structure	Subsystem Behavior	Subsystem Parameters	Subsystem S&R
			Component Requirements	Component Structure	Component Behavior	Component Parameters	Component S&R
	Implementation		Implementation Requirements				

Figure 1. MagicGrid framework

The paper examines their relevance to SysML models as used in MBSE. The authors, using the findings of their literature review, their experience, and the comments of the MBSE Working Group (INCOSE UK MBSE WG, 2019), have compiled a list of the techniques and principles they believe should be employed for an effective and robust approach to model V&V. One of the core principles is the importance of simulation and parametric solving. It is often easier to validate the model behavior via simulation than through the inspection of descriptive views (Debbabie et al., 2010). While applying simulation, the advantage of using an MBSE approach rather than a separate specification and simulation is that since all the engineering artefacts are within the same model, it allows traceability from the stakeholder requirements, through to the elements being simulated (Stevenson et al. 2018), thus improving both verification and impact analysis.

Though the V&V methods and MBSE methodologies can be analyzed separately, the closest comparison to our work would be utilizing the combination of both the V&V approach and the MBSE methodology. This way equivalent works can be compared directly and not overcomplicate the research. The following MBSE methodologies that support V&V activities have been analyzed: object-oriented systems-engineering method (OOSEM), IBM Rational Harmony for systems engineering, and the UML testing profile.

OOSEM. In this methodology, the validate and verify system activity verifies that the system design satisfies the system requirements and then validates that those requirements meet the stakeholder needs. For this, verification plans, procedures, and methods are developed. The primary inputs to the development of the test cases and associated verification procedures are system-level use cases, scenarios, and associated requirements. The verification system can be modeled using the same activities and artifacts described earlier for modeling the operational system. The requirements management database is updated during this activity to trace the system requirements and design information to the system verification methods, test cases, and results (INCOSE, 2015).

OOSEM provides a clear process of which steps need to be carried out in performing V&V; however, it defines no means of achieving it in practice.

IBM Rational Harmony for systems engineering. The IBM methodology covers a verification part related to real-time embedded systems and software. In a model-driven system development environment, the key artifact of the hand-off from systems engineering to subsystem development is executable models. The Harmony/SE Deskbook recommends an interactive verification using model execution, including model animation, and a visual comparison of the "as-is" behavior regarding the expected behavior. Integration test scenarios will be part of each composed subsystem hand-off package. These test scenarios can be used to verify a developed subsystem against the requirements (Hoffman, 2015).

The Harmony approach turns out to be very much tool-oriented and the exact modeling language constructs used to perform V&V are not clearly described. Moreover, the approach is embedded systems-oriented and does not describe how testing could be carried out in the higher levels of system hierarchy and with different types of systems.

UML testing profile (UTP). UTP is a part of the UML ecosystem and as such, it can be combined with other profiles of that ecosystem to associate test-related artifacts with other relevant system artifacts, for example, requirements, risks, use cases, business processes, system specifications, etc. This enables requirements engineers, system engineers and test engineers to bridge the communication gap among different engineering disciplines (OMG, 2019b). UTP provides a set of concepts to describe test planning, test architecture, test behavior, and data. In contrast, SysML has the only concept used for testing called 'test case.' In SysML, a test case is defined as "a method for verifying a requirement is satisfied" (OMG, 2019a). It has a single return parameter named verdict which is typed by the enumeration 'verdict kind.' It is important to note this is consistent with the UML testing profile.

Though UTP is a well-defined language for capturing testing concepts, it is not a methodology, nor a method. Application of UTP depends on the methodology used for systems engineering in the context of the specific project or organization.

Analysis of related works helps us to understand how V&V is addressed today in the model-based engineering environments. It also helps to identify core principles used in the context of MBSE, the importance of the model execution, and the gaps existing today that could be solved by our suggested approach.

SUGGESTED APPROACH

This section introduces an approach for performing system verification and

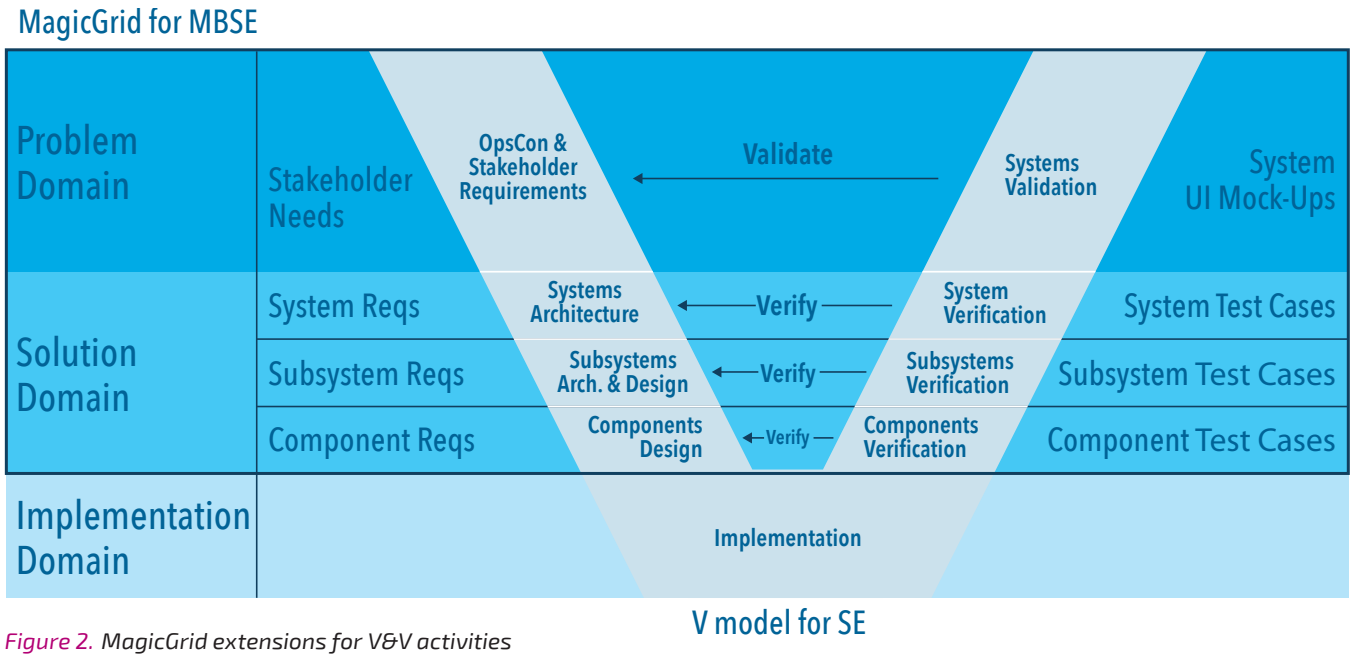


Figure 2. MagicGrid extensions for V&V activities

validation activities using SysML models developed following the modeling workflow defined by the MagicGrid framework. Relevant extensions to the framework for supporting V&V are proposed.

Introduction to MagicGrid

MagicGrid is a pure SysML-based framework for MBSE (Mazeika et al., 2016). The structure of the framework can be depicted as a 2-D grid, displayed in Figure 1.

The rows of the grid represent different layers of abstraction, also referred to as domains: *Problem, Solution*, and *Implementation*. Every organization must deal with them, unless the system they develop is very small and simple, for example, some mechanical component without any embedded software. In the case of complex systems development, they are inevitable (Morkevicius et al., 2017).

The columns of the grid stand for different aspects of the SysML model. Also referred to as four pillars SysML (Friedenthal et al., 2008), these are *Requirements, Structure, Behavior*, and *Parametrics*, and the recently added *Safety & Reliability pillar*.

A cell at the intersection of a particular row and column represents the view, which determines what SysML diagrams and elements should be utilized to capture information when visiting that particular cell. The order of visiting the cells is defined by the modeling workflow of the MagicGrid framework. It is based on the best practices of systems enginering and technical processes defined in ISO 15288 (Morkevicius et al., 2020).

MagicGrid Extensions for V&V

So far, the MagicGrid framework has been applicable to the systems engineering activities that belong to the left side of the V model, to support the system development all the way from stakeholder needs elicitation to the high-level (logical) design of system components. However, it does not provide much information on how to perform V&V activities.

Figure 2 illustrates how the framework can be extended to support early V&V of the system. This kind of V&V is carried out before starting the implementation of the system and can be very useful for detecting issues that normally are not discovered until the testing of the physical system begins. The early V&V is applicable not only when testing the system in realistic conditions cannot be achieved (for example, spacecraft, satellite), but also when it is not cost-effective.

Since the physical system does not exist during the early V&V, its solution domain model built by following the MagicGrid framework serves as input to both activities. The solution domain model specifies a precise logical architecture and high-level (logical) design of the selected system configuration, including user interface (UI) mock-ups. Therefore, if it passes the V&V against the system requirements and stakeholder needs, the system can be considered verified and validated as well.

Verification. The verification approach is bottom-up: it is performed gradually from the components' level to the system level (in this paper, the components' level is considered the lowest level of detail containing atomic elements of system structure).

The first iteration of verification can be performed as soon as component models are completed. Once these models pass the verification against the component level requirements, they can be integrated at the subsystem level. It is important to note that various design solutions can be proposed for each component; more than one configuration (or candidate solution) model at the subsystem level can be produced. The subsystem models can then be verified against the subsystem level requirements. After these models pass the verification, they can be integrated at the system level. As with the subsystem level, more than one configuration of the whole system can be built, which are subsequently verified against the system level requirements. To assess the alternative configurations and choose the preferred architecture for the physical implementation at each level of detail, a trade-off analysis is performed (Morkevicius at al., 2021).

The first iteration of verification can be performed as soon as at least one of the component models is completed. Once these models pass the verification against the component level requirements, they can be integrated at the subsystem level. It is important to note that various design solutions can be proposed for each component; more than one configuration (or candidate solution) model at the subsystem level can be produced. The subsystem models can then be verified against the subsystem level requirements. After these models pass the verification, they can be integrated at the system level. As with the subsystem level, more than one configuration of the whole system can be built, which are subsequently verified against the system level requirements. To assess the alternative configurations and choose the preferred architecture for the physical implementation at each lev-

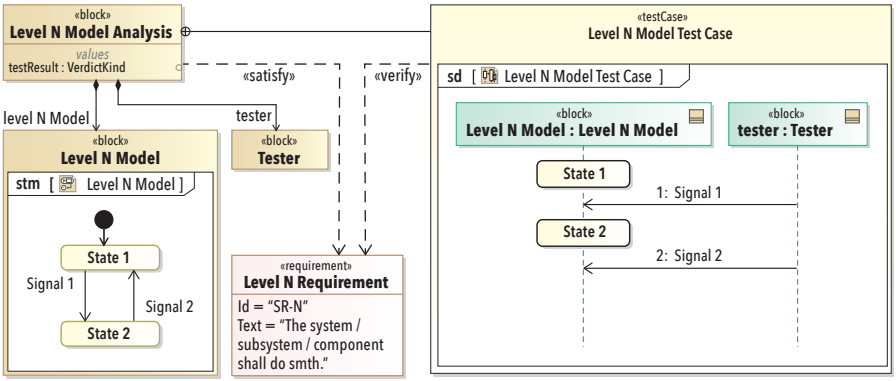


Figure 3. Typical model to perform early verification using MagicGrid

el of detail, a tradeoff analysis is performed (Morkevicius et al., 2021).

As defined by MagicGrid, the models at different levels of system hierarchy are usually created by different engineering teams or even organizations. Therefore, the verification of components and subsystems may be performed in isolation from each other.

The verification at each level of detail is performed by utilizing test cases, as a single test case enables users to check whether the related model or its fragment satisfies the relevant requirement. This information is captured as the verdict of the test case.

A test case can be captured in the model as SysML sequence, activity, or state machine. Each test case must be related to the relevant system, subsystem, or component level requirement by utilizing the «verify» relationship. Early verification of the system can be performed automatically by executing its models.

To perform early verification of the system, the same verification approach can be applied at each level of system hierarchy within the solution domain model. As Figure 3 depicts, a typical SysML model for performing early verification includes:

- **Model of the system/subsystem/component to verify.** It is represented here as the *Level N Model* block with the state machine as its classifier behavior. In the block name here and elsewhere in the Figure 3, *Level N* can be replaced with *system, subsystem*, or *component*.
- **Requirement against which the system/subsystem/component must be verified.** It is represented here as the *Level N Requirement* with testing specified as its verification method.
- **Analysis context.** It is represented here as the *Level N Model Analysis* block. The analysis context block is responsible for orchestrating the execution of the test case and the behavioral model at the particular level of system hierarchy. It includes: (i) the model to verify/and (ii) the *Tester* block which represents the person who performs the verification by sending external trigger(s) to the appropriate model element. As shown in Figure 3, that external trigger can be a signal provided by a message as part of the executable test case.
- **Test Case to capture the steps for verification.** It is represented here as the *Level N Model Test Case* test case captured in the model as interaction and displayed using the infrastructure of the SysML sequence diagram. As the test case is owned by the analysis context block, the lifelines of the related sequence diagram represent both blocks that take part in this analysis block: (i) the model to verify; and (ii) the *Tester* block. The test case has a parameter named *verdict* (it appears in the following figure, within the related analysis block), which is set to *pass* if the model passes the steps of the test case, or to *fail* if it does not. When the *verdict* is *pass*, the model can be considered as satisfying the related requirement.

Validation. Once the system configuration model is verified, it is time to validate it against stakeholder needs. As part of system architecture, UI mock-ups can be used for this. As Figure 4 shows, the UI mock-ups (or UI prototype) can be produced by utilizing the UI prototyping profile which

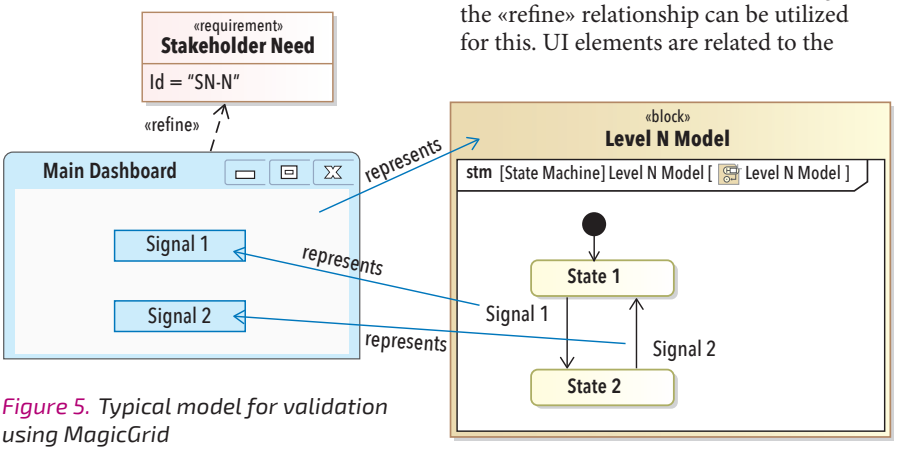


Figure 5. Typical model for validation using MagicGrid

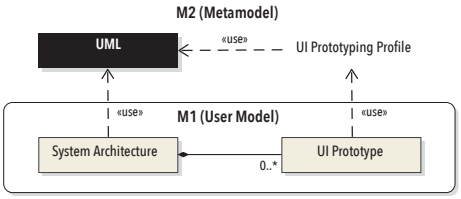


Figure 4. UI prototype modeling

(in addition to the SysML profile) extends the UML 2 metamodel to support stereotypes for UI prototype modeling (Silingas et al., 2010). Moreover, the UI prototyping profile enables users to integrate the UI mock-ups into the system architecture model, that is, relate them to the SysML model elements capturing relevant concepts of system architecture. Standard SysML / UML relationships, such as realization or "trace," can be used for this.

The UI prototyping profile is implementation-independent and includes a minimal set of UI elements, along with their properties for the most common UI prototyping needs. These elements extend certain UML meta classes, such as component or class, and are grouped into separate packages by type, as shown in Figure 4 (Silingas et al., 2010). Just like UML or SysML elements, UI mock-up elements can be semantically related to each other. Element representations are taken from the Java Swing external library. A user interface modeling diagram to display them, is based on the UI prototyping profile and created using a typical workflow for creating the domain-specific modeling environment as presented in Silingas et al. (2009).

UI prototyping can be applied at various stages of an engineering project. For example, it can be used to facilitate the problem domain analysis to communicate with stakeholders. When applying the MagicGrid, UI mock-ups can be used to validate the system architecture. UI mock-ups related to the elements of logical system architecture within the solution domain model must be related to the relevant stakeholder needs, as well. As shown in Figure 5, the «refine» relationship can be utilized for this. UI elements are related to the

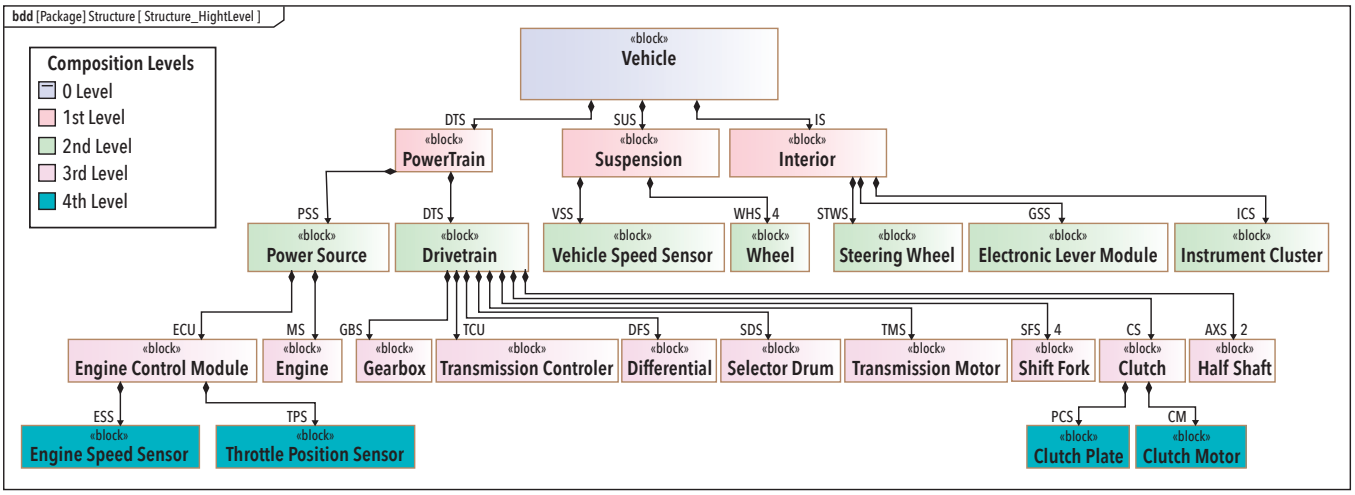


Figure 6. Car structure displayed within a SysML block definition diagram

architecture elements using the 'represents' metaproperty.

CASE STUDY

This part of the paper presents a case study example that illustrates the proposed approach. The modern car is a sophisticated system that holds many internal system elements that intricately interact with each other. The integration of different parts is a vital activity; hence this system can be recognized as a suitable example to demonstrate the proposed system V&V method.

In the case study example, the car model is developed using SysML language and MagicGrid method and most of the model creation activities are not disclosed. However, it is still important to understand the structure of presented system of interest (SoI) and its internal components. For this reason, Figure 6 is used to display the car structure.

Figure 6 shows an SoI structure that is divided into five composition levels, beginning from level zero that represents the car as one unit, and ending at level four, that characterizes the components of SoI. Due to the extent of the modern car systems and scope associated with them, it is not possible to disclose all required systems verification activities. For this reason, this case study example analyzes a small subset of internal car systems.

System Verification at Lowest Composition Level

The application of the presented approach is started from the second level of decomposition, depicted in Figure 6. At this level, the drivetrain system is chosen to be the target for system verification. This vehicle system is an elaborate system on its own and is composed of multiple internal systems that could require undergoing their own verification processes. In this paper, the assumption is made that internal subsystems of the drivetrain passed their system verification activities on the third and fourth levels of composition. For this reason, our main attention is directed to the verification of the drivetrain system.

For the drivetrain system, the transmission controller is one of the most important elements. Nowadays, the drivetrain component is implemented with integrated circuits, where software code is responsible for the correct application of this part of the vehicle. As reading and understanding software requires specialized skills and knowledge, it is a good idea to abstract it for a wider audience and retention of knowledge. The case study model accomplishes this by analyzing transmission controller behavior in the form of a SysML state machine diagram, displayed in Figure 7 part A. Figure 7 part B shows a SysML state machine diagram that describes how another drivetrain component, clutch, behaves. Each of the drivetrain system elements (Figure 6) potentially could have a similar behavior description but this case study example overlooks that due to the extent related to their analysis. Verification concentrates on the clutch and transmission controller systems.

For our purposes, the vehicle drivetrain is identified as an automated manual type. This system is a mix of automatic and manual transmission types and has the characteristics of both. For example, like a manual transmission, the automated manual transmission has a clutch component that needs to be engaged during the change of gears. However, in this type of transmission, clutch engagement is done

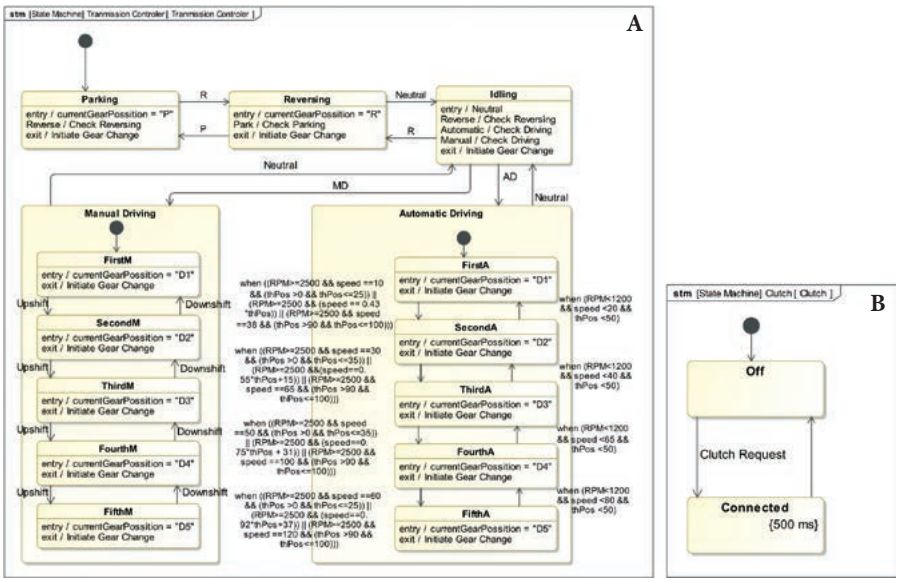


Figure 7 A. Transmission controller behavior description within a SysML state machine diagram; B. Clutch behavior description within a SysML state machine diagram

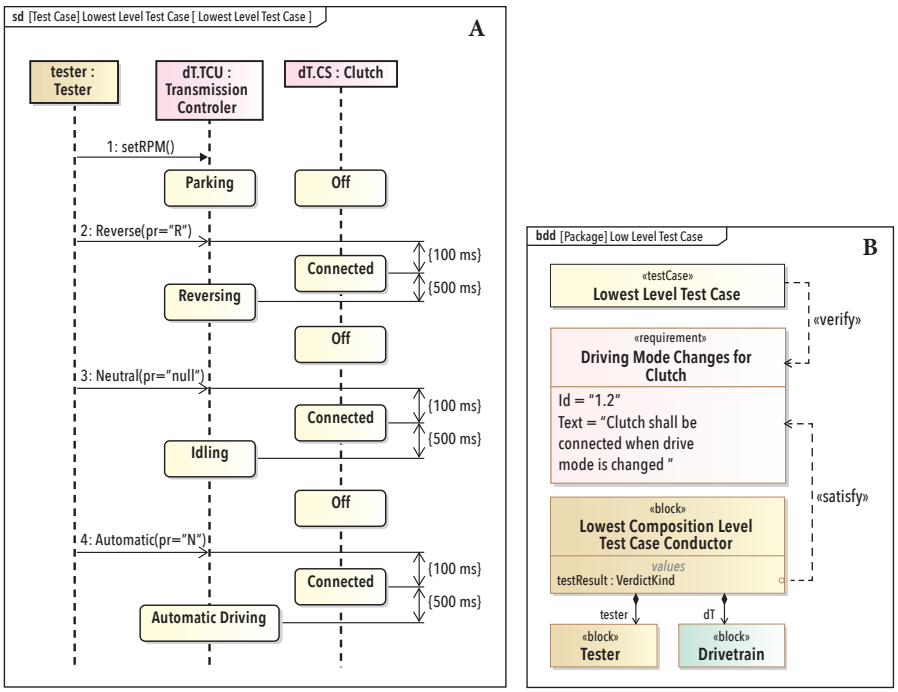


Figure 8. A. Second composition level test case description with SysML sequence diagram; B. Test case linkage to requirement

not by the driver but by the electrical motor component of the drivetrain system. For this reason, a verified system is required to ensure that the control logic of the transmission controller adheres to the correct clutch operation in a timed manner.

To verify system behavior integration, the proposed approach suggests creating executable test cases. The test case should define *a priori* system behavior. Failure of the test case could potentially imply incomplete system behavior definition and at the same time improper system verification at this composition level. Figure 8 part A displays one the possible test case used for the verification. Here, the test case checks that driving mode changes on the transmission controller system result in correct clutch system behavior. To achieve this, the tester sends an external stimulus "reverse" in a message to the transmission controller element. Upon acceptance of the message, the transmission controller element changes its states from "parking" to "reversing." The state change on the transmission controller system needs to be propagated to the clutch system in order to connect it mechanically to the gearbox system and allow a drive mode change for the drivetrain system. As consequence, the clutch should also change its state from "off" to "connected." Failure of the lifeline represented element to be in the required state denotes the failure of the test case. This is verified with a state invariant element in the case study system model. Note that sequence diagram usage for the test case definition is not the only available possibility. Other model elements could be used as well, although this would require different test case checking techniques.

One potential limitation of this approach is the need to manually define test cases. Due to the many available driving mode change permutations in the system model, the presented test case is reduced to a limited set of driving mode changes (Figure 8 part A). However, for complete verification of the drivetrain system, the model should pursue all of them. Though we consider this to be out of scope of our proposed approach and we do not explore available automatic test case generation methods from SysML models.

Figure 8 part B introduces a verify relation between a requirement and a test to show a logical and traceable link, in accordance to the proposed approach. As the proposed method is of executable type, a block is required for orchestrating model execution. In Figure 8 part B, the block "lowest level test case conductor" achieves this by associating the verified level SoI (drivetrain) and external stimulus (tester) blocks.

The verdict of the test case verification should be retained. This is achieved by using a value property element (testResult) in the presented system model. If it is necessary to run multiple test cases for the same SoI, several value properties should be used for storing each test case verification outcome. Lastly, multiple test case verification results could have different verdict values and there could be a need for their arbitration, hence the satisfy relationship between the value property and requirement is presented as an option.

Figure 9 shows the usage of the instance table to display multiple test case execution attempts in the form of an instance element. The "fail" value of the value property "testResult" specifies a test case violation and "pass" marks a successful run of it. In our approach, "pass" signifies correct system verification at this composition level.

System Verification at Intermediate Composition Level

At the intermediate level of an SoI structure, the case study example targets the powertrain system, composed of two subsystems: power source and drivetrain (Figure 8). Here, the proposed verification approach is repeated in the same way as in the lowest system composition level. First of all, a test case is defined using a SysML sequence diagram (Figure 10 part A). This test case checks that the drivetrain system changes driving mode when the parameters change on the engine system. Specifically, the drivetrain system part transmission controller should monitor the engine throttle position and engine speed sensors values. If monitored values comply with the predefined drive mode change patterns specified within the system requirements (Figure 10 part B), then the drive mode change should occur for the transmission controller system. In the test case, the correct driving mode selection is achieved with the state invariant element. Failure of the state invariant results in the failure of the

#	△ Name	testResult
1	lowest Level Test Case at 2021.10.14 16.30	fail
2	lowest Level Test Case at 2021.10.15 11.19	fail
3	lowest Level Test Case at 2021.10.21 13.03	fail
4	lowest Level Test Cases at 2021.10.21 13.23	pass

Figure 9. Multiple test case execution results displayed in instance table

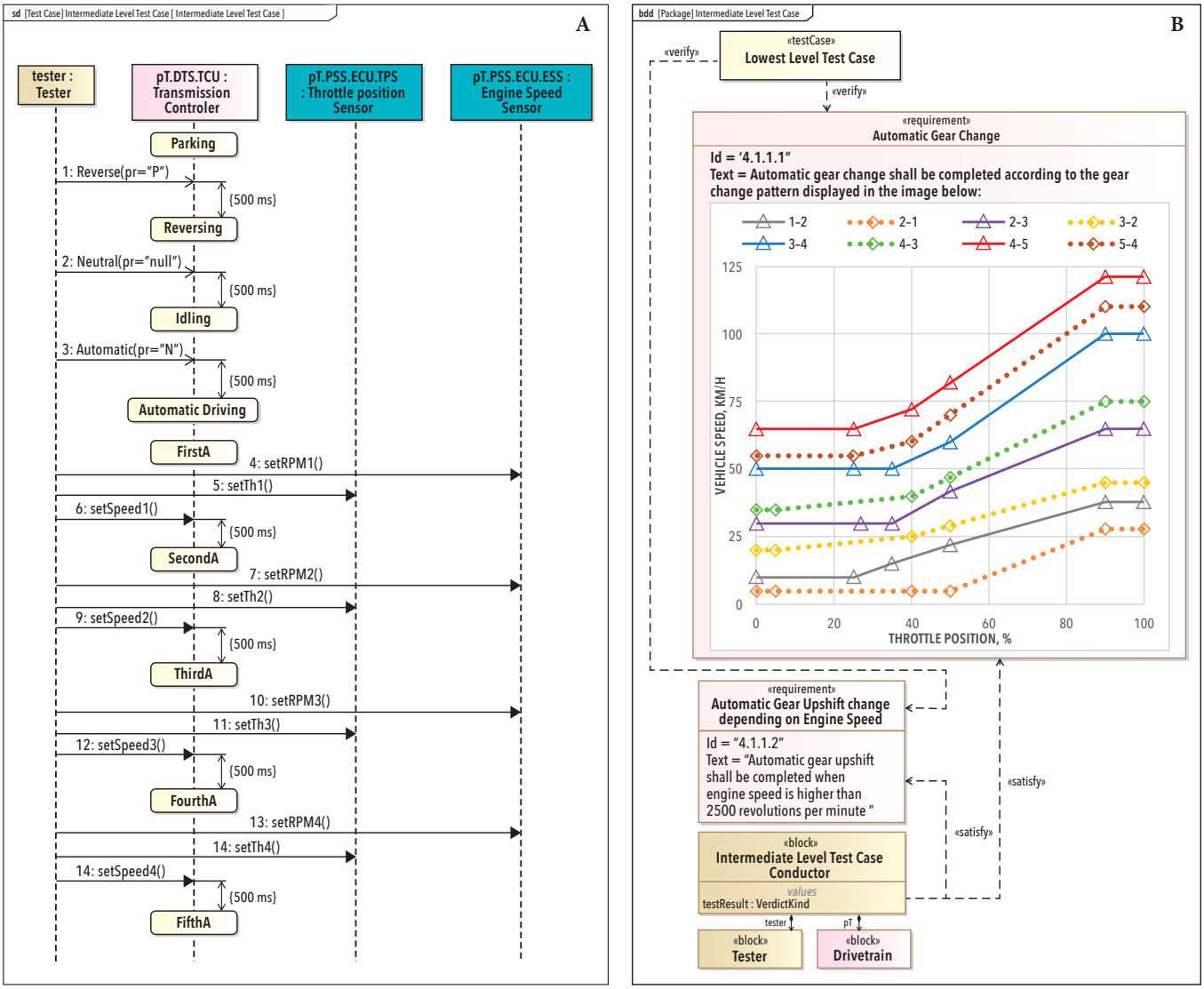


Figure 10. A. Intermediate composition level test case description within SysML sequence diagram; B. Intermediate composition level test case linkage to requirement

test case. Engine parameter changes are invoked by the external tester that sends synchronous messages to the verified system elements. Upon acceptance of the message, the receiver calls a designated operation that has a method description, which invokes the required variable change in the model and with operations we can reuse repeating system variable change patterns.

In accordance with the presented approach, requirements are connected to the test case element for traceability purposes and a block (intermediate level test case conductor) is introduced for the test case and SoI behavior orchestration. For the execution results storage, a variable is required, thus in a SysML model, a value property (Figure 10 Part B) is used. As in the lowest system composition level, the pass of the test case execution denotes successful system behavior verification (Figure 9).

System Verification at the Highest Composition Level

In the last part of the case study example, the highest SoI composition level is explored. At this level, a car is recognized as a single unit and system verification should be concerned with all vehicle systems and their incorporation into one working entity. The system verification procedure is repeated in the same way as before: definition of the test case, connection of the test case to requirements, executable model preparation and execution of it.

At the highest composition level, this case study example explores how a driver-invoked driving mode changes with the electronic lever module result into a gear change for the drivetrain system (Figure 11 Part A). As these two systems are on different vehicle structures, it is important to ensure their behavior synchronization for proper system verification.

For the test case definition, a SysML sequence diagram is used. Here, the state invariant is an element that confirms the test case correctness, because the system incapacity to be in the required state should result in test case failure. As before, related requirements are connected to the test case for traceability purposes and a block responsible for the test case and SoI behavior coordination is introduced (Figure 11 Part B).

Due to the large number of systems associated with modern cars, in this paper it is impossible to display all test cases that are required for complete car system verification. For this reason, the case study sample limits system verification to the successful pass of the test case, displayed in Figure 11 Part A. However, a car system model should cover all required test cases for the proper system verification.

Additionally, UI elements could be

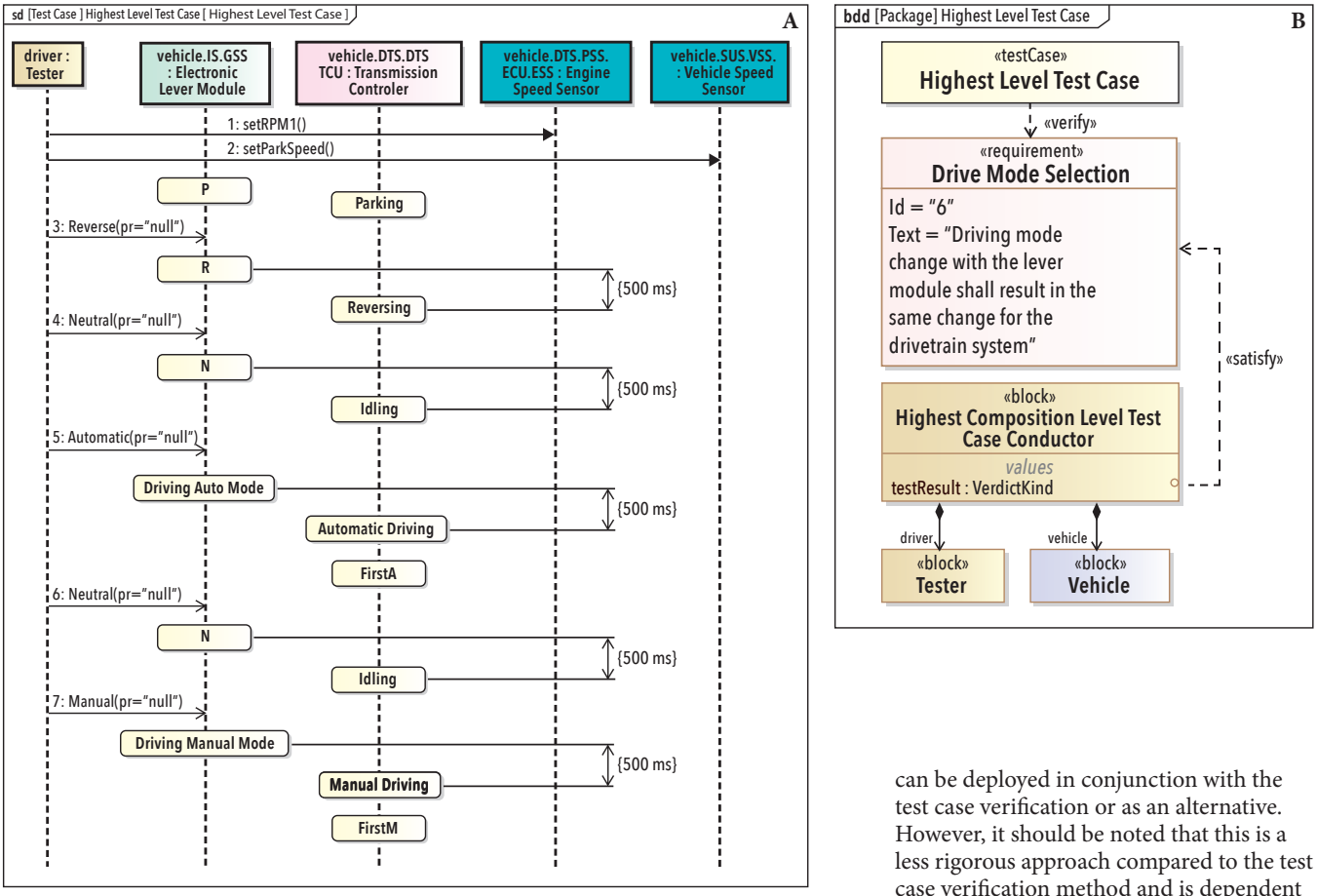


Figure 11. A. Highest composition level test case description with SysML sequence diagram; B. Highest composition level test case linkage to requirement

utilized for coordinating different systems behaviors and as a tool for validating a system model. Figure 12 Parts A and B denote UI usage in the case study example. Figure 12 Part C shows the necessity for displaying the correct gear in the instrument cluster when a driver changes the driving mode with the help of an electronic lever module in the form of a SysML requirement.
The presented UIs are used to abstract and visualize car instrument cluster and electronic lever module systems. A "refine" relationship is employed to link them to the requirement. Since UIs are interactive windows during real-time model execution, they can display system model variables, states, or other element values. Therefore, a model user can employ them as a means for the system model validation. In the case study sample, the failure of displaying the correct drive mode during the model execution signifies denoted stakeholder requirement validation failure.

Lastly, UIs use for systems validation can be deployed in conjunction with the test case verification or as an alternative. However, it should be noted that this is a less rigorous approach compared to the test case verification method and is dependent on the experience of the modeler.

CONCLUSIONS AND FUTURE WORKS

Analysis of related works revealed that there are only a few scientific researchers in the area of V&V in the MBSE environment. There are some MBSE methodologies addressing V&V, such as OOSEM and IBM Harmony; however, none of them clearly, step-by-step, define V&V element relationships to the system architecture and requirements in the different layers of the hierarchy of system elements.

The proposed approach clearly defines the application of SysML to model-based

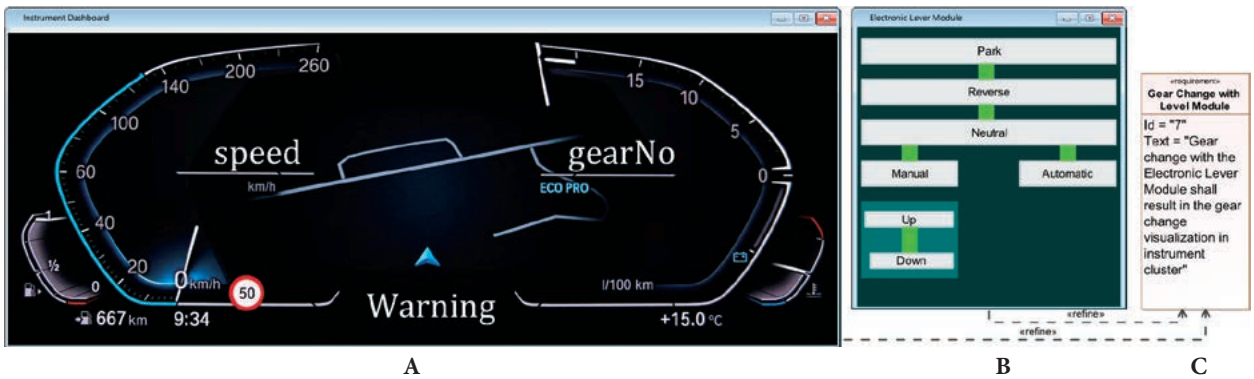


Figure 12. A. GUI for instrument cluster; B. GUI for electronic lever module; C. Requirement stating the need of displaying drive mode in instrument cluster

V&V in the context of the MagicGrid methodology. The applicability of the approach is demonstrated on the real-life vehicle model at every level of the hierarchy of system elements. Moreover, the approach uses pure SysML, without adding additional complexity by extending the standard language. Should it be necessary, organizations applying the approach can extend the proposed approach to support their required terminology.

A modern car (or any other current system) has a substantial complexity associated with it. Manual test case creation could be a laborious task that could potentially lead to inadvertent mistakes in the definition of the test case. Therefore, it is important to address it beforehand when verifying systems using the proposed approach. ■

REFERENCES

- Bankauskaite, J, A. Morkevicius and R. Butleris. 2021. "Model-Based Evaluation of the System of Systems Architectures Used to Perform Trade Studies and Sensitivity Analyses." IEEE Access 9: 114609-114621.
- Debbabi, M, F. Hassaïne, Y. Jarraya, A. Soeanu, and L. Alawneh. 2010. "Verification and Validation in Systems Engineering: Assessing UML/SysML Design Models." Heidelberg, DE: Springer.
- Friedenthal, S, A. Moore, and R. Steiner. 2014. *A Practical Guide to SysML (3rd Edition)*. Amsterdam, NL: Morgan Kaufmann OMG Press.
- Hazle, A., and J.Towers. 2020. "Good Practice in MBSE Model Verification and Validation." INCOSE UK Annual Systems Engineering Conference (ASEC), Virtual.
- Hoffman, H. P. 2011. "Systems Engineering Best Practices with the Rational Solution for Systems and Sofware Engineering." Deskbook, IBM Software Group, viewed 8 November 2021. https://jazz.net/library-content/wp-content/uploads/2020/11/ibm_rational_harmony_deskbook_rel_4.1.pdf.
- INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, San Diego, US-CA.
- INCOSE UK Model-Based Systems Engineering Working Group. 2019. *Model Verification & Validation*, viewed 1 September 2021. https://www.incosewiki.info/Model_Based_Systems_Engineering/index.php?title=Model_Verification_and_Validation.
- ISO. 2015. *Systems and software engineering — System life cycle processes*, (ISO/IEC/IEEE 15288:2015), Geneva, CH.
- Mazeika, D., A. Morkevicius, and A. Aleksandraviciene. 2016. "MBSE Driven Approach for Defining Problem Domain." 11th Systems of Systems Engineering Conference (SoSE), Kongsberg, NO, 12-16 June.
- Morkevicius, A., A. Aleksandraviciene, A. Armonas, and G. Fanmuy. 2020. "Towards a Common Systems Engineering Methodology to Cover a Complete System Development Process." Paper presented at the 31st International Symposium of INCOSE, Virtual, 20-23 July.
- Morkeicius, A., A. Aleksandraviciene, D. Mazeika, L. Bisikirskiene, and Z. Strolia. 2017. "MBSE Grid: A Simplified SysML-Based Approach for Modeling Complex Systems." Paper presented at the 27th Annual International Symposium of INCOSE, Adelaide, AU, 15-20 July.
- OMG 2019a. *Systems Modeling Language (OMG SysML) Version 1.6*, Needham, MA, viewed September 2021. https://www.omg.org/spec/SysML/About-SysML/.
- OMG 2019b. *UML Testing Profile 2 Version 2.1*, Needham, MA, viewed November 2021. https://www.omg.org/spec/UTP2/2.1/About-UTP2/.
- Silingas, D., and R. Butleris. 2009. "Towards Customizing UML Tools for Enterprise Architecture Modeling." IADIS International Conference Information Systems, Barcelona, ES, 25-27 February.
- Silingas, D., S. Pavalkis, R. Vitiutinas, and L. Nemurate. 2010. "Integrating GUI Prototyping into UML Toolkit."
- Silingas, D., R. Vitiutinas, A. Armonas, and L. Nemuraite. 2009. "Domain-Specific Modeling Environment Based on UML Profiles." Conference proceedings of the 15th International Conference on Information and Software Technologies, Kaunas, LT, 23-24 April, pp. 167-177.
- Stevenson, D., K. Vine, and J. Towers. 2018. "Verification and Validation of a New Type of Railway Signal using MBSE and Simulation." INCOSE UK Annual Systems Engineering Conference, Bedfordshire, UK, 20-21 November.

ABOUT THE AUTHORS

Aurelijus Morkevicius. Aurelijus has 17 years of experience in systems and software engineering. Currently, he is a senior manager of industry business consultants for cyber systems engineering in Dassault Systems, CATIA brand. His areas of expertise are model-based systems and software engineering, as well as defense architectures (DoDAF, NAF, UAF). Aurelijus works with companies such as Airbus, BAE Systems, Boeing, Deutsche Bahn, Ford, MITRE, ZF and others. Aurelijus is INCOSE ASEP and OMG Certified UML, Systems Modeling and BPM professional. He is also the co-chairman and the leading architect for the current OMG UAF standard development group. Aurelijus is also the main author of MBSE Grid framework. He is representative of Dassault Systemes in INCOSE and NATO Architecture Capability Team (ACaT). Aurelijus has delivered multiple presentations and tutorials in INCOSE events worldwide, including international workshop and international symposium. Aurelijus received his PhD in information systems engineering from Kaunas University of Technology in 2013. Aurelijus is a professor of practice in the same university where he teaches a course on enterprise architectures, author of multiple articles, and a speaker in multiple conferences.

Aiste Aleksandraviciene. Aiste holds the position of industry business consultant for cyber systems engineering at Dassault Systèmes, CATIA brand. Aiste takes responsibility for educating Dassault Systèmes clients on the three pillars of MBSE: language (SysML), method (MagicGrid), and tool (CATIA Magic software and integrations). She provides trainings, gives tool demonstrations, and participates in providing custom solutions. Aiste also works on training material and writes papers. She is a speaker at multiple MBSE conferences and other public events. Aiste holds a master's degree in information systems engineering from Kaunas University of Technology (Lithuania). Also, she is an OMG® Certified Systems Modeling Professional (OCSMP) and INCOSE Associate Systems Engineering Professional (ASEP).

Zilvinas Strolia. Zilvinas is an industry business consultant with Dassault Systemes and has been in this role since 2015. He is responsible for delivering various modeling solutions and consultancy to CATIA Magic clients. Zilvinas specializes in model-based systems engineering (MBSE) with a unique focus on executable models. Zilvinas holds a Master's degree in economics and Bachelor's degree in electronics engineering from Kaunas University of Technology. He is an OMG Certified Systems Modeling (OCSMP) and INCOSE Associate Systems Engineering Professional (ASEP).

Configuration Management for Model Based Systems Engineering — An Example from the Aerospace Industry

Adriana D'Souza, adriana.dsouza@airbus.com; and **Phanikrishna Thota,** phanikrishna.thota@airbus.com
Copyright ©2022 by Ariana D'Souza and Phanikrishna Thota. Permission granted to INCOSE to publish and use.

ABSTRACT
Model-based systems engineering approach is increasingly used to manage the complexity of modern systems and to reduce costs of their development. In the aerospace industry, modelling and simulation is not only a cost-effective verification and validation strategy where test rigs and flight tests are far more expensive but also is increasingly used in the certification process. Nevertheless, as with any digital artefact, if the models aren't configured and traceability isn't assured, then the models are not of much use. Configuration management comes into play as a key discipline to enable the use and maintenance of the models. This paper explores the use of configuration management for modelling and simulation in an aerospace setting, with a specific example involving landing gear and its surrounding systems.

INTRODUCTION

The main purpose of this paper is to emphasize the rigorous use of configuration management (CM) in all aspects of models and the resulting analyses that will form the basis for future "digital twins" (virtual representations of the physical product that replicate not only the architecture but also the behavior of the real aircraft to the required degree and can be interchanged seamlessly in part or whole of the real aircraft). We do this by presenting two real-world examples where we summarize the lessons learnt. This paper is organized as follows: a brief overview of configuration management and its principles are discussed in the introduction section followed by two real-world examples where configuration management principles have been applied to modelling processes with varying impact on product certification. The paper concludes on the usage of CM in the future where modelling and simulation will be used more extensively to perform verification & validation (V&V) and certification.

Mathematical models have been used in engineering, especially for V&V through simulation and proof of concepts for quite some time now but model-based systems engineering (MBSE) is a relatively new subject to the aerospace industry.

So, what do we mean by the word model? There are many definitions out there but here are some definitions in the context of systems engineering (SEBoK, 2020):

- A simplified representation of a system at some particular point in time or space intended to promote understanding of the real system. (Bellinger 2004)
- A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process. (DoD 1998)
- An abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system (Dori 2002).

We prefer the first and last definitions because they stress the fact that a model is a simplification/abstraction of reality as we more often than not can't replicate the system of interest to the nth detail. So, the process of modeling is to see which features/characteristics are key to be described and in the words of Brian Greene (2011) "It's the art of knowing what to ignore."

Another part of modelling and simulation is the concept of model-based engineering (MBE) where models are used to investigate the behavior of system-specific (for example, landing gear, fuel, flight controls, etc.) components and their integrations into the larger context of the aircraft. Typically, MBE uses specific solvers provided by a variety of commercial and business-owned tools that simulate a particular type of physics using

variable step integration methods. Especially in the development phases of a product such as an aircraft or system, MBE suggests the use of models (where needed on an ad hoc basis) to assist the development process, but not as a master.

On the other hand, model-based systems engineering (MBSE) is defined as "the formalized application of modelling to support systems requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" by the INCOSE Systems Engineering Vision 2020 (INCOSE, 2007).

The idea of MBSE is that we not only use models, but we use them in a consistent way in conjunction with requirements, V&V data, etc. throughout the lifecycle of the system/product/service. The fundamental objective is to ensure that different stakeholders of the system view the basic functionality, interfaces, and requirements to be correct and consistent. In order to use these views legitimately within the aircraft development and to ensure that the resulting artefacts can be used in the certification process, a robust traceability platform is needed, and this requires configuration management.

Configuration management. CM origins have roots in a widely spread story about a missile project (Gonzalez 2012):

When a successful demonstration was finally made and the projectile hit its target, the buyer said: 'build me 100 more', the industry found themselves in the following dilemma:

- Their prototype was expended…
- They did not have adequate records of part number identification, chronology of changes, nor change accomplishment. Technical publications did not reflect all the various changes...
- ... it was obvious that a second success could not be guaranteed, nor an identical article produced.

Therefore, CM as a formal management approach was developed by the US Air Force for the US Department of Defense in the 1950s as a technical management discipline for hardware material items – and it is now a standard practice in virtually every industry (Gonzalez 2012).

So, according to EIA 649 Configuration Management (CM) is defined as a technical and management process applying appropriate processes, resources, and controls, to establish and maintain consistency between product configuration information, and the product. (SAE, 2019).

CM today identifies five main activities (ISO 2003) and (SAE, 2019):

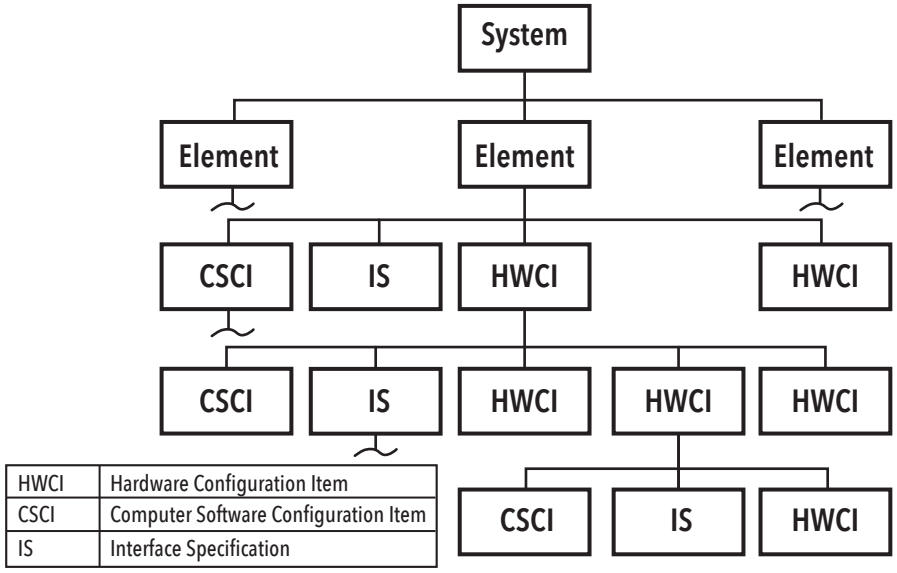


Figure 1. Example project specification tree (INCOSE. 2012)

- Configuration management planning – over the life cycle of a product is essential to achieving effective, predictable, and repeatable CM processes.
- Configuration identification – addresses the composition of configuration information, how each document, product, and unit or group of units of a product are uniquely identified (identifiers); how relationships are maintained in product structures; how elements of the configuration are verified and released; how the product configuration and components of it is/are baselined for change management; how interfaces are defined and managed; and how Configuration Items (CIs) are assigned/designated.
- Change control – includes managing both changes to and variances from the approved product configuration information, using a systematic, measurable process. The configuration change management function applies to all types of products and all program phases.
- Configuration status accounting – consists of the recording and reporting of information needed to trace and manage a configuration effectively by providing information and process status, as well as CM process performance data. The purpose of CSA is to capture, record, retrieve and report status and performance information about the product under configuration management and make the information accessible to support program/project activities as needed.
- Configuration audit – establishes that:
  - Appropriate CM processes are in place and that they are effectively operating to maintain consistency between the product and its product configuration information throughout the product life cycle.
  - The approved product configuration information is complete, accurate and current to produce the product, and applicable operation and maintenance instructions, training, and spare and repair parts.
  - The physical, functional, and interface requirements, defined in the approved product definition information, are achieved by the product.

Some of today's standards that describe the requirements for CM are the ISO 10007, EIA 649 C, EN 9100 and, in the domain of the civil aerospace industry, EASA and FAA regulation, as well as recommended practice such as specified in ARP4754.

CM FOR MBSE

Systems engineering addresses complexity of systems by decomposing them into several subsystems etc. In the version 3.2 of the INCOSE Handbook (INCOSE 2012) we have an example of such a decomposition, see Figure 1.

The product breakdown structure (PBS) is a key part of the development as it identifies "how" a particular functionality is implemented within the final product. The PBS along with its links to the functional decomposition give the impact assessment in case of failures. In the example shown above the breakdown is shown with some configuration items (CI).

A CI is defined as:

- any portion of hardware, software or composite item at any level in the system hierarchy designated for CM. A CI has defined functionality, is replaceable

as an item, has unique specification, and its form, fit and function is under formal control (Wiley 2015).
- an entity within a configuration that satisfies an end use function. Configuration: interrelated functional and physical characteristics of a product defined in product configuration information (ISO 2003).
- a product, allocated components of a product, or both, that satisfies an end use function, has distinct requirements, functionality and/or product relationships, and is designated for distinct control. (SAE 2019).

Now, what we define as a configuration item is an interesting debate and Steve Easterbrook from CMPIC wrote a white paper (Easterbrook 2016) that reflects the debates around this topic. According to him a document can't be a CI, but what about a model? Some of the definitions above are clear that the CI must be a portion of the product; others are happy for it to be a characteristic linked to the product.

The models that are referred to in this paper are associated with a specific product, the landing gear of an aircraft and correspond to a real-world example. The aircraft decomposition in the aerospace industry has been standardized for the benefit of easier maintenance by the ATA standard starting with ATA 100 then the ATA Spec 2200 and most recently the S1000D.

The landing gear is defined as ATA chapter 32 and then it further decomposes in the following sub-chapters:
- 00 General
- 10 Main Gear and Doors
- 20 Nose Gear and Doors
- 30 Extension and Retraction
- 40 Wheels and Brakes
- 50 Steering
- 60 Position Indication and Warning
- 70 Supplementary Gear.

So, we start having the first level of decomposition of an aircraft (A/C) already by the standards, then we further decompose the A/C in equipment and software and link them back to the ATA section/subsection they belong to.

Modelling for the A/C and subsequently landing gear links to the systems that are put on the A/C. Some models are even being used to simulate the behavior of A/C components for the training of pilots so they need to be an accurate representation of the equipment/system they are modelling.

In the next section we will look at two different real-world examples of modelling within the landing gear domain and how configuration management was performed (or lacked) and draw lessons learned and potential steps for the future. In the two examples below, an industry standard MBSE process covering a seamless integration of operational, functional, logical, and physical elements is not followed. In these examples, the use of the term MBSE is limited to the management and integration of domain-specific models into a unified aircraft model and a manual traceability to requirement management systems, for example, DOORS. The models, however, are built and change-managed using an internal model management process that becomes part of the company's business process and is certified by the authorities. Further details about this model management process where a single platform is used to specify, build, verify, validate, change and store models with complete traceability to the aircraft and system requirements is defined in the next section.

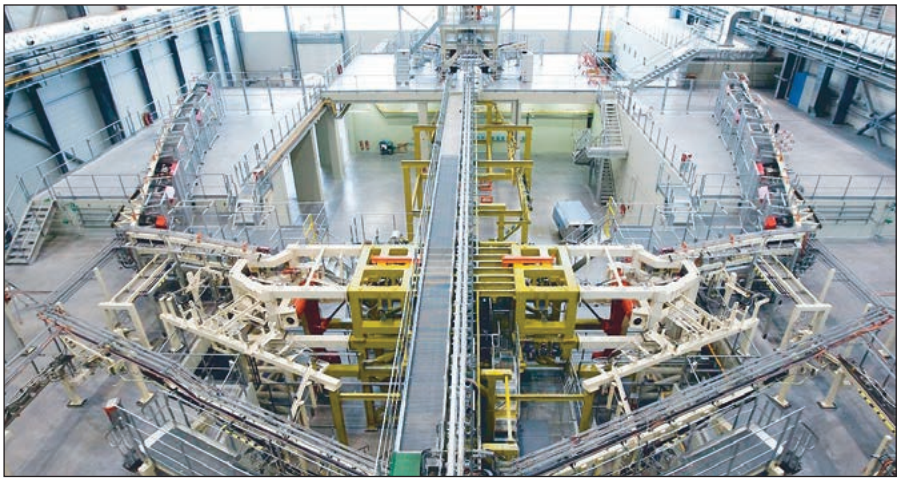


Figure 2. A full-scaled test rig nicknamed the "Iron Bird." (Airbus 2017)

VIRTUAL IRON BIRD FOR CERTIFICATION OF THE AIRCRAFT HYDRAULIC SYSTEM

Long before an Airbus jetliner takes to the skies, the flawless operation of its electrics, hydraulics, and flight controls is meticulously confirmed with the help of a giant test rig nicknamed the "Iron Bird." (IB) shown in Figure 2 (Airbus 2017). Building, operating, and maintaining an Iron Bird is expensive and amounts to millions of Euros during the development of the aircraft and beyond.

The sheer cost of operating the physical iron bird combined with the push towards a more digital future triggered an opportunity to build a virtual iron bird. A virtual iron bird would not only optimize the costs but also maintain the same rigor in testing. The virtual iron bird would be a digital equivalent of the physical iron bird in all sense and purposes. It will therefore be a representative of the actual A/C within the margins identified at the beginning of the project and which are also agreed with the aviation authority. In the past NASA looked at similar concepts for spacecraft repairs and maintenance (NASA 2004).

The Airbus A350 model version –900 aircraft (A/C) has a physical iron bird to achieve the integration tests of its hydraulic, electric, flight control system, and other transverse systems. These are necessary to cover verification and validation objectives in the frame of the A/C development life cycle.

The longer version of this A/C, the –1000 version, was considered a challenge in terms of the upgrade of the physical iron bird from the previous version A/C, –900 version. Therefore, it was decided to cover the verification and validation of the systems requirements mentioned above by several alternative test means including a virtual iron bird (VIB).

Test means for the -1000 A/C version included:
- Physical modifications of the actual iron bird
- The real-time virtual iron bird -1000
- The co-simulation platform related to the non-real time modelling
- Flight tests.

For developing a virtual iron bird that is a true image and functionality as the physical one it made sense to develop the virtual iron bird corresponding to the existing physical iron bird from the original model, the –900. This allowed us to test the virtual iron bird against the physical one.

Once the virtual iron bird for the -900 was developed and tested the work on the –1000 virtual iron bird could start. As there was no physical iron bird for this version the development of the virtual one followed the traditional method of injecting small changes to the validated –900 virtual iron bird to bring it to the desired configuration (mainly piping modifications).

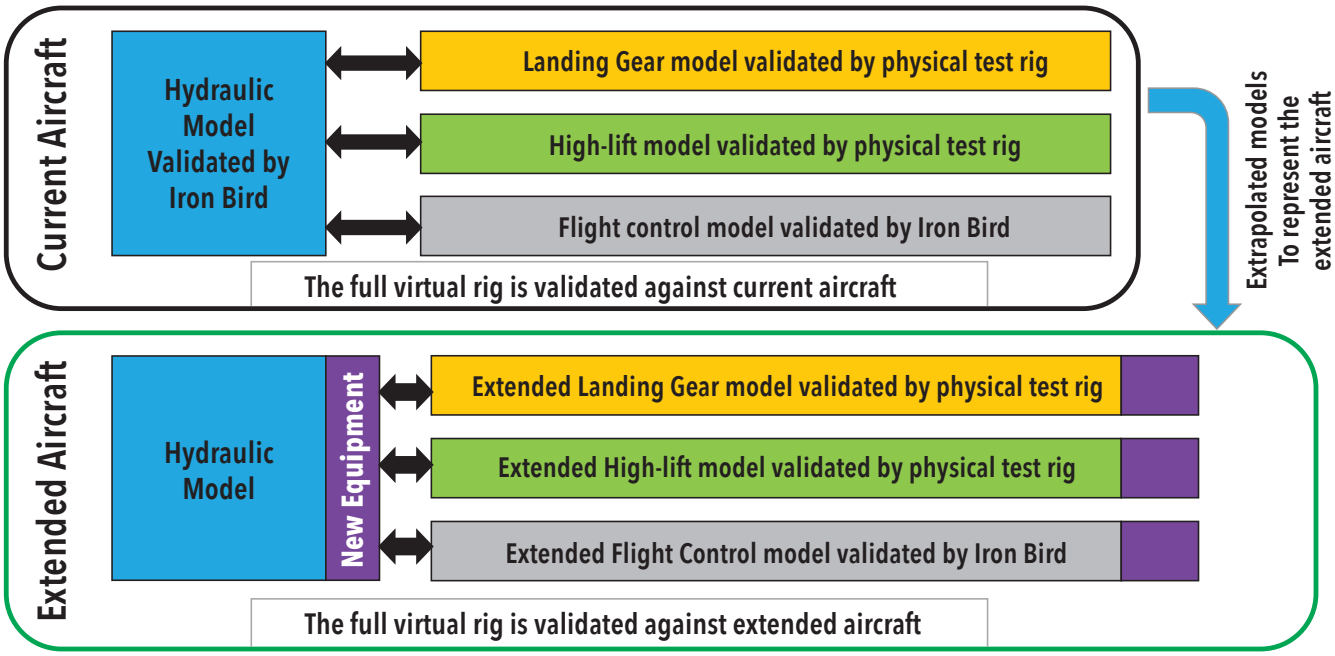


Figure 3. Systems overview of the iron bird

The –1000 virtual iron bird was integrated onto the flight simulators of the aircraft which have real avionics inputs but virtual engines and landing gears.

Even if the virtual iron bird results were used for the certification process alongside the flight test campaign its objective is extended to support safety of flight and development campaigns. even outside the type certification. It will also be used to perform integration tests and validate the sizing of the A350–1000 hydraulic circuit.

As shown in Figure 3, the virtual iron bird was made up of several models that represented the shared resource of the hydraulic architecture and its corresponding consumers, such as landing gear systems, flight controls, and high lift systems. Here, the current aircraft refers to the aircraft for which a real iron bird is built to the full scale and the data from which is used to validate the models. The extended aircraft refers to the variant of the current aircraft with a higher payload capacity and extended range. Figure 4 shows an illustration of the actual versus virtual iron birds. Specifically, the virtual iron bird is a combination of detailed (Amesim model) and real-time (Simulink model) hydraulic models built to investigate both the dynamic and steady-state nature of the system under consideration. While the Amesim models are integrated using the tools specific functionalities, the real-time models for various systems are integrated as a C code and business-owned platforms. The combination of models can also be referred to as a "digital twin" or "virtual twin" of the physical and real iron bird.

Figure 4. Actual iron bird versus virtual iron bird

One important aspect of building, analyzing, and comparing the virtual iron bird was to ensure a rigorous traceability exists between the equipment on the aircraft, equipment on the physical test rigs, and the corresponding equipment models in the virtual iron bird model. This means a comprehensive configuration management of both the equipment, and the models had to be in place to demonstrate that the representativity is fully achieved in all aspects of the systems under test.

In this paper we focus on the configuration management of the models but not the physical equipment, as the latter is part of the traditional aircraft development process that is mandatory for certification, operation, and maintenance. Specifically, the configuration management of the models was achieved using a "library approach," where the system under test is an assembly of subsystem models. These subsystem models form a library that is version controlled with a one-to-one relationship with the real equipment. The configuration management was extended to the analyses that were carried out on the virtual iron bird to ensure a seamless comparison with the flight test campaign.

In the sections below we will look at how configuration management was established for the models in this example.

Configuration management planning

Configuration management processes are standardized inside the company for the development of the aircraft as the system of interest. However, when it comes to the development of the associated models, etc., this management is less strict and more heterogeneous. For the models which will go on the flight simulators, the specifying, building, compiling, sharing, integrating, and changing is performed through a company internal standard. This standard ensures that the models have clear traceability to the software and hardware items and link to a particular airline and each A/C.

In the case of the models used for the virtual iron bird, the configuration management planning performed for the virtual iron bird was firstly a development from

scratch of the models for the –900 version and then incremental development/changes of the –1000. However, no formal CM Plan was issued for this model development just the CM strategy was outlined in various other deliverables.

Configuration identification

The formal identification of configuration items (CIs) is key. In this case the CIs were the models that represented the decomposed logical/physical system/equipment of the aircraft and are drawn from the PLM corresponding to the full aircraft. The CIs in the modelling world correspond to the system that is being modelled and its elementary equipment also form CIs in their own nature. A software-specific approach of versioning the CIs was implemented that generated a unique identifier that was linked to the actual software or physical item they were simulating.

As part of the standard model development process a whole series of modelling artefacts are created that include, documentation, model libraries, executable code, test scripts, analyses, and change requests. Therefore, each CI is allocated its associated set of artefacts and linked to the unique identifier that was generated earlier as part of the CI identification. Starting from the modelling requirements that include requirements of both functional and performance nature, the process follows to specifying the model to satisfy the requirements and its interfaces to other systems.

The model is given an "ID" (Programme_AirbusStandard_System_subsystem_ver_X_Y_Z) which is a concatenation of the program, Airbus standard, system details, and version that is consistent with the official Airbus aircraft simulation models data bank. The version in the naming convention is made up of three separate parts representing the type and severity of change (documentation versus correction versus evolution of the model).

Change control

As iterated above, the key to successfully creating the virtual iron bird (VIB) for the –1000 version from the –900 one was heavily reliant on successfully embodying the required changes to the system in a robust manner. For this, a manage model change process was used. The users (VIB, other test platforms, design office, integrators, etc.) of the delivered models raised change requests either as a correction or an evolution for the model but the change had to be approved by internal modelling management authority along with the relevant design office and model developers. It was then analyzed and planned for implementation. The model is then executed, tested, and released and finally integrated into the platform.

Create model change request: Model change request is created to identify any new need, evolution, or problem during plan simulation model(s) development task. The model coordinator creates a change request for any system evolution possibly impacting the model. A recording of model inputs and outputs is attached to the request identifying a functional problem.

Approve model change request: The model coordinator obtains approval and commitment of the model developer to take into account the change request for the next version during plan simulation model(s) development task. New user's needs are identified in this process.

Analyse and plan model change: The model developer describes and plans a technical answer to the problem/need during plan simulation model development and updates model delivery schedule according to the planned development.

Execute, test and release model change: The model developer modifies model items, following a change request indication, and implements associated verification (non-regression tests, dedicated verification tests). The model developer releases model change(s) identified in model delivery. This task is performed during develop shared simulation model.

Validate model change: Change request initiator provides the validation status of the change.

Configuration Status Accounting

As iterated in the previous section, links between the model evolution/correction to the product configuration items were established to record, approve, or disapprove, and coordinate changes to configuration items after formal establishment of their configuration identification. Configuration status accounting monitors the embodiment of changes and link with the physical product thus supporting the verification and validation of the models and subsequently of the two virtual iron birds.

The model verification plan is created before developing the model. This plan contains a detailed list of tests that need to be carried out on the model to ensure that it satisfies the requirements listed from the stakeholders. A matrix of compliance is produced with status *"compliant," "non-compliant" and "partially compliant,"* against the requirements. A model verification report is created along with the model delivery file that includes the summary of all the changes made and the applicable sections. The model delivery file is also the model description along with the parameters used in order for the end user to trace the modelling artefacts to its origins In addition to these documents, a whole array of change-related documentation exists that is applicable to the feedback on the model. For example, a model change request is raised against a model during its usage. This change request is then reviewed, approved, or declined, and the version controlled for traceability. If approved, the model is changed accordingly for the next delivery. All of these documents, along with the model are signed off by the relevant internal authorities, especially in light of the configuration management.

Configuration audit

The process was deployed to all impacted parties and internal audits must have been performed but none were performed during the time on this example.

NEW EXTENSION & RETRACTION SYSTEM

In this section we present a case where system certification could not be achieved due to the lack of rigorous CM implementation. Specifically, we highlight the deficiencies in the five main activities that are necessary to ensure a robust CM process.

The extension & retraction system is part of the landing gear (LGERS), which functions to reduce the overall drag on the aircraft by retracting the gear into the bay after take-off. To perform the V&V and certification activities related to this system, a physical test rig is built along with the accompanying hydraulic architecture of the shared resources. Typically, detailed models of the system are built and validated with the data from the physical rig and with the bench test data from the equipment suppliers. These validated models are then used for performance analysis and risk mitigation for first flight.

The type certification standard of the system design is replicated in the physical test rig and used to qualify the system with a standard set of tests. This, however, becomes quite expensive to replicate for every major modification of the system or its equipment. Modelling and simulation is seen as an alternative to physical tests from financial and time-to-market perspectives. As the representative models already exist that represent both steady-state and transient dynamics of the system, this was deemed as a viable alternative to physical tests.

However, during the V&V and qualification planning a major hurdle was identified to this approach. Specifically, lack of rigorous configuration management in place for the existing models, however detailed and representative they were, was a blocker. This was mainly because the

initial intention of building and validating these models was to use them for design analysis rather than to claim credit in the V&V and certification process. This put the whole plan into question although there was enough confidence that the models are a true and sufficient representation of the existing architecture.

Configuration Management Planning

The models that formed part of the development and performance analysis of the new LGERS were always used on an ad hoc basis and only stored as results rather than a planned version controlled platform that had a unique relationship to the product being modelled. Therefore, no planning was done to ensure a rigorous configuration management. Moreover, the performance models created during this program were not initially intended to be shared and hence the standard Airbus procedure for shared simulation models has not been followed. This shows that identifying the purpose of the models before creating them is crucial to make the best use of them in the future.

Configuration Identification

The models themselves were identified as configuration items but not the corresponding sub-models that represent the equipment. Therefore, no rigorous configuration identification has been performed. For example, the naming convention used was just to identify a change in the model but no details as to what the change specifically is and its severity.

Change Control

A clear one-to-one relationship between the changes in the models representing the components of the system and the real-world components was not truly established. Moreover, no change control record was managed with the rigor mentioned in the earlier section with the change request, approval, and validation as part of the process. Some of the change control was done through emails rather than a traceable platform that can be fully audited.

Configuration Status Accounting

The models are version controlled through a standard version control platform. Each version is given a number that would be traced for the changes made to the model. However, the details of each equipment and the relationship to the physical part numbers doesn't exist. Furthermore, the versions are only managed on an ad hoc basis to "save" the results and the corresponding model rather than the direct association with the real system and equipment.

Configuration Management Audit

The models were not subject to internal or external audit processes to ensure configuration management was in place.

CONCLUSION

In this paper we looked at current trends for configuration management in the field of model-based systems engineering looking at two examples from the landing gear domain. It is our hope that the paper highlighted the need to further strengthen the need for configuration management application in MBSE and overall modelling and simulation. Especially, with benefits being ripen in many ways and thus outweighing the efforts needed for an appropriate configuration management process.

One of the biggest challenges in deploying a robust modelling and simulation strategy with the necessary tools and frameworks is expensive. Although there is truth in that challenge, the return on investment (ROI) on such deployment can yield long term benefits and mitigate risks corresponding to complex system development. For example, the use of modelling and simulation to build a virtual iron bird needed several thousand hours of expertise and still resulted in more than a million dollars of overall development costs. This model will keep yielding the benefits in the future further increasing the ROI.

The use of modelling and simulation as a replacement for physical testing is not without its own risks. The representativity of a model can only be assessed in relation to an existing real-world system, which implies that a version of the system is already built, and similar tests have been performed on both the model and the physical system for validation. Therefore, soon, modelling and simulation can only be used to completely represent variants of a physical system rather than a brand-new entity. Even in such a situation, the physical representativity of the surrounding environment (systems and environment) is critical to build confidence in the model. Therefore, a thorough assessment of the ROI, both near and long-term, should be evaluated on an individual basis.

In the context of tomorrow's world of digital twins, configuration management plays a pivotal role in ensuring the success of the digitization program. If any A/C manufacturer intends to use a digital version of any given aircraft to analyze in-service issues or performance upgrades, then a digital version that is part-to-part identical needs to be built and that is only possible if all the models that are integrated are identified as configuration items, with change control in place with appropriate accounting for updates.

In fact, in the digital world each configuration item would probably have multiple fidelities of models which require a second layer of configuration management. This poses an additional challenge but is mandatory to do so as one model cannot satisfy the diverse and ever evolving needs of requests coming from the operational field of the aircraft.

As there are many digital twin versions of the same aircraft, configuration management of the modelling artifacts is mandatory to deliver a safe and efficient aircraft. This principle applies to other industries attempting the digital twin approach. ■

Acronym	Meaning
A/C	Aircraft
ATA	Air Transport Association of America
CI	Configuration Item
CM	Configuration Management
CSA	Configuration Status Accounting
CSCI	Computer Software Configuration Item
HWCI	Hardware Configuration Item
ID	Identification
INCOSE	International Council on Systems Engineering
IS	Interface Specification
LGERS	Landing Gears Extension and Retraction System
MBE	Model-Based Engineering
MBSE	Model-Based Systems Engineering

Acronym	Meaning
NASA	National Aeronautics and Space Administration
PBS	Product Breakdown Structure
ROI	Return On Investment
V&V	Verification and Validation
VIB	Virtual Iron Bird
VS	Versus (compared to)

REFERENCES

- Airbus. 2010. "Airbus opens A350 XWB Landing Gear Systems Test Facility in the UK." https://www.airbus.com/newsroom/press-releases/en/2010/10/airbus-opens-a350-xwb-landi ng-gear-systems-test-facility-in-the-uk.html.
- Airbus. 2017. "Taking flight with the Airbus "Iron Bird." https://www.airbus.com/newsroom/news/en/2017/05/taking-flight-with-the-airbus-iron-bird. html.
- ATA. 1999. Specification 100 - "Specification for Manufacturers' Technical Data, Revision No. 37." Air Transport Association of America.
- ATA. 2000. iSpec2200 Specification. "Information Standards for Aviation Maintenance."
- ATA. 2020. S100D Specification. "International Specification for Technical Publications." https://s1000d.org/ last accessed Oct 2020.
- Bellinger, G. 2004. "Modeling & Simulation: An Introduction." in Mental Model Musings. Available at: http://www.systems-thinking.org/modsim/modsim.htm.
- Greene, B. 2011. *The Hidden Reality*. New York, US-NY. Vintage Press, November, p 17.
- DoD. 1998. "'DoD modeling and simulation (M&S) glossary," in DoD Manual 5000.59-M. Arlington, US-VA: US Department of Defense. January. P2.13.22. Available at
- http://www.dtic.mil/whs/directives/corres/pdf/500059m.pdf.
- Dori, D. 2002. *Object-Process Methodology: A Holistic System Paradigm*. New York, US-NY: Springer. ISBN: 978-3-642-56209-9.
- Easterbrook, S. 2016. 2016 "Can a Document Be a Configuration Item (CI)?" CMPIC LLC. https://cmpic.com/whitepapers/configuration-management-configuration-item.htm.
- Gonzalez, P. J. 2012. "A Guide to Configuration Management for Intelligent Transportation Systems". By Mitretek Systems, Inc. US Department of Transportation. https://rosap.ntl.bts.gov/view/dot/37371.
- INCOSE Technical Operations. 2007. "Systems Engineering Vision 2020" version 2.03. Seattle, WA: International Council on Systems Engineering, Seattle, WA, INCOSE-TP-2004-004-02.
- INCOSE. 2012. "Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities", version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.
- INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities* version 4.0. Hoboken, US-NJ : John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0.
- ISO (International Organization for Standardization). 2003. ISO 10007:2003, "Quality management systems – Guidelines for configuration management". CH: ISO
- Maier, M. W. 1996. "Architecting Principles for Systems-of-Systems." INCOSE International Symposium 6:565–573. doi: 10.1002/j.2334-5837.1996.tb02054.x
- NASA. 2004. "NASA Working on Early Version of 'Star-Trek'-like Main Ship Computer." https://www.nasa.gov/vision/earth/technologies/Virtual_Iron_Bird_jb.html.
- NDIA. 2011. "Final Report of the Model Based Engineering (MBE) Subcommittee". Arlington, US-VA : National Defense Industrial Association (NDIA).
- https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering /modeling-and-simulation/reports/model-based-engineering.ashx.
- Friedenthal, S. "What is a Model?" in SEBoK Editorial Board. 2020. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 2.2 R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed 23/09/2020. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.
- SAE. 2019, ANSI/EIA-649C "Configuration Management Standard."
- Wiley, pub. 2015. "Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities". 4th edition. Prepared by INCOSE. Compiled and Edited by D. Walden, G. Roedler, K. Forsberg, R. D. Hamelin and T. Shortell. ISBN: 978-1-118-99940-0.

ABOUT THE AUTHORS

Adriana D'Souza (CSEP) is a configuration management process architect for Systems for Airbus, having previously worked as a systems architect and as a design and development engineer on challenging and complex projects like large border security projects and air traffic management projects both at the subsystem, system and system of systems level in the Airbus Group. She was awarded an Honours MSc in computational science and engineering from the Technical University of Munich (Germany) and a BSc in mathematics and computer science from the Ovidius University of Constanta (Romania). Adriana is also a certified systems engineering professional (CSEP) with INCOSE.

Phanikrishna Thota is an expert in the landing gear technical domain at Airbus focusing on modelling and simulation activities across all phases of development. He worked previously at Eaton as a manager in engineering as well as Airbus, before that working in modelling and simulation and innovation. He has a strong academic background with a PhD in engineering mechanics from Virginia Tech (USA) an MSc in mechanical engineering from the University of Kentucky (USA) and a BSc. in mechanical engineering from the Jawaharlal Nehru Technological University College of Engineering (India).

You Don't Save Money by Doing Less Testing – You Save Money by Doing More of the Right Testing!

Andrew C Pickard, Andy.Pickard@incose.net; **Richard Beasley,** Richard.Beasley@rolls-royce.com; and **Andy J Nolan,** Andrew.Nolan@rolls-royce.com
Copyright ©2021 by Rolls-Royce. Permission granted to INCOSE to publish and use.

■ **ABSTRACT**
Like so many aspects of life, we are looking for value-for-money. But we need to consider the value in terms of both short and long-term gains. Although certification standards require verification that requirements have been met, we need to recognize that verification is also there to bring value to a project and to the business as a whole. However, prioritizing the value to the project over the value to the business can result in sub-optimization and an overall higher cost to the business. This paper examines a specific case, the prediction of the fatigue lives of critical parts in gas turbine engines, to illustrate the more general case of performing tests to calibrate models that then have general applicability across multiple projects, rather than focusing testing on the needs of a specific project. In some circumstances, testing may not even be the best approach to take; if some level of error escape into service is acceptable (unlike the life prediction example given in this paper) then more focus on requirements validation and design review may provide a more cost-effective approach. This is where the linkage in a systems engineering model between requirements, functions, failure modes and effects analysis, verification test cases, and available calibrated models can help with identifying opportunities and risks.

INTRODUCTION

A project to develop a product or system has requirements of its own beyond the technical systems requirements, most commonly including cost and timescale. These can often be seen as in competition with the work needed to develop and verify a solution that meets the technical requirements of the end-user system stakeholders.

When a program of work is proposed to develop and verify a product it can often appear too long and expensive to meet the business needs. The project cost/timescale requirements cannot be ignored. Unfortunately, there is a risk of there not being a balanced decision, and the "wrong cuts" are made. For example:

- Cutting what is seen as "unnecessary" pre-work to understand the problem, as this causes what is seen to be "delay" – in fact proper problem understanding can be shown to be differentiator between successful and unsuccessful projects (Honour 2014).
- Removing activities to find and mitigate risk – without understanding that risks emerge as the solution develops (Pickard et al. 2010).
- Removing expensive testing – without realizing the purpose of the test.

These issues can be summarized in the heuristic "you don't make a project cheaper by not doing things; you make it cheaper by doing more of the right things" (Beasley et al. 2014), adapted for the title of this paper.

A product that produces a system "on time and on cost" that doesn't meet all the technical requirements, such as product safety, is not complete. A chief engineer accountable for safety would not allow its release, and expensive rework would be needed before the project can be described as complete. However, a product that is developed late and over cost would equally be seen as a failure.

There should be an understanding of both the technical requirements and what is needed to produce solutions to them – needing the capability to develop a product that can meet the technical issues and the program constraints (Beasley and Pickard 2020). So, with a capability that ensures that we know what the right things to do are, the "right tests" are properly understood.

The remainder of this paper looks at one specific example, approaches to life prediction of critical parts in aircraft gas turbine engines, to illustrate the importance of performing "the right tests" to reduce testing costs. It starts with a discussion of regulatory requirements for the engine's structural integrity, including a set of

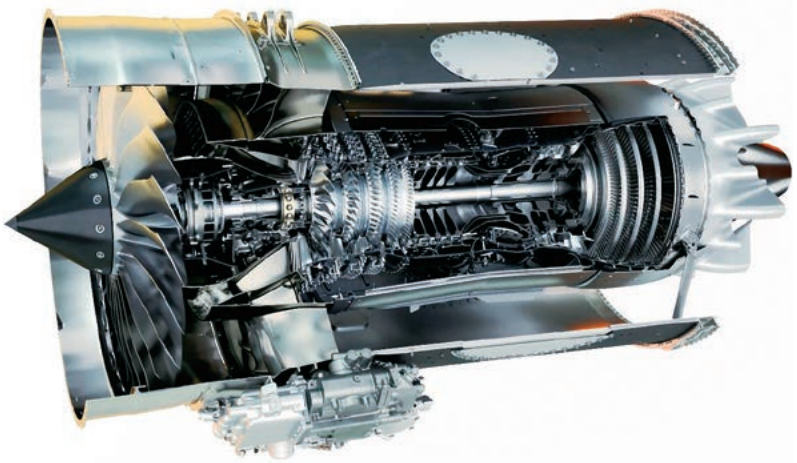


Figure 1. Typical two-shaft gas turbine engine

definitions that lead to the identification of "critical parts." Then there is a discussion of failure mechanisms, leading to requirements for reliable life prediction. The example focuses on fatigue life prediction for critical parts, comparing the "traditional" method for clearing fatigue lives for critical parts with a model-based approach. The findings are discussed, and some general principles are presented in the conclusions section of the paper.

AN EXAMPLE

The primary function of an aircraft engine is to provide thrust. The engine must meet stringent weight and fuel burn (efficiency) targets if this task is to be performed economically. This results in materials being used at high stresses and temperatures; under these circumstances there is a potential risk of failure of individual components.

The consequences of failure can be classified into various levels. A small loss of thrust, particularly if this is gradual, does not carry a high-risk factor. An abrupt change in, or total loss of thrust during take-off can be hazardous, although it may not be so at altitude. Failure that involves the release of high energy debris which could damage the aircraft, either structurally or functionally, form the highest risk category.

Figure 1 shows a sectional view of a typical gas turbine. The first step in establishing and minimizing the risk of failure for aircraft engines is to perform failure modes and effects analyses (FMEA) to assess the consequence of a failure of systems, subsystems and components. These product elements are classified as "critical" or "non-critical" depending on the consequences of their failure as identified by the FMEA. The various airworthiness regulators (Federal Aviation Authority – FAA, European Aviation Safety Agency – EASA, etc.) (FAA 2020; EASA 2018) have definitions and rules that are applied to this categorization, for multi-engine aircraft:

a) "Safe" engine failure: A "safe" engine failure is one in which the only consequence on the aircraft is partial or complete loss of thrust or power (and associated engine services) from one engine and should be regarded as a minor effect.
b) Minor effects: Minor effects will not occur at a rate more than that defined as reasonably probable.
c) Reasonably probable: This is unlikely to occur often during the operation of each aircraft of the type but may occur several times during the total operational life of each aircraft of the types in which the engine may be installed.
d) Non-critical component: Components whose failure would result in a "safe" engine failure are normally classified as non-critical.
e) Critical part: Where the failure analysis shows that a part must achieve and maintain a particularly high level of integrity if hazardous effects are not to occur at a rate in excess of extremely remote, then such a part shall be identified as a critical part.
f) Hazardous effects: The following effects should be regarded as hazardous:
    i) Significant non-containment of high energy debris.
    ii) An unacceptable concentration of toxic products being generated in air supplied to the airplane passenger or crew compartments.
    iii) Significant thrust in the opposite direction to that intended by the pilot, or complete inability to shut the engine down.
g) Extremely remote: This is unlikely to occur when considering the total operational life of many aircraft of the type in which the engine is installed but has to be regarded as possible.

Component Failure Mechanisms

During the assessment of both critical and non-critical component lives, all potential failure mechanisms must be considered. These failure mechanisms can be divided into three major classes:

a) Low life failures, usually associated with the incorrect application of Design Codes or unexpected overloading of the component.
b) Macroscopically non-localized damage accumulation failure mechanisms; uniform creep, corrosion or erosion are good examples here.
c) Macroscopically localized damage accumulation failure mechanisms, usually associated with the nucleation and growth of cracks.

For gas turbine components, stress margin requirements (relative to the yield and/or failure strength of the material) are normally arranged to ensure that low life failure mechanisms are not present. It is normal practice to back up analytical estimates of stress capability with testing of representative components.

Macroscopically non-localized failure mechanisms are usually avoided by applying time limits to the service use of a component. These life limits are normally calculated by evaluating the deformation and failure response of the entire component, for instance, predicting a turbine blade's life by full-scale creep analysis of the blade. An alternative is to perform a statistical analysis of experience, such as examining the scatter in creep growth of turbine blades in engines and applying statistical models to predict the minimum life for a given, acceptable level of growth.

Macroscopically localized failures are associated with the nucleation and growth of cracks, usually resulting from fatigue (cyclic loading), creep, environmental mechanisms, or their interaction. Occasionally, localized damage accumulation occurs from the start of service; a typical example is the fatigue failure of a component containing a material inhomogeneity or surface damage that is sufficiently severe to behave as a crack from the start of cycling. Alternatively, localized damage accumulation follows from a non-localized mechanism, for instance, in the fatigue of smooth specimens. Cracking results eventually from the accumulation of macroscopically non-localized slip damage.

Traditionally, fatigue life prediction has been accomplished by statistical analysis of past experience, including specimen and

Table 1. Component consistency – variables to consider

Material	Surface	Residual Stresses	Inspection Technique
Microstructure	Machining and Damage	Quenching	Repeatability
Homogenity	Scoring and Burning	Welds	Reproducibility
Defect Content	Dents and Bruises	Machining	Dectection Capability
Defect Type	Fretting		Level of Automation
	Processing		

full-scale rig test results. Fracture mechanics methods for fatigue life prediction have increased in importance and application, since it was realized that most, if not all structural materials contain inhomogeneities which can act as fatigue crack nucleators. The statistical nature of the distribution of inhomogeneities and the relative stressed volumes of components and specimens usually result in the risk of cracking from a large inhomogeneity or from surface damage being considerably higher for a component than for a specimen, highlighting the importance of performing full-scale component fatigue tests and incorporating the results into life calculations.

In the case of macroscopically localized damage mechanisms, the fundamental criterion for safety of operation of components in service can be stated quite simply:

"A crack in a component must not be allowed to grow to a size that can lead to the onset of rapid fracture under any engine condition", that is, commonly, a crack growth under low cycle fatigue (LCF, typically one or a few major cycles per mission) must not be allowed to reach the length at which either vibration or maximum load can promote rapid fracture (Asquith and Pickard 1988).

This emphasizes the importance of the need for reliable LCF life prediction models, which depend on understanding the fracture mechanisms and quantification of the controlling parameters.

Requirements for Reliable Life Prediction

Avoidance of in-service failure requires a sound knowledge of:

a) The environment and loads to which the component is subjected. This is fundamental to the design process.
b) The macroscopic response of the component to the applied loads and environment - temperatures, stresses, strains, corrosive/erosive conditions, etc. Accurate techniques for predicting local temperatures and stresses are essential here.
c) The microscopic response of the material from which the component is fabricated to the local temperature, stress, and environmental conditions. This requires a thorough understanding of material behavior and applicable life prediction techniques.
d) Consistency of component manufacturing — so that parts subjected to testing represent all the parts in service. Table 1 shows the main issues that need to be considered from a component manufacturing viewpoint.

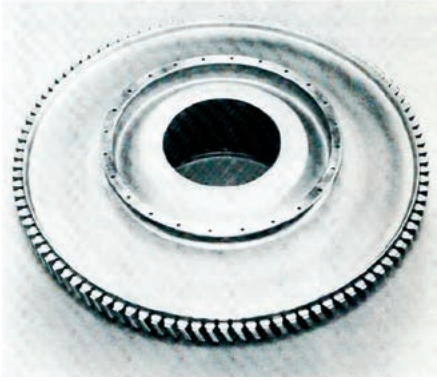


Figure 2. High-pressure turbine disk for a large commercial engine

Fatigue Life Prediction Methods for Critical Parts

This illustrative example is based on fatigue life prediction methods for critical parts in engines. Typically, all of the disks (also called discs or wheels) in an engine are critical parts, because their failure results in high energy debris that generally cannot be contained by the engine structure. Figure 2 shows a typical high-pressure turbine disk from a large commercial engine. The high-pressure turbine blades are located around the rim of the disk using features known as "firtrees". The disk locates the blades axially and radially in the engine; in the case of a turbine, the blades extract energy from the hot gases from the combustor, where compressed air is mixed with fuel and ignited, and converts this into rotational energy that is used to drive the high-pressure compressor. This in turn includes disks with blades that as they rotate increase the pressure (and temperature) of the air entering the engine. The high-pressure compressor delivers this compressed air to the combustion chamber. The engine illustrated in figure 1 is a typical two-shaft engine; the fan is driven by the low-pressure turbine and the high-pressure compressor by the high-pressure turbine.

In this paper, we will compare two different approaches to predicting the fatigue lives of these types of critical parts:

e) The "traditional" safe predicted total life method.
f) The "databank" fracture-mechanics (model) based life prediction approach.

Traditional Fatigue Life Analysis Methods

All "traditional" safe predicted total life analysis methods for aircraft engine components rely on specimen or full-scale component tests to define, statistically, a minimum life for the set of parts in service. All parts are then withdrawn from service before, or in the limit when they achieve this life. The statistical definition of minimum life is usually based on identifying the -3σ (1 in 739) or 1 in 1000 point in the distribution of component lives.

Although this approach sounds simple, and has been very successful in preventing in-service failures, there have been several areas of debate. The first of these relates to the choice of specimen or representative component tests to define the minimum life. Testing of a representative component guarantees that the life obtained is relevant to that part, with the correct standard of surface finish (machining, handling damage, etc.) and residual stress distribution. Full-scale component tests are expensive and time consuming, however, and it is rare for sufficient test results to be available on any individual part to allow the full definition of the statistical distribution of component lives to be obtained. Specimen tests can normally be performed in sufficient numbers to allow a full statistical definition of the specimen life distribution, but the volume (or surface area) of material subjected to representative stresses is considerably smaller than for full-scale components, which implies that the component minimum life may be lower than that of the specimens, requiring additional statistical interpretation of the specimen results. Figure 3 illustrates this by comparing several Titanium alloy (Ti-6-4) disks bore test results with plain 0 – max load control specimen results; one of the disk results falls below the statistically derived minimum life plain specimen results

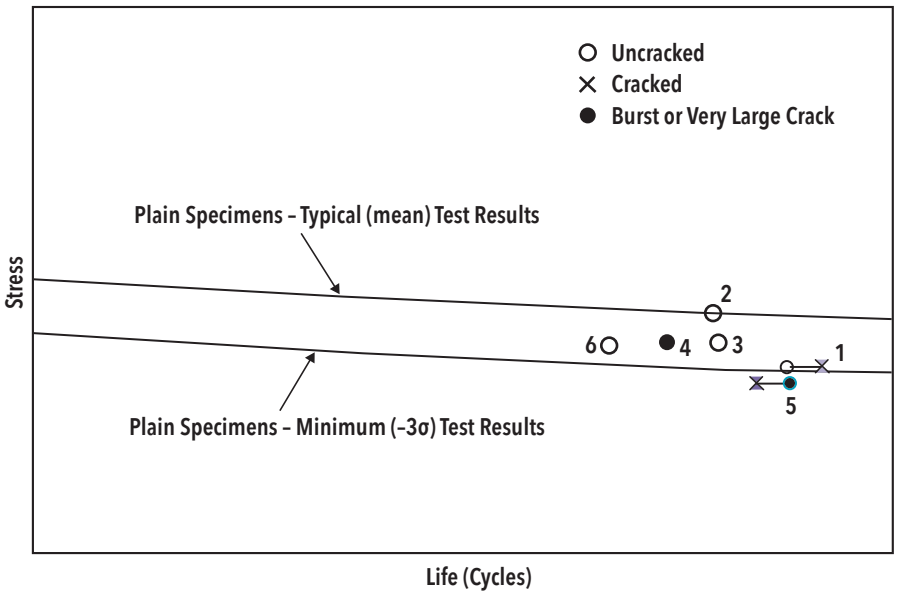


Figure 3. Titanium alloy (Ti-6-4) plain specimen and disk bore test results

(Asquith and Pickard 1988). In this case, laboratory investigation revealed handling damage marks in the disks from which the early cracking had originated; fracture surface striation counting indicated that most, if not all the disk life was in the crack propagation phase, with very little if any life consumed in crack nucleation.

In general, fatigue life predictions based solely on laboratory specimen test results need to be treated with caution because of surface finish and stressed volume considerations. Representative component testing is the "traditional" life prediction approach preferred by the civil engine certification authorities. Figure 4 identifies the factors normally applied to an individual component result to define the predicted safe cyclic life (PSCL) (Pickard et al. 1987).

In this approach, the component cyclic life is considered to follow a log-normal Gaussian distribution, with a ratio of 6 to 1 between +3σ and -3σ lives for most materials. The component tested is assumed to be in the top 5% of the distribution, and a ratio of 4 to 1 is applied to the tested life to obtain the minimum (-3σ) life. This gives 95% confidence that the PSCL is truly the -3σ life. Typically (68.3% of the time) the component tested will lie between +1σ and -1σ area of the distribution, giving some additional life margin. Performing additional component tests can be used to reduce this factor and extend the PSCL.

The second area of debate, with representative component testing, is the choice between running the test at the engine stress level or with an overstress factor. Testing at engine stress level is time consuming (for example, 80,000 cycles can take 30 to 60 days of continuous cycling based on typical cycling rates for full scale components) and typically will not result in cracking or failure of the component. This "life clearance" approach results in suspended (no life endpoint) results in the statistical distribution, which limits their use when creating fatigue life prediction models.

Typically, the relationship between life and overstress factor is that a factor of 4 to 1 in life corresponds to a factor of 1 to 1.3 in stress–so that a given life can be cleared by testing to this life with an overstress factor of 1.3. From a life clearance approach, testing with an overstress factor of greater that 1.3 does not offer benefit because the

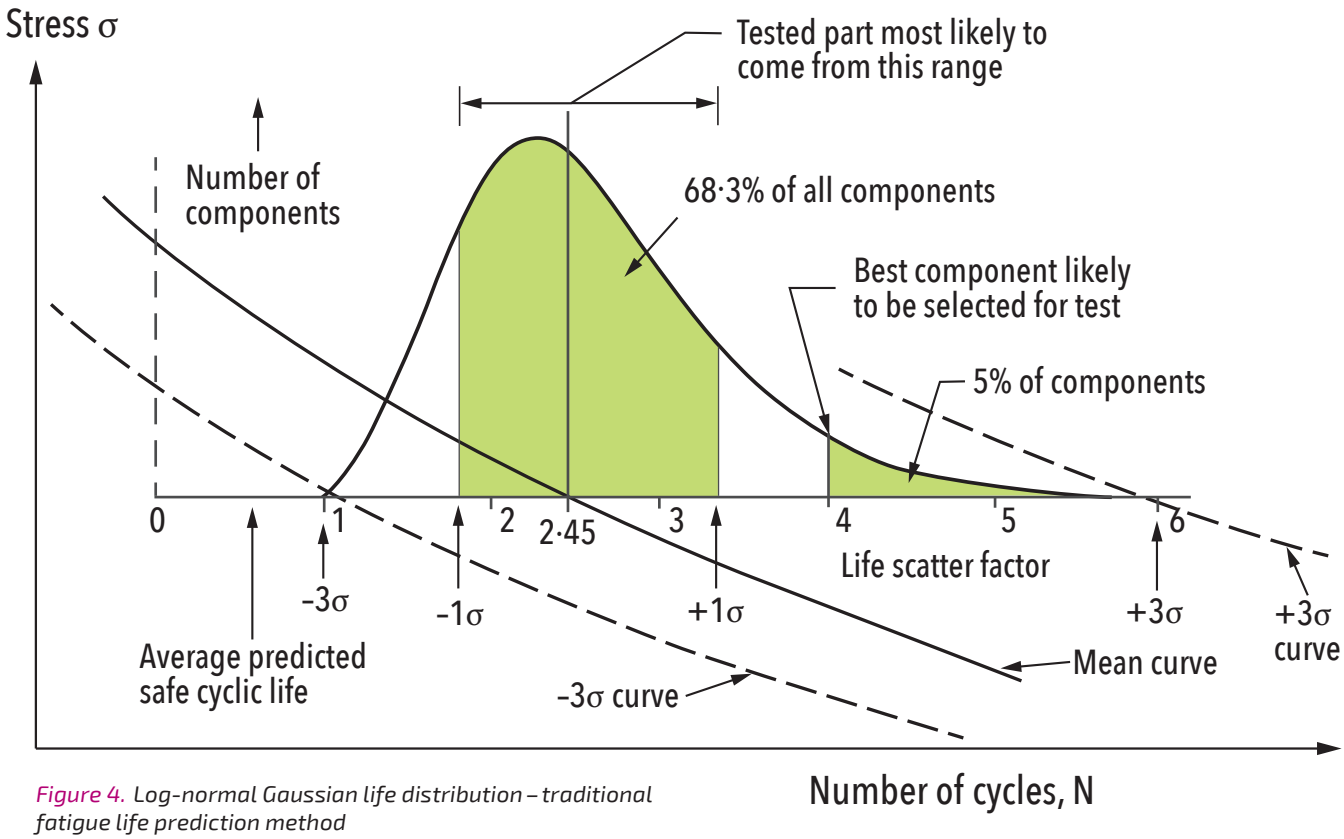


Figure 4. Log-normal Gaussian life distribution – traditional fatigue life prediction method

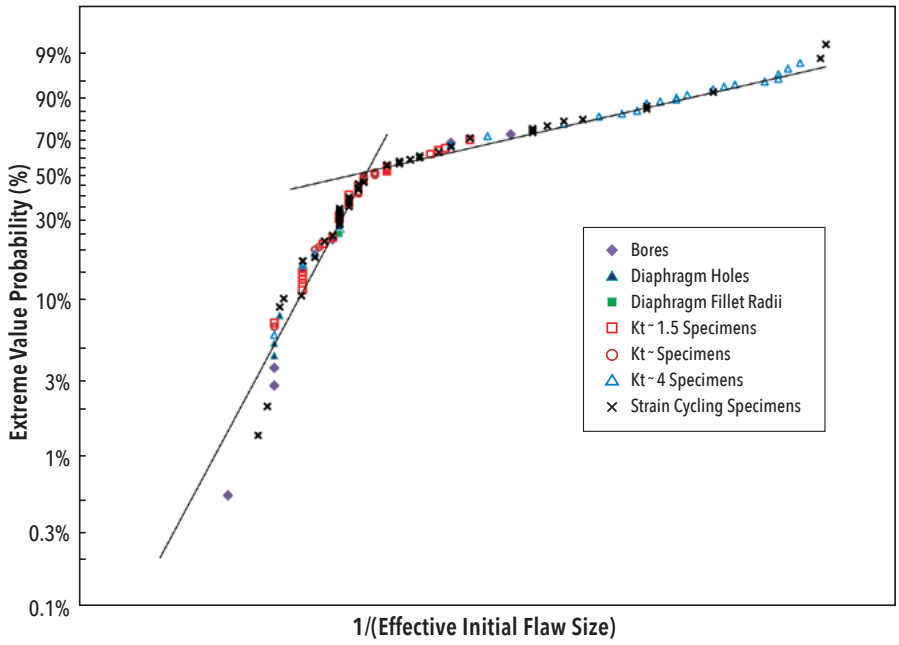


Figure 5. All results – effective initial flaw size

regulators require that the test must run for at least the cleared cyclic life of the part. Testing with an overstress factor occasionally results in cracking of the component, which gives a finite life. This also gives a cost-effective opportunity for a research project to fund continued running of the test to cracking or failure. However, projects often perceive that testing with a low or no overstress factor as "less risky". This perception drives "life clearance" as the project objective, rather than testing to cracking or failure to support model creation.

The "Databank" Model Based Life Prediction Approach

One solution to the problem of lack of statistically relevant samples of tests on individual components is to find a means of correlating fatigue test results on various full-scale components and laboratory specimens such that all available results on parts with microstructure and surface finish representative of parts in service can be included in the statistical assessment of the life distribution and hence of minimum life.

The Rolls-Royce "databank" approach uses linear elastic fracture mechanics (LEFM) models and fatigue crack growth information to analyze each individual test result to determine an "effective initial flaw size" (EIFS) that would have had to be present, from the start of cycling, to result in failure or the size of crack seen in each test result. Several studies have been performed to validate the use of LEFM to predict crack growth in components (Pickard et al. 1983; Jenkins and Pickard 1988). Statistical analyses of the distribution of effective initial flaw sizes may then be used to identify maximum size, on a +3σ or 1 in 1000 basis, that will be consistent with minimum component lives. The LEFM models can then address variations in component geometry, stress level, stress gradient and residual stress fields when predicting minimum fatigue life for crack growth from the maximum EIFS (Pickard et al. 1987; Asquith and Pickard 1988).

Figure 5 shows the results of an analysis of the inverse of EIFS (inverse to explore the largest EIFS) for the Titanium alloy Ti-6-4. The distribution is bimodal, with mostly specimen results in the upper branch and a mix of component and specimen results in the lower branch.

Figure 6 shows an analysis of the lower branch of the distribution. A three-parameter Weibull distribution has been fitted to the distribution, resulting in the identification of the maximum EIFS for the family of parts and specimens.

Figure 7 shows a comparison of the Titanium alloy disk bore test results from Figure 3 with the linear elastic fracture mechanics model-based prediction, for initial cracks equal to the maximum effective initial flaw size, and to half the maximum EIFS. The model-based approach now predicts a lower minimum life for test 5 than was achieved in the test.

An analysis was performed for all of the representative component tests, including suspended points. Figure 8 shows the ratio of actual to predicted life for each test. A three-parameter Weibull distribution is fitted to the results and the asymptote is equal to 1.

DISCUSSION

This fracture mechanics – based fatigue life prediction "databank" model was developed in the mid 1980's. Many of the specimen and component test results that supported the development resulted from a research project called the life and methods program (LAMP) that was conducted during the mid–1970's to early 1980's time period. Databanks were developed at that time for the Titanium alloy shown in this presentation, a high-temperature Titanium disk alloy and a Nickel based superalloy disk material. Subsequently databanks have

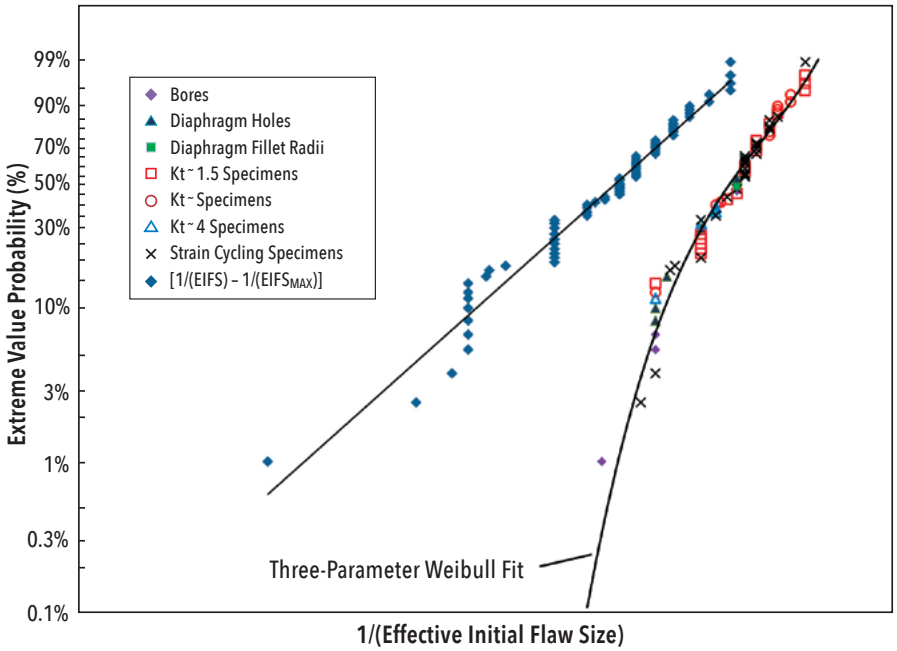


Figure 6. Lower branch – effective initial flaw size

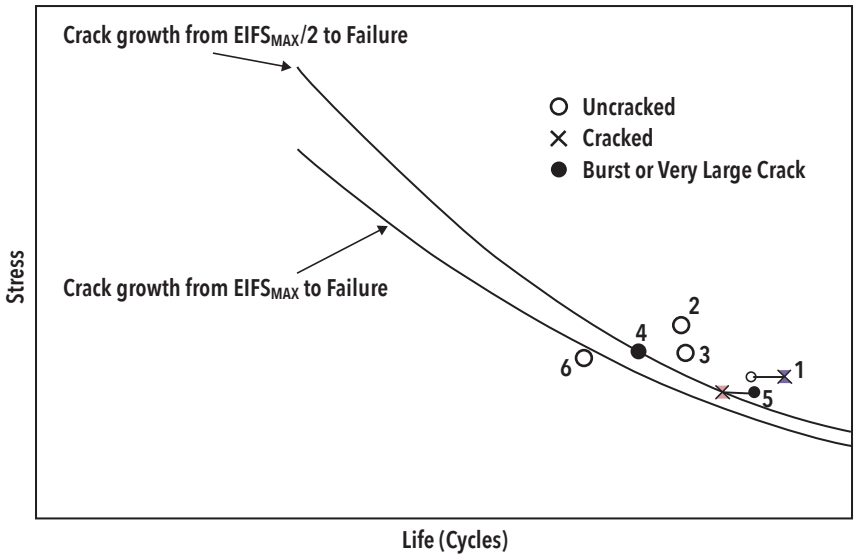


Figure 7. Titanium alloy (Ti-6-4) disk bore test results and model – based prediction

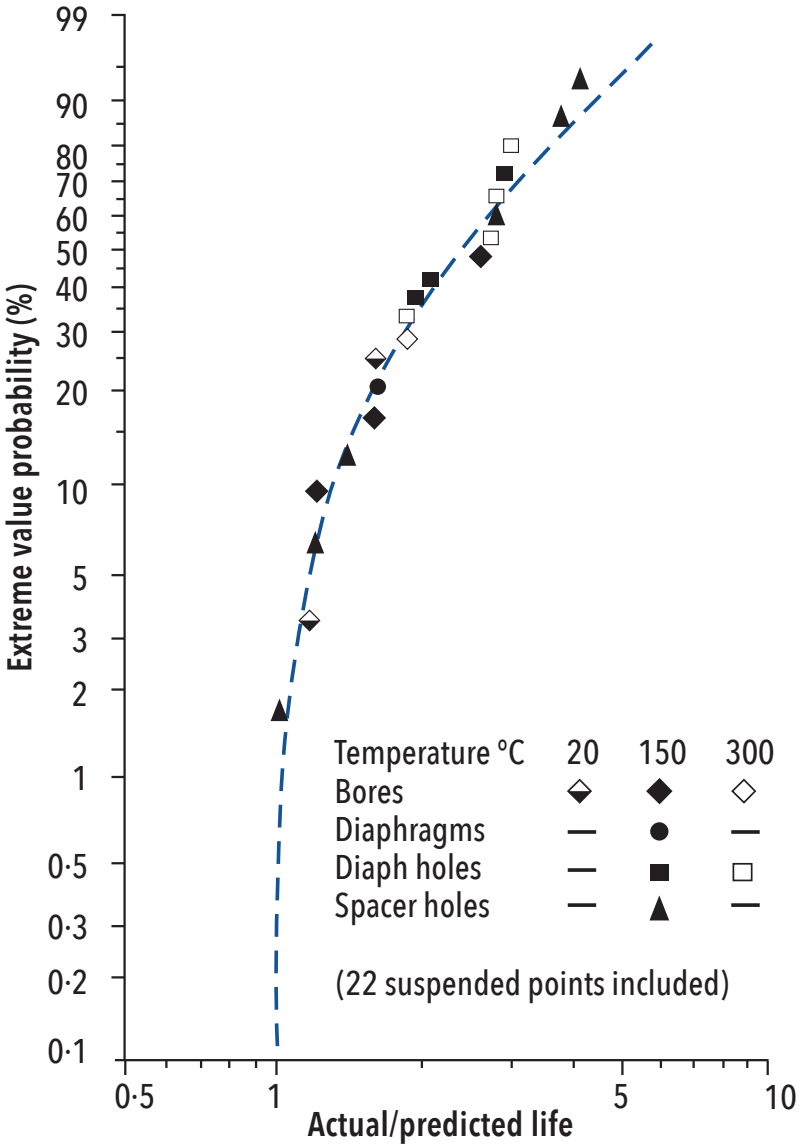


Figure 8. Actual versus predicted life, all full-scale component tests

been developed for additional materials. The regulatory bodies (at the time the British Civil Aviation Authority and the United States Federal Aviation Administration) approved these databanks in the mid 1980's as an alternative means of compliance for predicting the fatigue lives of critical parts, following over two years of review and discussion They have been used following this regulatory approval by Rolls-Royce for prediction of the lives of critical parts in a variety of gas turbine designs, and have been extended to cover other component features.

The regulatory approval is that the databank critical part life prediction models could be used for components that have features (bores, diaphragms, holes, etc.) that are included in the databank model and are of similar geometry and consistent method of manufacture without performing substantive additional full scale component tests, but further testing would be required to extend the databank to be used for a new feature or broader temperature range.

Performing representative component tests to cracking or failure as part of the LAMP program was a key factor in being able to develop this model-based approach. This illustrated to programs the benefit of testing at higher overstress factors for materials still using the traditional safe life approach, to allow these to be run on to achieve finite lives that can then be used to support databank model development.

There are a number of advantages to this model-based approach:

- Reduction in the number of representative component tests required to establish and approve the lives of critical parts.
  - Prior to the introduction of the databank life prediction approach, over 40 tests were performed on three engine types to clear the lives of the features covered in the databank. Following the introduction of the approach, life clearance testing was only required for one new feature type incorporated in the next engine design. This new feature has subsequently been incorporated into the databank.
- Use of the databank methodology to establish maximum stress levels during design of new components to ensure that they meet predicted safe cyclic life requirements.
- Use of the databank methodology to predict the minimum and typical stress – life curves for new features, to allow component test stress levels to be set with reasonable probability of a finite end point (cracking or failure) when testing to extend the databank to cover these new features.

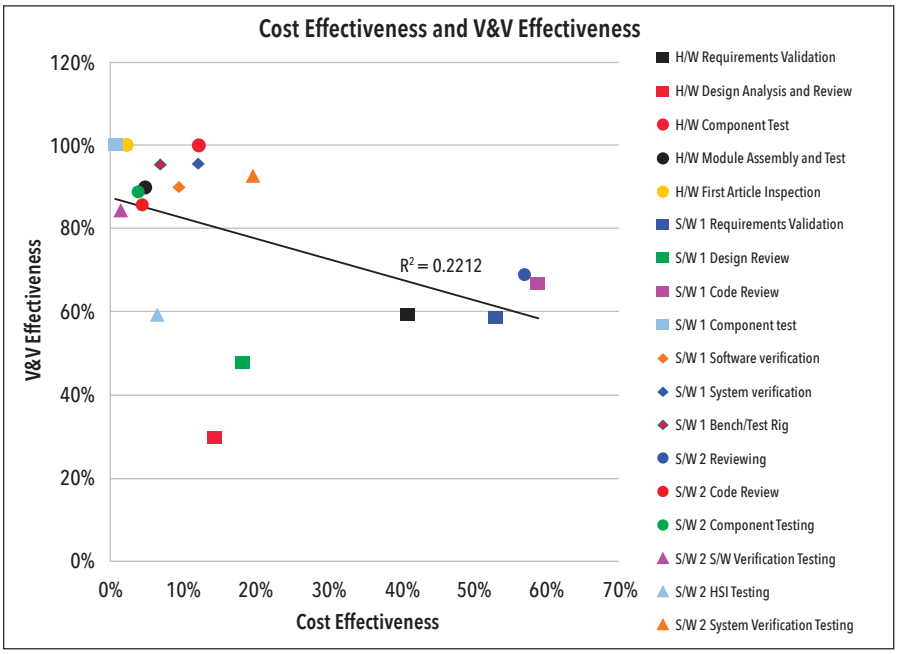


Figure 9. V&V effectiveness versus cost effectiveness

The databank has a scope, so when planning activities, there are some issues that have to be considered:

- Will the new design have features that are out of scope of the databank? If so, you may need to plan to extend the databank or perform traditional life clearance testing of the new features. This can be a constraint on design style evolution.
- Will a new design extend to temperature ranges outside the scope of the current databank? If so, you may need to plan to extend the databank or perform traditional life clearance testing.
- What are the rules for deciding who pays to extend the databank? Is it a research activity, or if only one project needs the extension, does the project pay?

WHAT TESTS SHOULD I PLAN?

The discussion above is about how to organize testing to obtain the best resulting value for the effort. However, sometimes testing may not even be the best way of achieving the desired result. Another factor is the effectiveness of validation and verification (V&V) activities (Pickard and Nolan 2013). In this earlier paper, we examined the relationship between V&V effectiveness and cost effectiveness of V&V activities (Figure 9).

Although the correlation is not strong, the implication is that more cost-effective V&V methods like requirements validation and review tend to have lower V&V effectiveness, whereas high V&V effectiveness methods are not very cost effective.

The implication here is that the selection of V&V methods to use depends very much on how acceptable it is to have issues escaping into the released product. For safety critical systems, such as the example discussed above, where any undetected escape may have unacceptable consequences, it is clearly important to employ high V&V effectiveness methods like hardware component test, hardware module assembly, and test and hardware first article inspection.

Where some undetected escapes may be tolerable, it may be more cost effective to spend extra effort on hardware requirements validation and hardware design analysis and review.

CONCLUSIONS

- Use of model-based approaches to reduce costs is nothing new — the example shown was introduced in the mid — 1980's.
- Use of this model-based databank approach was subjected to rigorous review by the regulatory authorities; acceptance by the UK Civil Aviation Authority and the US Federal Aviation Authority took over two years of review and discussions.
- The approach depends on a change in behavior – don't test at engine stress levels to clear life, as in the traditional fatigue life analysis methods, but test with overstress factors to achieve finite (cracked or failed) component test results.
- You don't save money by doing less testing – you save money by doing more of the right tests! This is a specific instance of the heuristic "You don't make a project cheaper by not doing things; you make it cheaper by doing more of the right things" (Beasley, et al. 2014).
- And there's more! The example model-based approach to critical part life prediction helps when creating new designs and introducing new design features.
- Consider alternatives to testing to reduce the cost to develop products — but for safety critical systems, make sure that any model-based approaches are calibrated using a databank of test results, including representative full-scale component tests. ■

REFERENCES

- Asquith, G., and A. C. Pickard. 1988. "Fatigue Testing of Gas Turbine Components." in "Full Scale Testing of Components and Structures", ed. Marsh, K. J., Butterworths, August, ISBN 0 408022 44 2. Also published in High Temperature Technology, 6.3, p.131, August.
- Beasley, R, and A. C. Pickard. 2020. "The Capability to Engineer Systems is a System Itself!" Paper presented at the 30th Annual International Symposium of INCOSE, Virtual, 20-23 July.
- Beasley, R., A. J. Nolan, and A. C. Pickard. 2014. "When 'Yes' is the Wrong Answer."
- Paper presented at the 30th Annual International Symposium of INCOSE, Las Vegas, US-NV, 30 June – 03 July.
- European Aviation Safety Agency (EASA). 2018. "EASA Certification Specifications for Engines, CS - E", Amendment 5, 13 December; see CS-E 15 for terminology, CS-E 510 for Safety Analysis, CS-E 515 and AMC E 515 for Engine Critical Parts; https://www.easa.europa.eu/sites/default/files/dfu/CS-E%20Amendment%205.pdf.
- Federal Aviation Administration (FAA). 2020. "Code of Federal Regulations, Part 33, Airworthiness Standards: Aircraft Engines." 15 October, Title 14, Chapter 1, Subchapter C, Part 33, §33.75, "Safety Analysis"; https://www.ecfr.gov/cgi-bin/text-idx?SID=eed43786296c5051130faf9170d05790&mc=t rue&node=pt14.1.33&rgn=div5.
- Honour, E. 2013. "Systems Engineering Return on Investment." PhD thesis, University South Australia. http://www.hcode.com/seroi/ (accessed 23rd October 2020).

- Jenkins, L. M., and A. C. Pickard. 1988. "Lifing of Discs." in 2nd International Parsons Conference (Materials Development in Turbomachinery Design), Institute of Metals, Cambridge, UK.
- Pickard, A. C., C. E. Brown, and M. A. Hicks. 1983. "The Development of Advanced Specimen Testing and Analysis Techniques Applied to Fracture Mechanics Lifing of Gas Turbine Components." in ASME Int. Conf. On Advances in Life Prediction Methods, eds. D. A. Woodford and J. R. Whitehead, Albany, US-NY, 18-20 April.
- Pickard, A. C., M.A. Hicks, and R. H. Jeal. 1987. "Applications of Fatigue Analysis: Aircraft Engines." in "Third International conference on fatigue and fatigue thresholds, Charlottesville, US-VA, 28 June – 3 July, EMAS.
- Pickard, A. C. 1988. "Design and Life Assessment of Aeroengine Components." in "Recent Advances in Design Procedures for High Temperature Plant." I Mech E symposium, MEP, ISBN 0 852986 80 7.
- Pickard, A. C., A. J. Nolan, and R. Beasley. 2010. "Certainty, Risk and Gambling in the Development of Complex Systems." Paper presented at the 20th Annual International Symposium of INCOSE, Chicago, US-IL, 11-15 July.
- Pickard, A. C., and A. J. Nolan. 2013. "How Cost Effective is Your V&V?" Paper presented at the 23rd Annual International Symposium of INCOSE, Philadelphia, US-PA, 24-27 June.

ABOUT THE AUTHORS

Andrew Pickard joined Rolls-Royce in 1977 after completing a PhD at Cambridge University in fatigue and fracture of metals and alloys. He is a Rolls-Royce associate fellow in system engineering, a fellow of SAE International, a fellow of the Institute of Materials, Minerals and Mining, a chartered engineer, and a member of INCOSE. He is immediate past chair of the SAE Aerospace Council, represents Rolls-Royce on the INCOSE Corporate Advisory Board and is chief of staff for INCOSE.

Richard Beasley joined Rolls-Royce in 1986 with a physics degree from Bristol University, and an MSc in gas turbine engineering from Cranfield University. After working on integration serodynamics, safety, reliability and life cycle engineering, he became global chief of systems Engineering. In 2011 he became Rolls-Royce Associate Fellow in Systems engineering. He was part of the BKCASE SEBoK author team, a leading author on the INCOSE systems engineering competency framework, is a past-president of the UK INCOSE Chapter, and is currently on the INCOSE board of directors as deputy director for services. He is a chartered engineer, fellow of the Royal Aeronautical Society, INCOSE ESEP, and was a visiting fellow to the Systems Centre at Bristol University.

Andrew Nolan joined Rolls-Royce in 1989 as a software developer. He was the chief of software improvements for over 10 years before becoming the chief of project estimation in 2013. Andy is full time in the development and deployment of estimation capability across Rolls-Royce.

Volunteer Opportunities

Join the Board of INCOSE

Help set the Standards in the field of Systems Engineering for the Global Community

There are several volunteer board positions coming up for election and now is the time to submit your application. The positions that are up for election are:

- President-Elect
- Chief Information Officer (CIO)
- Director for Outreach
- Sector Director, Asia-Oceania Sector
- Treasurer

Application Deadline: 6 August 2023

To find out more about the positions and to submit your application visit www.incose.org/volunteer or email voadmin@incose.net

INCOSE®
International Council on Systems Engineering
A better world through a systems approach

Inconsistent and Incomplete Datasheet: The Case for Systematic Use of Requirement Engineering

Lorraine Brisacier-Porchon, lorraine.brisacier@ensta-paris.fr; and **Omar Hammami,** Omar.hammami@ensta-paris.fr
Copyright ©2022 by Author Name. Permission granted to INCOSE to publish and use.

■ **ABSTRACT**
The lifecycle of a system is extended from its early conception to its retirement of service. The lifespan of unmanned ground vehicles (UGVs) can be expected to last over 50 years in the defense market. In this context, the rising complexity of UGV systems imposes engineering steps that would ensure both capabilities of the system and resilience to its future inclusion in a system-of system context. During its operational usage, the UGV is supposed to be maneuvered for specifically designed purposes following user manual datasheet of the components off-the-shelf (COTS) that were integrated. This paper exposes the public user datasheet relevance compared to the system engineering requirements that are the artifacts of system design architecture. The use of connecting COTS user manual to system requirements is discussed, all the more if the systems are to be re-used in a system production line. This article is intended to explore system of system conception methods for future robotized battlefield.

I. INTRODUCTION

Mobile robots are found in a wide spectrum of applications, such as farming, excavation, demining, or military. There are categories of usage and size for ground robots. They are considered as complex systems, because the cost of testing or simulating a significant amount of their behavior exceeds the investment and time that could be spent on design. Therefore, methods and process for prototype documentation is the key to manage its behavior. In this article, we consider the unmanned ground vehicle (UGV) basic functions are all-terrain navigation, heavy loads carrying, minimal maneuverability while being piloted. In the defense domain, where we intend to include swarm of robots, the UGV shall be robust, man walking fast, with the highest battery autonomy. Among others, we benchmarked a robot off the shelf, as if it were a product in development in a fictional company. The supposed expected performance for our system is presented in Table 1 system specification from datasheet. It shows small-scale problems assessed in complex system engineering and addresses a test scenario that draws integration of a robot in system of system battlefield.

The system mission on the field will be referred to as "mule concept," which includes remote piloting a mule that would carry loads instead of the pilot in a predefined time at a predefined speed. The degree of autonomy given to the mule is not at stake in this article. We will consider that the capability of "carrying loads" has been exploited to describe the system requirements, and that this set of requirements has been used to build the robot.

In this article, we place ourselves as buyers of such a UGV system. Making the acquisition of such a robot represents an amount of money, and we describe a method to verify the systems expectations.

As described in Stevens (2017), validation and verification (V&V) time and costs are a gamble for the company as well as the client. In this article, we explore how to use the V&V system engineering documentation in other contexts: user manual description and next product specification reuse.

The goal of this article is to discuss the relation between requirements, system V&V, system capabilities and multi-objective optimization as mathematical foundation for systems engineering. Figure 1 system driven engineering flow shows a process proposition for system development that includes optimization techniques. This foundation could lead to the creation of a method to shift from system engineering process as described in

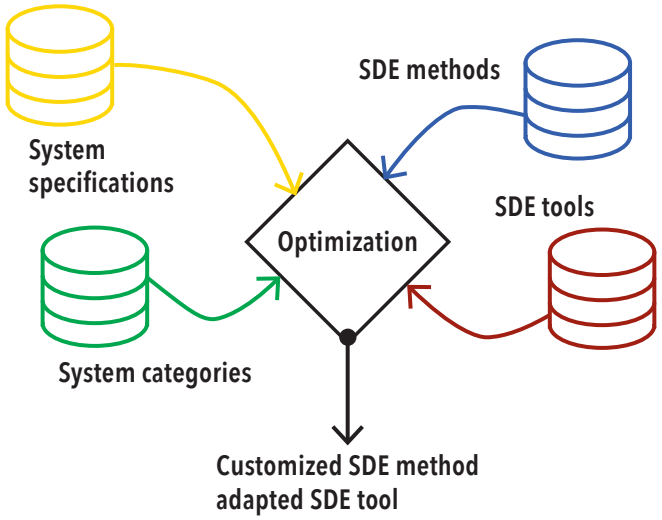


Figure 1. System driven engineering flow

AFNOR (2015) to a digitally based process (Henderson et al., n.d.) that would directly generate the best suited methods and tool for the mule system design in our fictional company.

II. STATE OF THE ART

a. Requirements engineering

Requirement based engineering is the most popular practice for defense system engineering acquisitions and architecture design (Huldt and Stenius 2019). The state of the art on requirement redaction is consistent, and recent publications shows that the interest in "good" requirements expression exceeds the systems engineering domain.

As model-based engineering (MBSE) practices progress in being associated to requirements (Bonnet, Voirin, and Navas 2019) the definition of system in early design is improving as complexity challenges rises. The return on investment of such practices (Duffy et al. 2021) has been identified. The vision of INCOSE for 2025 (INCOSE 2015a) seems to be carried out. Furthermore, the vision of the future (Voirin et al. 2020) extends MBSE to all models to implement the best systems in the most reduced cost and delays is being profiled.

The profile of model specifications could be adapted to the system context and categorization (Younse, Cameron, and Bradley 2021). It is a widely accepted prolongation to the SysML 1.3 success in system engineering (Wolny et al. 2020), because it adds views and connections to the system requirement expressions. The model aspects based on SysML does not express all required aspects of system design, which makes it incomplete, such as cited in Younse, Cameron, and Bradley (2021). This is a motivation for acquirers to specify systems in architecture frameworks, and to dig deeper on semantic approach (Duprez and Ernadote 2020) for system specification ambiguity reduction.

Publications on MBSE do not cover all expected transdisciplinary system behavior representations (Watson et al. 2020): mechanical or electrical requirements are rather considered as model-driven engineering (MDE). If those are the modeled expression of expected behavior, MBSE fails to represent all transdisciplinary aspects of the system. Therefore, the performance of the system will remain ambiguous until tests are performed in a real environment. The system scale compromises defined in system engineering cannot be represented.

To the best of our knowledge, none of these foretold systems engineering methods are the basis of system user manual generation. On the opposite, the teaser of requirement-based methodology state of the art is oriented in a top-down approach on the very early stages of system design. (Hahn et al. 2020) states that placing the 'design phase' above all considerations in order to gain time and/or costs is no guarantee of money efficiently spent. The article rather states the need for methods and tools tailored to the expected outcome of the product. If whole system lifecycle, for instance 50 years, is taken in consideration, focusing efforts on modeling and requirement expression might be more efficient if it is used during all the lifespan of the system rather than until the system architecture is conceived.

The application of requirements seldom includes the generation of end-user system datasheet. This can be caused by the difference of model details required at the process stage: the user manual exposes details of finite usage of the system, whereas the concept elaboration requires high level information. The mix of model granularity and fineness blurs decision making. The problem of multiple level of models that work together is addressed in digital twin usage questions such as in Bachelor et al. (2020), but no information can be found on how to mix them.

To conclude, system specifications are used to engineer the systems. User manual is written to make the best usage of the system. Yet, the effort to ensure maintenance and usability seems to be redundant with the effort to specify the system, especially during its integration phase. Since we have not found publications that would imply this, we expose a method that establishes a connection between requirements and datasheet to avoid redundancy. This two-ways connection could simultaneously improve the next-in-line product while reducing the costs and delays of engineering in comparison to its predecessor. The effect of our proposition will also be beneficial to the end-users because system capabilities description as requirements should be used to generate user manuals.

b. Experienced feedback on MBSE and IVV of complex systems to check the COTS engineering quality

The engineering process that has resulted in the creation of the COTS is easily identifiable from the behavior of the system. If the supplier has executed system engineering to design the COTS for the system, then the system passed a V&V qualification procedure. The testing environment that resulted in the system performance claims would be easy to reproduce. The method applied in this article is exposed in that way: what if the acquirer of the UGV COTS were to verify the system capabilities?

The set-based vision of verification strategies (Xu and Salado 2019) would be of help in passing the tests, but we would want to keep close to real-life situations where the scenario for our mule concept is too simple to afford long considerations before use. Furthermore, the changes in process and company organization to make this set-based vision a reality leaves the last call decisions to business investment. As resumed in Huldt and Stenius (2019), a lack of knowledge to integrate a model-based approach with current business processes is one of four arguments that prevent the development of MBSE. If the views are not provided, the common business/engineering vision cannot be built. Investment in "best effort" often prevails.

There are also publications measuring the benefits of the MBSE approach in managing the complexity of a robotic space system (Younse, Cameron, and Bradley 2021). But it makes no mention on how to re-use the specifications and how to organize all verifications that cannot be represented in SysML 1.3-like languages, such as measuring the tension of the battery in relation with the complete vehicle system. The multi-disciplinary requirements that should be the basis of system scale V&V cannot be represented in one model, and the best suited method and tools options for

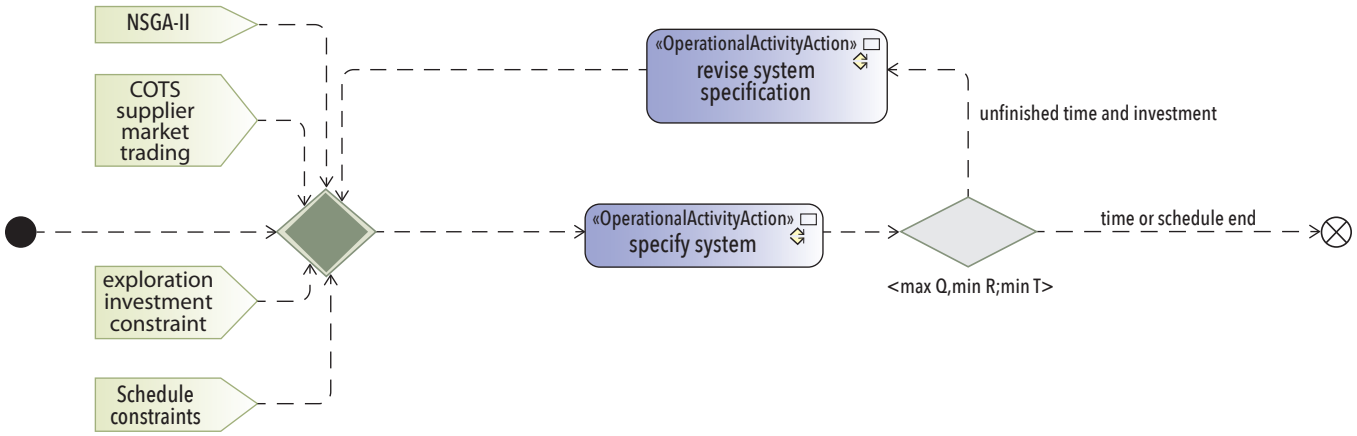


Figure 2. System specification flow proposition inspired of Wu et al. (2019)

software development objectives cannot be the best compromise on a multi-objective point of view.

The absence of formal expressions, methods or process associated to the language, as well as to poor investment in early stages of requirement specification (INCOSE 2015b) is proven to be a plausible cause of ambiguities in engineering (Duprez and Ernadote 2020). But there is no practical mention on how to enhance the existing process while reducing engineering costs and time in engineering. Furthermore, the mention of a model that would measure system outcome linked with investment in testing methods and tools for all physical performances is not discussed. Furthermore, the client satisfaction depends on how well the announced performances are met as described in datasheet. To the best of our knowledge, there are no recent research that would combine the user manual descriptions and qualification datasheet, contractors trust, and risk management. This research question was raised in Stevens (2017) but not exploited yet from a sub-system contractor point of view.

c. Theoretical proposition: optimization process

The state of the art in multidisciplinary and multi-objective optimization is also consistent. The mention of a need for a semantic extension to MBSE (Wach and Salado 2020), is still an issue to cover specific system categories and requirements and there are no quantified application feedback of those methods. Pate, Gray, and German (2014) presents graphical representations of non-dominated compromises in a system design evaluation. Optimization offers a consistent model for multidisciplinary approach. But it contains no link to a specific system goals or category in particular. We have chosen to use anonymized datasheet on the specifications used as our usage profile for the UGV system to organize the tests sets the optimization objectives as systems engineering process, tools and methods selection guide, depending on the equation:

$$(Q(x) \rightarrow max, R(x) \rightarrow min, T(x) \rightarrow min) \quad \text{(GOAL 1)}$$

The engineering flow proposed is selected by computer assisted search in which illustrated in Figure 1. It proposes system-driven engineering, based on four pillars: specifications, system categories, method, and tool. All pillars are linked with optimization through a common language: a combination of models that are either objectives or constraints. Its main advantage is the focus on the multi-disciplinary decisions that can be made, evaluating simultaneously the best process options for system development, and at the same time sketching the expected resulting system architecture. Detailed design costs are anticipated in that way. Figure 1 illustrates a proposition of method that would implement SDE. Instead of standard system specification and validation with a "V" or "W" cycle, the representation focuses on the revision of specification (functional and non-functional), functional architecture, component modeling, and physical architecture followed by multi-objective optimization in order to pick the most valuable information from the previous specifications. In this article, quality maximization is at glance. It is called method-tools-system engineering problem (MTS-EP). It states Q as a function depending on system specifications and category. Let $x \in S$ be a system design possibility. The MTS-EP problem can be formulated as:

$$(Q(x) \rightarrow \max \textit{ subject to } g(x) \leq 0 \textit{ and } h(x) = 0 \quad \text{(MTS-EP 1)}$$

The goal for Q maximization is therefore to find the $x \in S$ combinations that achieve the system defined objectives subject to constraints. The detail of the mathematical model is related to the investment and time input to the system concept elaboration. This problem alone is NP-hard based on the Knapsack problem. Functions g and h in (MTS-EP 1) are the mathematical expression of constraints of the problem, issued from specification, system category, and company's system experience from past projects. The flow in Figure 2 matches the representations for adoption of MBSE methodology sketched in Wu et al. (2019) and adds a constraint that is not underlined in Wu's publications, which is the end of the process, when allowed budget or schedule for system conception ends. This is an application of the principles of Wheaton and Madni (2018) on UGV system category: the identification of a Pareto front where the system scale budget and schedule constrains the concept choices.

In this article, x is instantiated by the UGV as it was designed and integrated. We want to put the UGV to a test and evaluate its quality as a mule system and evaluate the quality of its specification. If the system concepts definition quality is seen as an optimization problem, we could evaluate the performance of the acquired system with respect the specification, and at the same time make propositions that would simultaneously increase quality of the system and reduce method and tools investment to an acceptable compromise. The expected outcome of our system specification flow proposition (SSFP) can be measured in the next product that inherits of lessons learned from the previous solution. If system definition ambiguities can hardly be measured, its decrease can be measured. The adoption of our method can result in mastering of current expenses on concept exploration, direct generation of user manual automatically generated from the validation datasheet, maybe even obsolescence management, following Morgan, Holzer, and Eveleigh (2021) adding multi-physical dimensions to their claims.

Figure 3. Different views of the UGV instance

d. A mobile robot: UGV system instance description

Our instance is an all-terrain unmanned ground vehicle (UGV) for rapid prototyping and research applications and a robotic development platform. The capabilities of UGV can be expanded by multiple accessories such as a LIDAR, a GPS, a camera, and an inertial measurement unit (IMU). It can be used in the perception, navigation, manipulation and teleoperation and it is fully supported by ROS (robot operating system). Figure 3 shows UGVi views through different angles. The choice of the UGV instance (UGVi) is motivated by its popularity, modularity, and availability. Its hull design is supposed to host a load plate, a radar, a camera, and a radio command. All mountable equipment components off-the-shelf COTS are not equipped in our system, because the mule concept scenario does not include all features available. We do not exploit all possibilities.

UGVi is a satisfying example for mule concept because the datasheet in Table 1 offers a 50kg payload UGV. Our integrated human-augmented future battlefield required mules that would follow humans to help them carrying loads. It is also popular and has been the subject of publications for years in specific fields such as SLAM integration techniques, UGV positioning and vision/static and dynamic exploration, mixing optimization and further needs for UGVs, UGVi manipulations, specific objectives that can be stated for the UGVi in a software point of view. But the state of the art does not present multi physical propositions for UGVi usage modeling, nor its V&V process while being included in a wider system of system. Yet mono-disciplinary optimizations on the UGVi can be questionable regarding the expected multi-disciplinary use of the robot.

These aspects could be treated using the methodology depicted in Figure 2, giving multidisciplinary dimensions to cited applications of MBSE. Theoretically, the more precise and reliable COTS specifications are, the more chances system of system capabilities can be realized in acceptable costs and delays. The replacement of datasheet with specifications, all the more augmented by MBSE and optimization models to elicit the link between specifications and project costs and delays would lead to system of system simulation for early validation. This article depicts a practical experimentation to observe that theory in practice: if the datasheet proves to be incomplete and/or inconsistent, system of system modeling and simulation will not be available before buying 20 k€ robots and spending hours of time to describe its behavior.

III. DATASHEET PRESENTATION

a. Datasheet as requirements

For the experimentation, we have chosen to exploit only four of the presented specification: numbers 4, 5, 7 and 9. It is an experimentation field that we figured sufficient to address optimization quality model problem on practical view, because it presents contradictory requirements that induce concept dimensioning choices, while being easy to measure and to confirm. The battery autonomy will be highlighted as a substantial multi physical compromise. If the operating time defined in #9 is typically 3 hours, in which conditions can it be expected to reach this performance? If the laws of physics apply, the more load the mule is carrying, the more energy will be

Table 1. System specifications from datasheet

1	Dimensions	990 mm length 670 mm width 390 mm height	39 in length 26.4 in width 14.6 on height
	Track	555 mm	21.9
2	Wheelbase	512 mm	20.2
3	Weight	50 kg	110 lbs
4	***Maximum payload***[1]	***75 Kg***	***165 lbs***
	All-terrain payload[1]	***20 kg***	***44 lbs***
5	***Speed (max)***	***1.0 m/s***	***3.3 ft/s***
6	Ground clearance	130 mm	5 in
7	***Climb grade***	***45°***	***100% slope***
	Traversal grade	***30°***	***58% slope***
8	Operating ambient temperature	–10 to 30°	14 to 86°
9	***Operating time***	***3 hours typical; 8 hours standby (no motion)***	
10	Battery	24V 20Ah Sealed Lead Acid	
11	Battery charger	Short-circuit, over-current, over-voltage and reverse voltage protction	
12	Charge time	10 hours	
13	User power	5v / 12V / 24V; Each fused at 5A	
14	Communication	RS-232; 115200 Baud	
15	Wheel encoders	78,000 ticks/m	
16	Internal sensing	Battery status Wheel odometry Motor currents	

ID	Title	Requirement (mission)	Test Method
001	Maximum Payload various ground	The system shall carry 50kg load following an 3h mission on campus	Test pass if the battery runs out in more than 3h with load = 50kg on various grounds (tarmac and dry grass)
002	All-terrain payload dry grass	The system shall carry 20kg load following a 3h mission on dry grass	Test pass if the battery runs out in more than 3h on grass with load = 20kg
003	All-terrain payload gravel soil	The system shall carry 20kg load following a 3h mission on gravel soil	Test pass if the battery runs out in more than 3h on gravel soil with load
004	All-terrain payload tarmac	The system shall carry 20kg load following a 3h mission on tarmac	Test pass if the battery runs out in more than 3h on tarmac with load = 20kg
005	Speed (max on flat tarmac	The system shall maintain a 1m/s speed when following the mission onn tarmac	Test pass if the speed is measured to be ≥ 1m/s with 20kg load on average on 10m
006	Climb grade on various grounds	The system shall climb 45° slope carrying 50kg load when following the mission in the campus	Test pass if the robot climbs 5.5° and 25° on tarmac, and if it climbs 18.5°, 22°, 24° with load = 50kg
007	Climb test onn dry grass	The system shall climb 45° slope carrying 20kg load when following the mission in the campus	Test pass if the robot climbs 35° slope with load = 20kg on dry grass

consumed. What is "typical" referring to? The experimentations in place all refer to the battery extinction.

For example, it is mentioned that the operating time was about 3 hours, but it was not explained on which type of ground, the supplied energy, with how much payload and with which speed, and if it were compatible with slope climbing. Furthermore, if the UGVi was optimized for requirement #9, we imagine it would be so in all cases explained in requirement #4, supposedly on all grounds with maximum payload.

On the other hand, it is expected not to outreach the expected performance, otherwise it would mean that the system is over-qualified for some features. A perfect match between datasheet and real performance is at stake.

This selection of focus on four requirement exploitation, however arbitrary, covers all questions raised in the theoretical problem (MTS-EP 1). It addresses multi physical capabilities of the system with respect to its expected usage.

b. Maintaining the integrity of the specifications

The mule concept is designed to follow human pilot, while carrying his supplies for 3hr at human walking speed (5km/h). We have observed previously that connection between the remote control and the robot starts to fail when the battery capacity lowers. If the tension falls, the electronical devices that controls the connection will observe a disconnection. The witness of the robot indicating the battery levels do not indicate the voltage threshold that makes the communication fail. To measure the decaying power, we plugged a voltmeter to our experiment, which we found not intrusive.

Our system specifications at a glance are the following (see Table above). Note that our use case scenario was executed at the maximum speed the robot could reach, which is 3.6 km/hr. It is far from our objective of 5 km/hr.

IV. TESTS AND BENCHMARKING

We were inspired by the frameworks for vehicle tests: vibration, slope, and military standard procedures with adapted requirements. We have focused on battery level measurement to follow, inspired by Dogru and Marquez (2018). In this section, we will test the autonomy and energy specifications, the speed specification, as well as the climb capacities.

a. Measurements methods and means

To carry out these tests, we used some measuring instrument:

- Distance ruler precision ±0:5mm.
- iPhone telephone timer to calculate the duration ±0:05s
- Voltmeter at the entry of the battery precision ±0:05V.

b. Energy tests

i. TEST 001: Energy test, 50kg load on various ground

The environmental conditions, the nature of the soil and the payload have the biggest impact on the robot mobility. This has motivated us to test the 3 hours battery specification of requirement n°9 in Table 1 with 50kg payload and various ground nature as "best effort."

- Test stop condition: Battery running out.
- Environmental conditions presented in Table 2.

Table 2. Test log on various grounds, 50kg load

Temperature	Machining and Damage	
Humidity	Scoring and Burning	
Precipitation	Dents and Bruises	
Weather	Cloudy weather	
	50 kg	
Powered components	Lidar, GPS, calculator, WiFi, camera	
Flat Ground	**Type of soil**	**Duration**
	Tarmac	1hr 15min
	Wet grass	1 hour

Observations: test fail

1. After 1hr 30min of operation, we started to lose the communication with the robot (about 12 times in 30 min)
2. After 2hr 15min of operation, we totally lost communication. The autonomy requirement has not been met in our conditions.

Since our first test was unsuccessful, we planned to be more specific with the ground nature and load, and to report the behavior of the power system. The system will be loaded with 20kg in the next tests. The ambiguity of the Table 1 requirements will be resolved with our next tests.

ii. TEST 002: Energy test, 20kg load on grass

- Test stop condition: battery running out.
- Test results in Table 3.

Table 3. Test log energy on grass, 20kg load

	Starting time	Ending time
Time	13hr 45min	16hr 30min
Temperature	26°	27°
Humidity	42%	39%
Precipitation	0	0
Wind	7.2 km/hr	7.2 km/hr
Weather	Sunny weather	Sunny weather
Voltage	27.55V	24.05V
Load	20 kg	
Powered components	Lidar, GPS, calculator, WiFi, camera	
Flat Ground	**Type of soil**	**Duration**
	Dry grass	24hr 45min

Figure 4. Dry grass test environment

Observations: test fail

1. After 1hr 55min of operation, we started to lose communication with the robot (about 8 times in 50 min), we measured the voltage at each communication loss, and we presented the values in Table 4.
2. After 2hr 45min of operation, we have totally lost the communication. The 3 hours have not been reached.
3. We noticed that the communication with the robot depends on the Wi-Fi, but the minimum voltage and current required for normal operation of the Wi-Fi are 24VDC and 0.3A, this can explain the total loss of communication in this test and which occurred when the voltage was 24.05V.
4. The site chosen for this experiment is a dry grass, which is relatively flat, without tall grass, and therefore the robot can move forward without wasting much energy to submerge the grass, and there is not an up or down process during the operation.
5. Compared to test 001, the battery autonomy was extended by 30 minutes in similar conditions. This is a clue to the impact of the load on the system autonomy
6. from 16hr 20min, that is, in the last ten minutes of the experiment, even if the robot is controlled to go straight ahead, it cannot drive straight ahead (when controlled to go straight ahead, its trajectory is affected.

Table 4. Voltage measurements test grass, 20kg

Time	Voltage
15 hours 40 minutes	24.53 Volts
15 hours 57 minutes	24.45 Volts
16 hours 09 minutes	24.27 Volts
16 hours 16 minutes	24.16 Volts
16 hours 20 minutes	24.12 Volts
16 hours 24 minutes	24.09 Volts
16 hours 27 minutes	24.05 Volts
16 hours 30 minutes	24.05V

iii. TEST 004: Energy test, 20kg load on tarmac

- Test stop condition: Battery running out or loss of communication.
- Test results in Table 5.

Table 5. Test log energy on tarmac, 20kg load

	Starting time	Ending time
Time	14hr 20min	17hr 55min
Temperature	26°	27°
Humidity	45%	46%
Precipitation	0	0
Wind	10.8 km/hr	10.8 km/hr
Weather	Sunny weather	Sunny weather
Voltage	27.4V	23.79V
Load	20 kg	
Powered components	Lidar, GPS, calculator, WiFi, camera	
Flat Ground	**Type of soil**	**Duration**
	Tarmac	3hr 35min

Figure 5. UGVi on tarmac

Observations: test successful

1. Compared to test ii, the battery life was extended by another 50 minutes this time. This could indicate an influence of the ground nature on the battery autonomy.
2. After 3hr 35min of operation, we totally lost communication, but before that, we did not lose it. However, during the last almost 20 minutes of the experiment, the response speed of the robot to various operations degraded, and the rotation speed was significantly reduced.
3. We chose a flat tarmac. Compared to grass test 002, the tarmac has fewer gullies, so the robot has fewer bumps when moving forward. As a result, the power consumption was slower, and the communication system had less "chattering" because the direct current was not drawn to make punctual efforts to pass gullies.

iv. TEST 003: Energy test, 20kg load on gravel soil

- Test stop condition: Battery running out or loss of communication.
- Test results in Table 6.

Table 6. Test log energy on grass, 20kg load

	Starting time	Ending time
Time	14hr 20min	17hr 10min
Temperature	26°	27°
Humidity	45%	45%
Precipitation	0	0
Wind	10.8 km/hr	10.8 km/hr
Weather	Sunny weather	Sunny weather
Voltage	27.47V	23.92V
Load	20 kg	
Powered components	Lidar, GPS, calculator, WiFi, camera	
Flat Ground	**Type of soil**	**Duration**
	Dry soil with gravel	2hr 50min

Figure 6. Gravel soil illustration

Observations: test fail

1. After 1hr 45min of operation, we started to lose the communication with the robot (about 7 times in 45 min), we measured the voltage at each communication loss, and we presented the values in the following table.

Table 7. Voltage during test 003

Time	Voltage
16 hours 35 minutes	24.36 Volts
16 hours 49 minutes	24.28 Volts
16 hours 58 minutes	24.15 Volts
17 hours 03 minutes	24.06 Volts
17 hours 06 minutes	23.99 Volts
17 hours 08 minutes	23.95 Volts
17 hours 10 minutes	23.92 Volts

2. After 2hr 40min of operation, we have totally lost the communication.
3. There is a lot of coarse gravel on the ground, which can increase the friction when the robot moves, and make the progress of the robot very bumpy, where the bumps are much bigger than on tarmac and on dry grass.
4. Due to these gravels, the rotation speed of the robot decreased significantly, the rotation became very difficult and bumpy.
5. Although the experiment lasted 2 hours and 50 minutes in total, at the beginning it took us about 40 minutes to bring the robot to the test site, which is the ground with gravel. During these 40 minutes, the robot was moving on tarmac. In comparison with test 004, we assume that 40 minutes on tarmac could only improve the performance on our test on gravel.
6. This test should be run again to know if the performance on gravel is better than grass or not. But the fact that the test was still unsuccessful even with 40 minutes on tarmac and extra care on that day proves our point for this article: the requirement on "3hr typical" was not met in those conditions.

c. TEST 005: Load and speed tests

The maximum speed of the robot supposed to be equal to 1 m/s (as indicated on the Datasheet), So we did a speed test with 20kg load on Tarmac to put to test requirement n°5 from table 1.

Table 8. Test log combining load and speed on tarmac

Temperature	25° C
Humidity	42%
Precipitation	0%
Wind	11 km/hr
Weather	Sunny weather
Load	20 kg
Flat ground	Tarmac

Observations: test fail

1. The robot covered 9.57m in 9.90sec (on average) (for 5 trials: 10.00sec, 9.98sec, 9.78sec, 9.71sec, 10.02sec) on tarmac. speed= 0.967 m/sec.
2. The robot covered 17.94m in 18.47sec (on average) (for 5 trials: 18.55sec, 18.45sec, 18.26sec, 18.62sec, 18.46sec) on tarmac. speed= 0.971 m/sec.

3. We used the lines perpendicular to the edge of the road as the starting line and the ending point, but they are not completely vertical, so the robot was tilted when it moves forward, so the actual walking route is greater than the distance between the lines that we measured. Therefore, the measured speed is less than the actual speed at that time. If the robot tilts as it moves forward, the further it moves, the larger the distance measurement error and the larger the speed measurement error. At the end of the experiment, we found that if the side of the robot is aligned with the edge of the road at the beginning, the robot can better avoid tilting.

For accuracy, we have +/-0.5 mm at 1 m for the tape measure that was used to measure the distance and the error of the iPhone timer was +/-0.05s. However, we believe that the main source of error in the timing of the experiment is human reaction time. The coordination of the start timing, the reaction time when crossing the finish line, these errors will be greater than the instrument error and will depend on the timekeeper. This introduces a bias in the validation team that would motivate automatic simulation in later testing process.

d. TEST 006: Climbing tests

The maximum climb grade of the robot supposed to be 45° as indicated on the requirement n°7, so we did a climb test on wet soil, with minimal payload. Figure 7 represents the conditions:

Table 9. Test log climbing 34.5° slope, 20kg load

Temperature	12°	
Humidity	27%	
Precipitation	10%	
Wind	24 km/hr	
Weather	Cloudy weather	
Load	20 kg	
Powered components	Lidar, GPS, calculator, WiFi, camera	
Sloping Ground	**Type of soil**	**Slope angle**
	Wet soil	34.5°

Figure 7. UGVi climbing test 35° slope on wet grass

Observations: test fail

1. We tested the robot on a 35° slope on wet soil: the robot failed to climb it. It climbed about one meter upthe slope then it started to diverge to the right and started to tumble down the slope. as seen in Figure 7. The requirement was not met in these conditions.

e. TEST 007: Stress tests

The robot is suppose to climb 30 slopes with 20kg load, so we did a stress test with 50kg load on different slopes. This exceeds the all-terrain recommended payload in requirement n°4. Figure 8 represents the Tarmac test conditions, and Figure 8 to 12 represent the wet grass tests conditions.

Table 10. Test log stress slope, 50kg load

Temperature	12°	
Humidity	72%	
Precipitation	10%	
Wind	24 km/hr	
Weather	Cloudy weather	
Load	50 kg	
Sloping Ground	**Type of soil**	**Slope angle**
	Tarmac	5.5°, 25°
	Wet grass	18.5°, 22°, 24°

Figure 8. UGVi on tarmac 5.5° slope

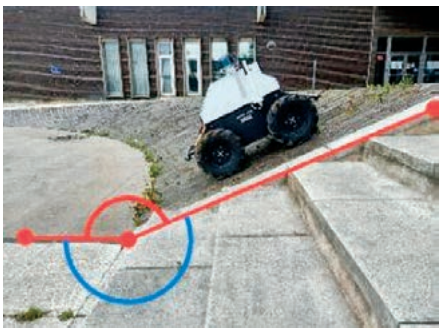


Figure 9. UGVi on tarmac 25° slope

Figure 10. UGVi on grass 22° slope

Figure 11. UGVi on grass 18.5° slope

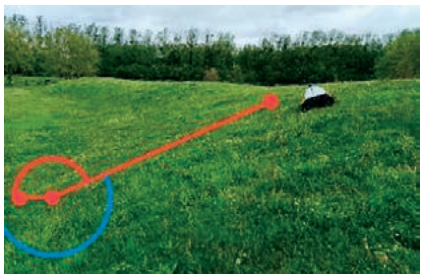


Figure 12. UGVi on grass 24° slope

Observations: test pass

1. We tested the robot on two slopes less than 20 °, the first one is about 5.5° on tarmac and the other one is about 18.5 ° on wet grass.
2. We tested the robot on three slopes less than 30°, two slopes on wet grass of 22 ° and 24 °, and a 25 ° slope on tarmac.

This indicates that for some requirements, the UGVi performs better than expected. This indicates that for this purpose only, the specification is not efficient, because these performances were never clearly specified.

V. DISCUSSION

The V&V on our UGVi demonstrated that the datasheet furnished no insight/hint to system usage, and that the UGVi system was not fit for our mule capability. Moreover, since almost no datasheet performance was met, we could not determine whether the system is intended for our purpose or not. At this point, we would have preferred to read the initial requirements, which are meant to express the system functions. Last, if the system were picked from a set of non-dominated optima solution to (MTS-EP), there would be a perfect match between the original intended capability of the system and its performance. The following actions could be set: lower the buyer's expectations, which could lead to the end of business, improve the datasheet, and improve the system performance toward some chosen capabilities.

We have introduced a loop that ensures the increase of company knowledge, and at the same times links directly the system requirements to its intended use cases and user documentation. Failing all tests on UGVi performance gives enough reasons to invest in a method that would benefit both user manual descriptions and correlation between system and its intended capabilities. The inclusion of MBSE in the process of capability description would decrease ambiguities on the intentions of the builder in the requirements and in the user manual. Our process can also include physical models at the highest level of conception. The complete optimized system-driven engineering architecture conception flow will include some re-engineering to transform the UGVi into an optimized mule system. This involves re-writing the system specifications and re-designing it in the first step, and then propose a re-design in those objectives that would fit the objectives better. The solutions could be explored with simulation rather than real-life V&V, to reduce the costs of re-design. The test logs could be smartly combined to improve the next specifications for future product developments.

We have considered the "mule concept" as the systems expected capabilities. There were no declared capabilities on the UGVi design details. There is no proof that the UGVi was designed as a "mule concept" more than another purpose. The concept architecture capabilities have never been exposed by the builder. This could be improved with more visibility on the system capabilities by the builder, using for instance the operational views in domain specific architecture frameworks and/or detailed representations of the system instance in model-based design or MBSE.

The SSFP design flow described is a proposition that matches requirement improvements, V&V practice, and user manual redaction challenges. It introduces a method that enhances the quality of the system V&V context, shares the best-known intended performances of the system, and automatically generates the user manual datasheet. Furthermore, the SSFP flow introduces a re-engineering method that would continuously improve engineering specifications while mastering products engineering costs and delays over the concept and maintenance phase of the system lifecycle. This could smoothly replace the "best effort" system engineering by a promising optimized system driven engineering. Further research shall focus on quantification of the gains using such a method UGV-category product development.

The correlation between the objective expression will be subject of other publications and will bring insight to the complete GOAL equation rather than only the "MTS-EP". If our proposition could help enhancing the quality of the system, the outcome on resource and time of design are considered "best effort." Further publications will fill the other goals of (GOAL) introduced in the state of the art.

For defense system definition, robots on the battlefield will follow the military standards given by ISO and STANAG, which will advise contract managers on how V&V for system behavior on battlefield will be considered (Michelson 2021). Therefore, the COTS which cannot prove to be compliant with the standards will be excluded from defense contracts because it will be impossible to integrate them in the larger system of system capability forecast.

VI. CONCLUSIONS

Datasheets are not adequate to reflect the properties of a system but can be replaced by system engineering specifications which improve accuracy of subsystem behavior description. The lack of completeness and ambiguity carried out by the very structure of a usual datasheet ends up with cumulative ambiguity and over-testing with no guarantee of results for end systems. Should datasheets be banned in favor of systematic requirement engineering files or documents requires that all suppliers of components and subsystems are trained and master the systems engineering process.

On the opposite side, datasheets easily reveal the lack of knowledge in systems engineering and requirements engineering. Therefore, companies have three options: (a) stick to the usual "reliable" supplier with no investment in writing the "should-be" requirements and avoid evolving equipment, (b) stick to the usual "reliable" supplier with consistent investment in writing the "should-be" requirements and pay for the component evolutions, or (c) impose systems engineering exchange protocols. Some industries such as the semiconductor and the embedded system industry are not familiar with systems engineering and make extensive use of component and subsystem datasheets. If the aim is to lay out seamless system engineering flow down to physical architecture and up to V&V, there is a major overhaul to be conducted.

In our multi-physical UGVi example, system tests failed: technical validation was not reached. If the system was sold to a client, none of the claimed capabilities could be covered as proved in our bench-marking. The only objective that was reached was not even specified, therefore would have generated no outcome. The experiment presented here illustrates the necessity of introducing environmental conditions and objectives to technical specifications. It also indicates that the original requirement artifacts for the system would do a better job at describing the expected performance of the system, especially if it is enhanced by architecture frameworks and MBSE concept design, which are the trends to seek completeness and unambiguous system description. ■

REFERENCES

- AFNOR. 2015. ISO/CEI/IEEE 15288:2015 *Systems and Software Engineering — System Life Cycle Processes*. AFNOR. www.afnor.org.
- Bachelor, Gray, Eugenio Brusa, Davide Ferretto, and Andreas Mitschke. 2020. "Model-Based Design of Complex Aeronautical Systems Through Digital Twin and Thread Concepts." *IEEE Systems Journal* 14 (2): 1568–79. https://doi.org/10.1109/JSYST.2019.2925627.
- Bonnet, Stephane, Jean-Luc Voirin, and Juan Navas. 2019. "Augmenting Requirements with Models to Improve the Articulation between System Engineering Levels and Optimize V&V Practices." *INCOSE International Symposium* 29 (1): 1018–33. https://doi.org/10.1002/j.23345837.2019.00650.x.
- Dogru, Sedat, and Lino Marquez. 2018. "A Physics-Based Power Model for Skid-Steered Wheeled Mobile Robots." *IEEE Transactions on Robotics* 34 (2): 421–33. https://doi.org/10.1109/TRO.2017.2778278.
- Duffy, James B, Robert Combs, Jingyao Feng, and James P Richardson. 2021. "Return on Investment in Model-Based Systems Engineering Software Tools," 15.
- Duprez, Jean, and Dominique Ernadote. 2020. "Towards a Semantic Approach of MBSE Frameworks Specification." *INCOSE International Symposium* 30 (1): 1405–19. https://doi.org/10.1002/j.2334-5837.2020.00794.x.
- Hahn, Heidi Ann, Nick Lombardo, Ann Hodges, Mitchell Kerman, and Frédéric Autran. 2020. "Implementing Systems Engineering in Early Stage Research and Development (ESR&D) Engineering Projects." *INCOSE International Symposium* 30 (1): 433–48. https://doi.org/10.1002/j.2334-5837.2020.00732.x.
- Henderson, Kaitlin, Virginia Tech, Perry St, and Durham Hall. n.d. "Is CAD A Good Paradigm for MBSE?" 15.
- Huldt, T., and I. Stenius. 2019. "State-of-Practice Survey of Model-Based Systems Engineering." *Systems Engineering* 22 (2): 134–45. https://doi.org/10.1002/sys.21466.
- INCOSE. 2015a. "A World in Motion - SE VISION 2025."
- ———. 2015b. *INCOSE Systems Engineering Handbook 4e*. WILEY.
- Michelson, Brian. 2021. "Why NATO Needs Lethal Autonomous Weapon Standards." https://cepa.org/why-nato-needs-lethal-autonomous-weapon-standards/.
- Morgan, Markeeva, Thomas Holzer, and Timothy Eveleigh. 2021. "Synergizing Model-based Systems Engineering, Modularity, and Software Container Concepts to Manage Obsolescence." *Systems Engineering*, June, sys.21591. https://doi.org/10.1002/sys.21591.
- Pate, David J., Justin Gray, and Brian J. German. 2014. "A Graph Theoretic Approach to Problem Formulation for Multidisciplinary Design Analysis and Optimization." *Structural and Multidisciplinary Optimization* 49 (5): 743–60. https://doi.org/10.1007/s00158-013-1006-6.
- Stevens, Jennifer Stenger. 2017. "Warranting System Validity Through a Holistic Validation Framework: A Research Agenda." *INCOSE International Symposium* 27 (1): 654–71. https://doi.org/10.1002/j.2334-5837.2017.00385.x.
- Voirin, Jean-Luc, Olivier Constant, Eric Lépicier, and Frédéric Maraux. 2020. "Dream the Future: Systems Engineering in 2030." *INCOSE International Symposium* 30 (1). https://doi.org/10.1002/j.2334-5837.2020.00754.x.
- Wach, Paul, and Alejandro Salado. 2020. "The Need for Semantic Extension of SysML to Model the Problem Space." 18th Annual Conference on Systems Engineering Research, Oct. 8-10.
- Watson, Michael, Azad Madni, Bryan Mesmer, and Dorothy McKinney. 2020. "Envisioning Future Systems Engineering Principles Through a Transdisciplinary Lens." *INCOSE International Symposium* 30 (1): 1517–32. https://doi.org/10.1002/j.2334-5837.2020.00801.x.
- Wheaton, Marilee J., and Azad M. Madni. 2018. "Model Based Tradeoffs for Affordable Resilient Systems." *INCOSE International Symposium* 28 (1): 30–39. https://doi.org/10.1002/j.23345837.2018.00465.x.
- Wolny, Sabine, Alexandra Mazak, Christine Carpella, Verena Geist, and Manuel Wimmer. 2020. "Thirteen Years of SysML: A Systematic Mapping Study." *Software and Systems Modeling* 19 (1): 111–69. https://doi.org/10.1007/s10270-019-00735-y.
- Wu, Quentin, David Gouyon, Sophie Boudau, and Éric Levrat. 2019. "Towards a Maturity Assessment Scale for the Systems Engineering Assets Valorization to Facilitate Model-Based Systems Engineering Adoption." *INSIGHT* 22 (4): 37–39. https://doi.org/10.1002/inst.12274.
- Xu, Peng, and Alejandro Salado. 2019. "A Concept for Set-based Design of Verification Strategies." *INCOSE International Symposium* 29 (1): 356–70. https://doi.org/10.1002/j.23345837.2019.00608.x.
- Younse, Paulo J., Jessica E. Cameron, and Thomas H. Bradley. 2021. "Comparative Analysis of a Model-based Systems Engineering Approach to a Traditional Systems Engineering Approach for Architecting a Robotic Space System through Knowledge Categorization." *Systems Engineering* 24 (3): 177–99. https://doi.org/10.1002/sys.21573.

ABOUT THE AUTHORS

Lorraine Brisacier-Porchon is a PhD student in systems engineering with ENSTA Paris. She obtained her Engineering degree in 2014 at ENSTA Bretagne, Brest. She applied SysML 1.3 for MBSE as a system architect on land defense systems during 7 years at Nexter Systems. She undertook the PhD project in 2020 after taking part in an international program development.

Omar Hammami is a professor at ENSTA ParisTech, France and an expert in complex systems and systems engineering. He has served as an expert for projects evaluation for several international organizations in the field of systems engineering with applications in transport and energy, organizations, and value chains in semiconductor and embedded systems. Omar Hammami has published over 200 papers in international conferences and journals in the fields of computer science and system engineering for complex systems. He is involved on a regular basis in the organization of several conferences and events worldwide related to large scale systems. He is involved in MENA region for large scale systems engineering projects. He holds a PhD in computer science from Toulouse University, France, and a HDR in physics from Orsay University, France.

His research interests are in theoretical foundations of systems engineering and multidisciplinary optimizations.

Exploring the Test and Evaluation Space using Model Based Conceptual Design (MBCD) Techniques

David Flanigan, david.flanigan@jhuapl.edu; and **Kevin Robinson,** kevin.robinson@shoalgroup.com
Copyright ©2019 by David Flanigan and Kevin Robinson. Permission granted to INCOSE to publish and use.

ABSTRACT
During the initial concept development phase, systems engineers focus on defining the problem space and system functions to explore candidate concepts that may address the systems engineers' problems. Model-based conceptual design (MBCD) techniques may be used to assist the customer and other stakeholders develop a greater understanding of the system concept, as well as identifying areas in the system that are affected by changes in requirements. This approach has generally been documented for describing the system concept in the early stages in the lifecycle, without significant focus on the test and evaluation (T&E) space that would be needed to evaluate these concepts, or identifying where the T&E space would be affected with a change in requirements. Our hypothesis is that decision makers would equally gain insight into the T&E considerations as well as system space considerations using MBCD techniques. An approach is offered to extend the previously published MBCD methodology to better consider the T&E space.

APPROACH / OUTLINE

Developing a system concept requires defining the problem space and required capabilities, functionality, and interfaces of the system concept space. A Model-based conceptual design (MBCD) technique can describe the linkage of these problems and potential solution space in order to visualize the impact of changes from the problem to the solution space, and vice versa. The MBCD approach is conveyed via a structured entity-relationship *descriptive* model instead of a traditional static document, which may promote rapid understanding of the causality of changes and may encourage quicker decision making and become informed of the problem space.

This paper offers an additional emphasis to the existing MBCD process by extending it to integrate test and evaluation (T&E) artifacts and interfaces more thoroughly to the operational domain, system domain, and analysis domain as previously described by Robinson et al. (2010). The paper starts by describing the motivation in incorporating the T&E domain to the existing methodology, and how the test domain artifacts can be modeled and analyzed, to quantify the impacts of changes to the other domains. For clarity of reading, an illustrative example is offered to explore the modified technique and offers examples of what these metrics may look like to provide insight to decision makers.

INTRODUCTION / MOTIVATION

The MBCD technique has been introduced to aid in understanding the problem space. Like existing model-based systems engineering (MBSE) techniques employed later in the lifecycle, it helps to visualize and structure systems engineering information. It allows for a richer visual picture to structure how changes in the capabilities / requirements may result from changes in the problem definition space, ultimately influencing the system capability space, concept of operations, or interfaces needed to successfully complete the mission. This approach can be helpful in the initial conceptual phase but does not currently consider in detail the T&E phase of the project during development of a conceptual system design. By including the information that more fully describes the T&E activities of the project, additional insight into the full system design may be considered, to include the requirements / capabilities to be tested as well as the complex test ranges and equipment to verify these requirements and changes in the requirements. Decision makers may receive equal insight into the entire system concept by incorporating the T&E elements into the MBCD process, as well as considering operating and system concepts.

LITERATURE REVIEW

MBCD is implemented through a series of models to provide communication between the various system development elements (developers, stakeholders, users, etc.) and is described by Wylie et al. (2016), and

Aluwihare et al. (2014) from which this paper takes motivation to extend the current methodology. Cook et al. (2015) uses the MBCD approach to assess the technical risk of concepts using modeling and the understanding of interdependencies between the different models, which can inspire the use of modeling to conduct analysis on the various concepts. Do et al. (2014) have used MBCD to explore the interactions that are needed when exchanging information and insights while executing contracts between the acquirer and supplier. Tetlow et al. (2013) utilize the MBCD approach to further explore the requirements and to assess the mission success of the conceptual system using a model-based approach. Do and Tetlow's descriptions detail the linkage between the user needs and system modeling, to develop a credible and valid system model for further analysis of the needs. Other uses of models to inform system simulations have been produced (Yaroker et al., 2013) that utilize a similar methodology as the MBCD approach.

It can be concluded from the literature review that MBCD has a wide range of applications and user communities, which provides motivation to incorporate the T&E community in this methodology.

DESCRIPTION OF THE APPROACH

The modified MBCD approach is described in five separate segments. The first defines how MBCD is used for system concept development and discusses the relevant artifacts, actors, and information. The second describes the proposed T&E extension to the MBCD technique. The third segment describes the linkage between the test domain and the other MBCD domains (notably the operational, system, and analysis). The fourth segment offers additional considerations to evaluate the entire system model. The last segment offers an approach to evaluate the new linkages and to visualize the insight gained when one domain causes changes to the other domains.

First Segment: MBCD Usage

MBCD is used to structure and link information about the understanding of a problem to possible solutions. Wylie et al. (2016) describe the usage of MBCD using *descriptive* models to describe the problem space, what the system is comprised of, and how the system interfaces are described. In their approach, they provide a logical design-based process to define the traceability, and therefore design rationale, between strategic guidance, operational activities, operational needs, functions, functional requirements, refined requirements, and software components. Through use of the descriptive models, the software developers

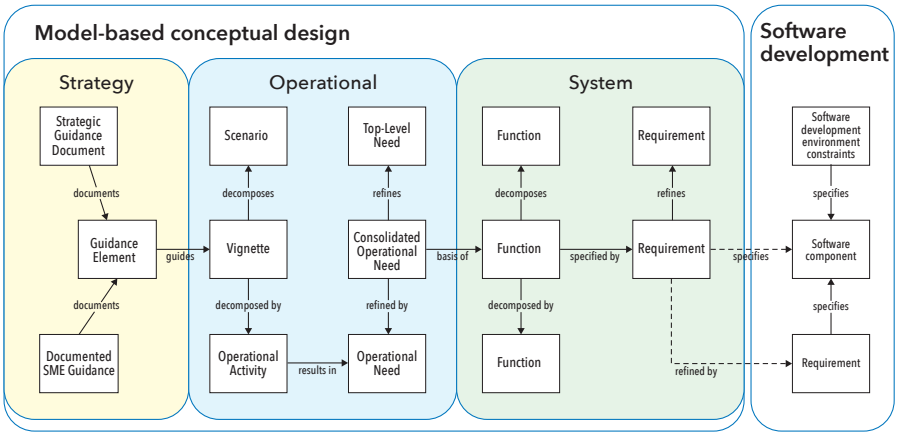


Figure 1. Traceability from conceptual design to software development (Wylie et al., 2016)

are then able to develop their model of the system, and how it traces to the previously described artifacts. This traceability visualization can then aid the software developers and decision makers to appreciate where changes in the modified artifacts could affect the current software development plans. This level of insight can assist the decision makers to address the right problem and assist the developers to focus on the right solution set. Figure 1 provides an example of this traceability between domains through an abstraction of the schema employed to structure the model.

Second Segment: T&E Extension

An additional domain is proposed for inclusion into the MBCD methodology to address the T&E domain. This includes information elements that would describe the activities needed to test the requirements and functions, trace the tests to the requirements, and include the system components that would need to be tested. Proposed elements of the test domain would include test plans, test ranges, test events, test articles, test targets, and test constraints. Based on the authors' experiences across the conceptual design and T&E domains, a high-level example of the schema of this test domain is provided in Figure 2.

Third Segment: Test-Domain Linkage to Existing MBCD Domains

The newly formed test domain model may be integrated into the MBCD model through the integration of the schemas. Robinson et al. (2010) define a model-based systems engineering approach to describe a complex capability to include the enterprise context, operational domain, system domain, and the analysis domain. The strategic domain (enterprise context) focuses on the guidance. The operational domain focuses on the mission tasks, operational environment, and service requirements. The system domain focuses on the functions needed to address the mission, as well as the specific components that perform the functions. The analysis domain supports the studies and analysis to analyze the operational and system domain. Figure 3 (next page) augments the existing schema of operational domain and system domain with the test domain, thus providing the framework for developing the enhanced descriptive model, including the T&E activities.

As more domains are included with the model, the abstracted schema represented in Figure 3 increases in complexity and becomes less readable. For clarity of reading, the interfaces and directionality from Figure 3 have been converted into an interdependency matrix, shown in Table 1.

The table is intended to be read from left to right, from the source node (row) to the target node (column). A number of "1" indicates there is an interface from that specific source to target node. Note that the directionality should be reflected in this matrix, as not all interfaces have a 2-way direction, although can if desired for usability

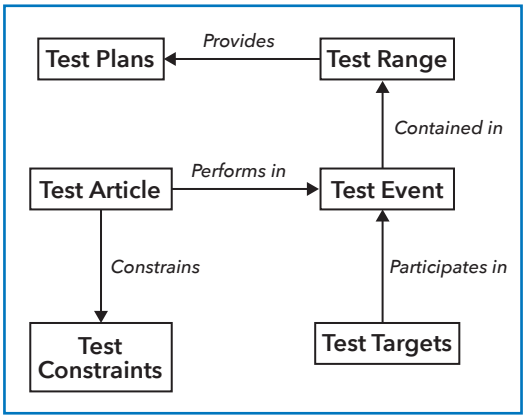


Figure 2. Test domain MBCD model

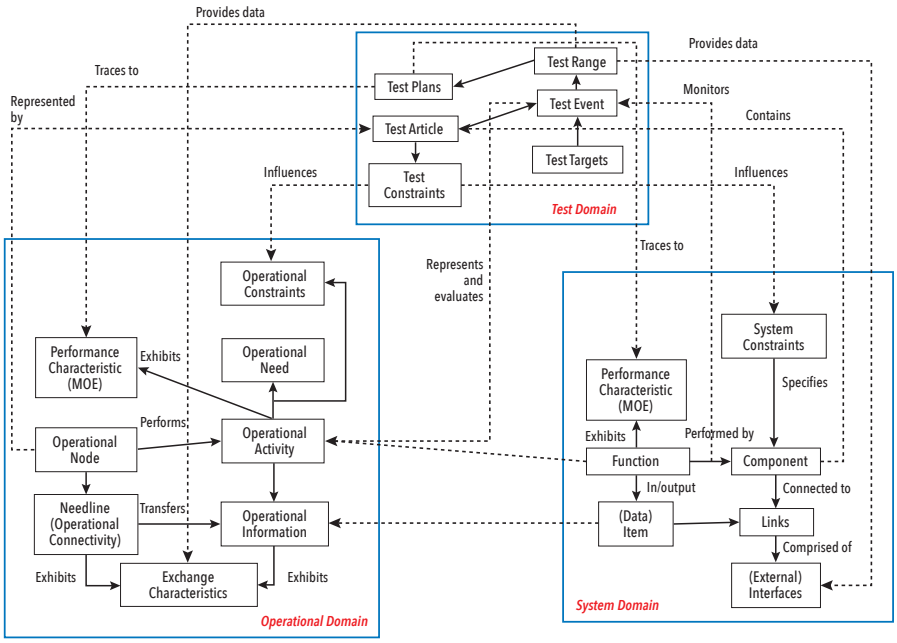


Figure 3. Modified MBCD model abstracted schema

and readability. This table shows the three domains (operational, system, and test), which each have three possible domain interactions (one internal and two external).

Fourth Segment: Evaluation of the New Linkage

The fourth segment evaluates how the rest of the overall descriptive model is affected when one domain element is changed. Changes may be viewed from different perspectives: the decision makers will view changes to the model as a change in capability or fielding date, which may affect their investment strategy. Developers may view changes to the model as changing their delivery dates or scheduling of efforts. Analysts may view changes to the model as updating their assessment of the system capability, which then may affect the decision maker's insight of the system's capability. Testers may view changes to the model that may affect their existing testing capabilities or future testing capabilities that need to be developed.

Table 1. MBCD Model interdependency

		Operational Domain								System Domain							Test Domain					
		Operational Contraints	Operational Need	Operational Activity	Operational Information	Exchange Characteristics	Performance Characteristic (MOE)	Operational Node	Needline(Operational Connectivity)	System Contraints	Component	Links	(External) Interfaces	Performance Characteristic (MOP)	Function	(Data) Item	Test Plans	Test Article	Test Constraints	Test Range	Test Event	Test Targets
Operational Domain	Operational Contraints																					
	Operational Need														1							
	Operational Activity	1	1				1															
	Operational Information					1																
	Exchange Characteristics																					
	Performance Characteristic (MOE)																					
	Operational Node			1					1									1				
	Needline(Operational Connectivity)				1	1																
System Domain	System Contraints										1											
	Component											1						1				
	Links												1									
	(External) Interfaces																					
	Performance Characteristic (MOP)																					
	Function			1							1			1		1					1	
	(Data) Item				1							1										
Test Domain	Test Plans						1							1								
	Test Article																		1		1	
	Test Constraints	1								1												
	Test Range												1				1					
	Test Event				1															1		
	Test Targets																				1	

Table 2. MBCD model interdependency primary and secondary impacts

		Operational Domain								System Domain						Test Domain						
		Operational Contraints	Operational Need	Operational Activity	Operational Information	Exchange Characteristics	Performance Characteristic (MOE)	Operational Node	Needline(Operational Connectivity)	System Contraints	Component	Links	(External) Interfaces	Performance Characteristic (MOP)	Function	(Data) Item	Test Plans	Test Article	Test Constraints	Test Range	Test Event	Test Targets
Operational Domain	Operational Contraints																					
	Operational Need														1							
	Operational Activity	1	1				1															
	Operational Information					1																
	Exchange Characteristics																					
	Performance Characteristic (MOE)																					
	Operational Node			1					1									1				
	Needline(Operational Connectivity)				1	1																
System Domain	System Contraints										1											
	Component											1						1				
	Links												1									
	(External) Interfaces																					
	Performance Characteristic (MOP)																					
	Function			1							1			1	1						1	
	(Data) Item				1							1										
Test Domain	Test Plans						1							1								
	Test Article																		1		1	
	Test Constraints	1								1												
	Test Range												1				1					
	Test Event				1															1		
	Test Targets																				1	

		Operational Domain								System Domain						Test Domain						
		Operational Contraints	Operational Need	Operational Activity	Operational Information	Exchange Characteristics	Performance Characteristic (MOE)	Operational Node	Needline(Operational Connectivity)	System Contraints	Component	Links	(External) Interfaces	Performance Characteristic (MOP)	Function	(Data) Item	Test Plans	Test Article	Test Constraints	Test Range	Test Event	Test Targets
Operational Domain	Operational Contraints																					
	Operational Need														1							
	Operational Activity	1	1				1															
	Operational Information					1																
	Exchange Characteristics																					
	Performance Characteristic (MOE)																					
	Operational Node			1					1									1				
	Needline(Operational Connectivity)				1	1																
System Domain	System Contraints										1											
	Component											1						1				
	Links												1									
	(External) Interfaces																					
	Performance Characteristic (MOP)																					
	Function			1							1			1		1						
	(Data) Item			1								1										
Test Domain	Test Plans						1							1								
	Test Article																		1		1	
	Test Constraints	1								1												
	Test Range												1				1					
	Test Event				1															1		
	Test Targets																				1	

Changes to one domain may affect other domains described in the model. For example, if there are changes to the operational domain (such as requirements), this may affect the system development efforts if there are new capabilities needed, or if the system design approach needs to be modified. As a result of this operational requirements change, testing approaches may need to be changed, which may affect the scheduling of the test facility or modification of the test articles or targets.

Fifth Segment: Impacts of the Changes

Once the MBCD model has been modified, evaluation of the model should be conducted to ensure that the MBCD concepts are still valid, and the decision makers and other actors gain insight into the problem when changes to one domain are introduced. Several means are offered to evaluate how the linkages may be conducted. One method could be to leverage the network science community, to describe the number of nodes that are affected by a node that will change (for example, changing requirements and understanding what impacts this change would have in the other domains). Other network science metrics are size, average degree, average path length, connectedness, node centrality, and node influence.

As the triad of systems, operational, and testing domains are affected by changes in one of the domains, we may observe a change in both the primary and secondary influences that a domain has on the rest of the system. Using Table 1, changes to the test domain could affect the system and operational domain as the primary influence. However, each of these domains has their own potential influences, creating a secondary influence (system domain may affect the test and operational domain, and operational domain may affect the system and test domain). There exists a potential for the primary change in one domain to indirectly affect itself through the primary domain influenced. It may be postulate that a lesser impact will be seen through the secondary domain effect but leave this for future work to quantify the primary and secondary impacts. Table 2 provides an example of such a primary (left side) and secondary (right side) of impacts based on one modification (function from system domain). An example of a primary impact is by affecting the "function" within the system domain, will affect five elements in the system (highlighted in orange). An example of the secondary effect is that each of these elements will have their own influence on the operational, system, and test domain, moving up and down the columns (shown on the right side in blue), affecting seven elements within the system.

Illustrative Example

An illustrative example is offered to evaluate if the modified MBCD technique has merit and offers additional value to the stakeholders when changes are introduced. Here an existing example that uses MBCD to evaluate fire and emergency services (Spencer and Harvey 2014) is leveraged and simplified. This example was developed for the Department of Fire and Emergency Services (DFES) of the Government of Western Australia. The MBCD process was

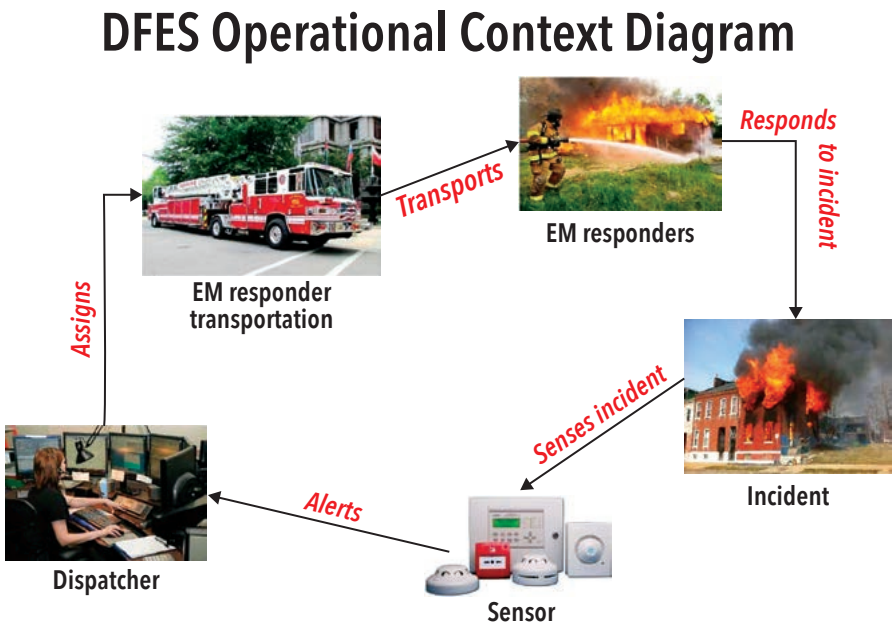


Figure 4. Modified DFES operational context diagram (OV-1)

Table 3. DFES domain elements

Functions	Systems	Actors
Sense smoke/incident	Sensor	Fire service personnel
Send alert	Telephone/radio	Rescue coordinators
Confirm incident/select action	Data Terminal	
Dispatch response units	Tanker, pumps, hoses	
Respond to incident	Transport vehicle	

Table 4. DFES test elements

Functions	Test Objective	Test Element
Sense smoke/incident	Determine if incident is properly detected	Sensors, fire source, facility environment
Send alert	Determine if alert is sent timely	Communications (transmitter and receiver), communications environment
Confirm incident/ select action	Determine if response is correctly determined	Dispatcher, displays, dispatcher environment
Dispatch response units	Determine if dispatch is correctly executed	Response units, transportation environment
Respond to incident	Determine if response is adequately executed	Responders, fire source, facility response environment

followed to define the DFES mission, drivers, and capabilities, and used the capability management framework to consider during the planning, development, and execution of the capability development more thoroughly.

This example is utilized to introduce the MBCD process and the test domain. The intent of the example is to exercise the interdependency and quantification of the impact of changes when portions of the entire model are changed.

Operational Domain Description

The operational domain is defined by the DFES mission to detect, analyze, and respond to emergencies and incidents. Depicted in Figure 4 is the mission in graphical form using an OV-1. Within each of these domains, the following elements are defined in Table 3.

System Domain Description

There are numerous systems that are used in this example. These are organized by the various phases of the operation (sensing, alerting, processing, dispatching, transporting, and responding). These systems are also listed in Table 3. Note that these systems will also include the actors that will operate the systems and other aspects not included in the simplified example.

Test Domain Description

The test domain will identify several elements that would be used to test the various mission activities that are being evaluated. From our example, four capabilities would be tested, listed in Table 4 that are organized by capability test objective, and applicable test elements.

Insight and Utility of the Modified MBCD Process

The modified model can be utilized to incorporate the test domain along with the operational and system domains. While the stakeholders, development team and test team are developing their respective efforts, we would expect numerous interactions between the three teams during the capability development. Expected questions in response to a domain change should start with "how does that affect the other domains?"

The model would be developed and then verify with the three domain teams to ensure that the elements and interactions are correct. Data would be elicited through tailored interviews and workshops to determine if sufficient insight was gained by all parties during the system development.

CONCLUSIONS / NEXT STEPS

This paper has offered a modification to the existing MBCD process to incorporate the test domain into the conceptual development phase. The aim being to ensure that the testing community and capabilities are also considered during the initial development to identify long-lead capability development, or how interdependent the operational and system development teams are to affect the test capabilities.

Next steps would be to identify an example project that this approach could be applied to and gain concurrence by all three domains. A model would be developed to describe the specific domains and follow the MBCD process during the system development lifecycle. Data could be collected at relevant milestones (for example, preliminary design review, critical design review, test readiness review, etc.). If the hypothesis proves correct that insight is gained by all domain stakeholders, the project could progress to a larger and more interdependent system concept for a further proof of concept. ■

REFERENCES

- Aluwihare, Chanaka, Michael Waite, and Christopher French. 2014. "Model Based Systems Engineering: a Methodology for Collaborative Requirements Engineering." Australian Defence Engineering Conference 2014. Engineers Australia.
- Cook, S. C., et al. 2015. "Evaluation of a Model-Based Technical Risk Assessment Methodology."
- Do, Quoc, Stephen Cook, and Matthew Lay. "An Investigation of MBSE Practices Across the Contractual Boundary." *Procedia Computer Science* 28: 692-701.
- Robinson, Kevin, et al. 2010. "Demonstrating Model-Based Systems Engineering for Specifying Complex Capability." 2010 Systems Engineering Test and Evaluation Conference: SETE. Engineers Australia.
- Spencer, Daniel, and David Harvey. 2014. "A Model-Based Approach to Capability in Fire and Emergency Services." 2014 Systems Engineering Test and Evaluation Conference: SETE 2014. Engineers Australia.
- Tetlow, M. et al. 2013. "Modelling Requirements for Mission Success Prediction." MODSIM2013, Adelaide, AU.

- Wylie, Matthew, David Harvey, and Tommie Liddy. 2016. "Model-Based Conceptual Design Through to System Implementation-Lessons from a Structured Yet Agile Approach." 2016 Systems Engineering Test and Evaluation Conference: SETE 2016. Engineers Australia.
- Yaroker, Yevgeny, Valeriya Perelman, and Dov Dori. 2013. "An OPM Conceptual Model-Based Executable Simulation Environment: Implementation and Evaluation." *Systems Engineering* 16 (4): 381-390.

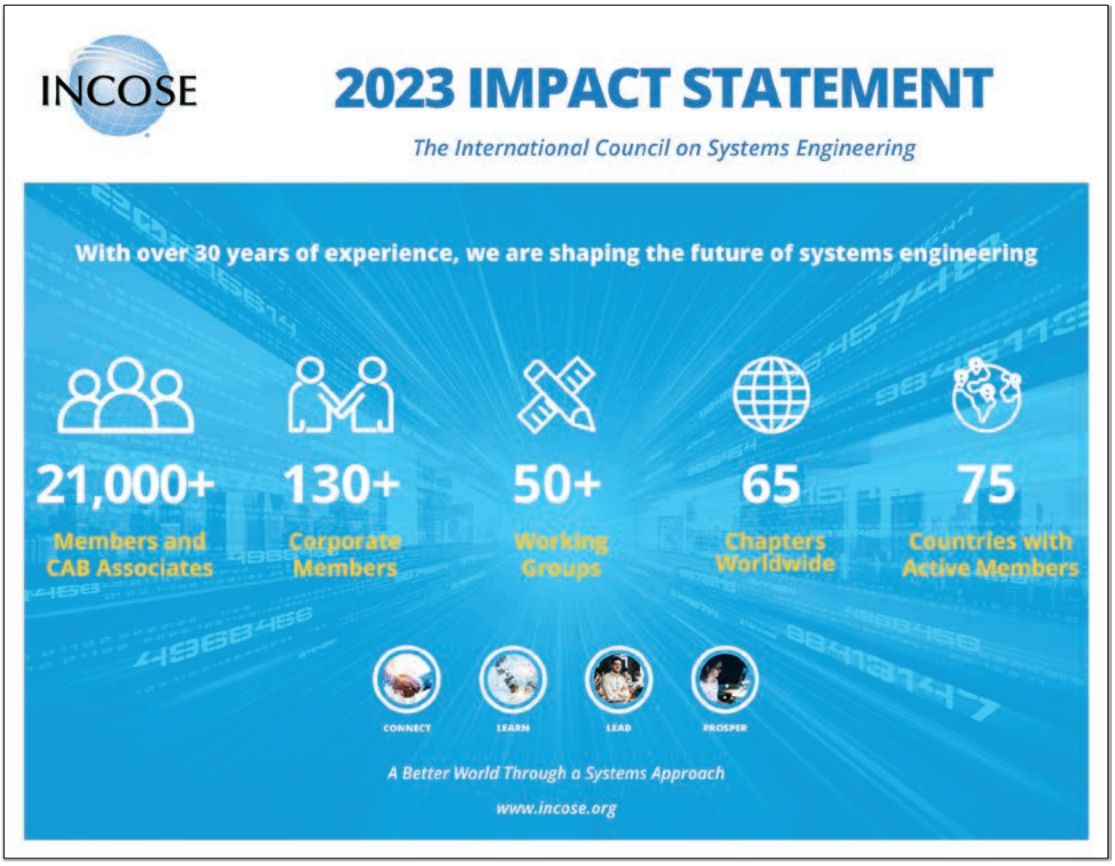
ABOUT THE AUTHORS

David Flanigan is a member of the principal professional staff for The Johns Hopkins University Applied Physics Laboratory, providing systems engineering services to various Department of Defense and Department of Homeland Security clients, and has 20 years of active duty and reserve service with the US Navy. A graduate of the University of Arizona, he holds a MS in information systems and technology, a MS in systems engineering from the Johns Hopkins University, and a PhD in systems engineering and operations research from George Mason University. Dr. Flanigan is a member of INCOSE, INFORMS, and MORS.

Kevin Robinson is the chief engineer at Shoal Engineering with a distinguished career in the field of guided weapons in both the UK Ministry of Defence and Australia's Department of Defence. He has made significant contributions to the development of advanced guided weapons through modelling and analysis, research, and leadership of large cross discipline teams. Throughout his career, Kevin has taken a leadership role in advancing the field of model-based systems engineering (MBSE) via his publications and contributions to the systems engineering community. He initiated and chaired Australia's first annual MBSE symposium, formed and chaired INCOSE's model-based conceptual design working group, delivered a keynote address to INCOSE's international symposium in 2016, and has made contributions to INCOSE's Systems Engineering Handbook and related standards. He has joined INCOSE's future of systems engineering (FuSE) initiative core team.

2023 INCOSE Impact Statement

A summary of INCOSE's achievements and plans for the future



Download your copy at incose.org/impact

Framework for Formal Verification of Machine Learning Based Complex System-of-Systems

Ramakrishnan Raman, ramakrishnan.raman@honeywell.com; **Nikhil Gupta,** nikhil.gupta4@honeywell.com; and **Yogananda Jeppu,** yogananda.jeppu@honeywell.com
Copyright ©2021 by Ramakrishnan Raman, Nikhil Gupta, and Yogananda Jeppu. Permission granted to INCOSE to publish and use.

■ **ABSTRACT**
A complex system is characterized by emergence of global properties which are very difficult, if not impossible, to anticipate just from complete knowledge of component behaviors. Emergence, hierarchical organization, and numerosity are some of the characteristics of complex systems. Recently, there has been an exponential increase on the adoption of various neural network-based machine learning models to govern the functionality and behavior of systems. With this increasing system complexity, achieving confidence in systems becomes even more difficult. Further, ease of interconnectivity among systems is permeating numerous system-of-systems, wherein multiple independent systems are expected to interact and collaborate to achieve unparalleled levels of functionality. Traditional verification and validation approaches are often inadequate to bring in the nuances of potential emergent behavior in a system-of-systems, which may be positive or negative. This paper describes a novel approach towards application of machine learning based classifiers and formal methods for analyzing and evaluating emergent behavior of complex system-of-systems that comprise a hybrid of constituent systems governed by conventional models and machine learning models. The proposed approach involves developing a machine learning classifier model that learns on potential negative and positive emergent behaviors, and predicts the behavior exhibited. A formal verification model is then developed to assert negative emergent behavior. The approach is illustrated through the case of a swarm of autonomous UAVs flying in a formation, and dynamically changing the shape of the formation, to support varying mission scenarios. The effectiveness and performance of the approach are quantified.

■ **KEYWORDS:** Complex System-of-Systems, Emergent Behavior, Machine Learning, Formal Verification

INTRODUCTION

A system can be considered as an integrated and interacting combination of elements and/or subsystems to accomplish a defined objective (INCOSE 2015). These elements may include hardware, software, firmware, and other support. Systems-of-systems (SoS) are systems of interest whose system elements are themselves systems (Jamshidi 2008). SoS has evinced keen interest among the systems engineering community, and there has been significant research pertaining to principles and practices on the architecture design, development, deployment, operation, and evolution of SoS (Lane 2013; Nielsen et al. 2015; INCOSE *INSIGHT* 2016; Raman and D'Souza 2018; and Raman and D'Souza 2019). Applications of SoS principles and practices span many domains, including electrical power distribution, and Internet-of-Things. SoS characteristics discussed in literature include operational/managerial independence, emergent behavior, and evolutionary development.

In a general sense, the adjective "complex" describes a system or component that by design or function or both is difficult to understand and verify. A complex system is characterized by emergence of global properties which are very difficult, if not impossible, to anticipate just from a complete knowledge of component behaviors (Aiguier et al. 2008). Emergence, hierarchical organization, and numerosity are some of the characteristics of complex systems (Ladyman et al. 2013). Specifically, for complex SoS, the "stringing" together of the constituent systems results in unique functionality and emergent behavior being exhibited at the SoS level that is very difficult to envision and predict and cannot be attributed to any of the constituent systems individually. Understanding measures of effectiveness (MOEs) (INCOSE 2015), is critical to analyze the impact of the emergent behavior at SoS level. There are different types of complexity measures dis-

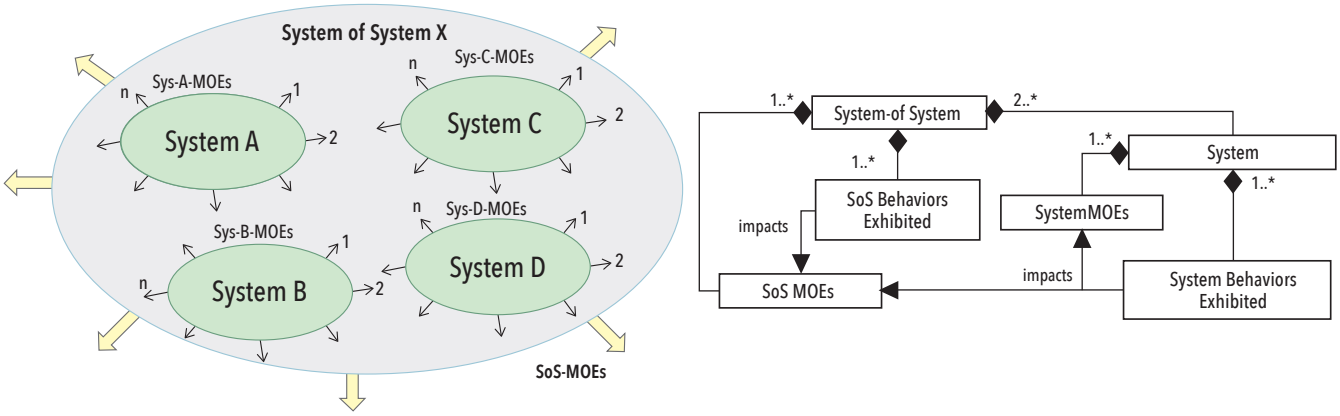


Figure 1. SoS MOEs and constituent system MOEs

cussed from different perspectives (Kinsner 2008). The perspective of complexity used in this paper is with respect to the degree of difficulty in accurately predicting the future behavior. This complexity is determined by the entity being observed, the capabilities of the observer, and the behavior that the observer is attempting to predict. This paper proposes an approach for analyzing and evaluating emergent behavior of complex SoS. In our proposed approach, the entity being observed is a complex SoS, the observer being a machine learning classifier, and the behavior being attempted to predict is the positive or negative emergent behavior of the complex SoS.

The rest of the paper is organized as follows: The next section discusses key elements pertaining to the proposed approach, namely emergent behavior, MOEs, machine learning, and formal methods. The subsequent section discusses the proposed approach and illustrates it through case of a complex SoS that comprise a hybrid of constituent systems governed by conventional and machine learning models. The case taken is a swarm of autonomous UAVs flying in a formation, and dynamically changing the shape of the formation, towards supporting different mission scenarios. Finally, benefits and limitations of the proposed approach, conclusions and future work are discussed.

EMERGENT BEHAVIOR, MOEs, MACHINE LEARNING, AND FORMAL METHODS

This section discusses some of the key elements pertaining to the proposed approach.

Emergent Behavior

Emergence refers to the ability of a system to produce a highly structured collective behavior over time, from the interaction of individual subsystems (Kinsner 2008). Common examples include a flock of birds flying in a V-formation, and ants forming societies of different classes of individual ants, wherein these patterns are not induced by a central authority. For a system, emergent behavior refers to all that arises from the set of interactions among its subsystems and components. Complex systems are expressed by the emergence of global properties which are very difficult, if not impossible, to anticipate just from a complete knowledge of component or subsystem behaviors (Giammarco 2017). Emergent behavior can be characterized as positive or negative, depending on the impact on the MOEs. The challenge for complex systems is that there is inadequate knowledge on combination of events that would result in a negative emergent behavior. The intent of our proposed approach is towards learning from emergent behaviors exhibited and asserting for occurrences of negative emergent behaviors for complex SoS.

Measures of Effectiveness – MOEs

MOEs. Measures of effectiveness are the operational measures of success that are closely related to the achievement of the objective of the system of interest, in the intended operational environment under a specified set of conditions (INCOSE 2015). It reflects the overall customer and user satisfaction, and it manifests at the boundary of the system. MOEs are independent of the specific solution (INCOSE 2005). Example of MOEs include service life of a satellite, search area coverage, and survivability. Failure of the system to meet an MOE implies that the system does not meet its purpose and objectives (Smith and Clark 2006).

SoS MOEs versus System MOEs. In the context of SoS, each constituent system of the SoS has its own MOEs. The MOEs for a constituent system can be independently measured to assess its success. MOEs of the SoS are the operational measures of success for the SoS as a whole. Figure 1 illustrates SoS MOEs versus constituent system MOEs. System A can have MOEs: SysA-MOE-1, SysA-MOE-2, and SysA-MOE-3. The MOEs of System A represent the measures of success for System A as an independent system, and the MOEs for System A can be independently measured to assess the success of System A. In addition to each constituent system having its own MOEs, MOEs are also relevant at the SoS level, that is, SoSx would also have its own MOEs. The MOEs at the SoS level

		MOEs of constituent Systems								
Relative importance of each SoS MOE										
System of System MoEs	SoS MoE Weight	SysA-MoE-1	SysA-MoE-2	SysA-MoE-3	SysB-MoE-1	SysB-MoE-2	SysC-MoE-1	SysC-MoE-2	SysD-MoE-1	SysD-MoE-2
SoS MoE1	9	9	9	7					1	
SoS MoE2	7			5	7	7	7		1	
SoS MoE3	5	7	9	5					7	
SoS MoE4	1	9	9	7				7	5	5
System A is a key player in the SoS	Raw score	125	135	130	49	49	49	7	56	5
	Relative Weight	21%	22%	21%	8%	8%	8%	1%	9%	1%
	Rank	3	1	2	5	5	5	8	4	9

Relative impact of each System MOE on each SoS MOE

Figure 2. SoS – System MOE relationship matrix

represent the measures of success for the SoS as a whole. Figure 1 further illustrates the impacts on MOEs at system level and at SoS level. The MOEs of the system are impacted by the behaviors exhibited by the system. Similarly, the MOEs of the SoS are impacted by the behaviors exhibited at SoS level. Further, the behaviors exhibited at constituent system level also impacts the SoS MOEs.

As discussed earlier, one of the characteristics of SoS is that the stringing together of the constituent systems results in unique behavior and functionality that gets exhibited at the boundary of the SoS, that is, the behavior may not be attributed to any of the constituent systems functioning independently. With this being the case, the relationships between the MOEs of the SoS vis-à-vis the MOEs of the constituent systems might turn out to be complex and dynamic. There are different means to analyze the MOE relationships between the constituent systems and SoS. SoS-System MOE relationship matrix (Raman and D'Souza 2017; and Raman and D'Souza 2019) is one of the means to analyze the relationships, as indicated in Figure 2. The impact of different system MOEs on the SoS MOEs could vary. There might be scenarios where a specific constituent system might be meeting all its MOEs, but the SoS MOEs might not be met. Similar scenarios will be discussed in this paper in the next section.

Machine Learning

Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions (Mohri et. al. 2012). Here, experience refers to the past information available to the learner, which typically takes the form of electronic data collected and made available for analysis. Machine learning represents the field of study that allows computer programs to learn without being explicitly programmed. The often-used definition is: "A computer program is said to learn from experience E with respect to some task T and some performance P, if its performance on T, as measured by P, improves with experience E" (Mitchell, 1997). Artificial neural networks (NN), inspired by biological neural networks in brains, comprise a collection (organized in layers) of interconnected units (nodes), with each node having the capability to receive a signal, process the signal, and transmit the processed signal to other units linked to it. NN has been used on numerous learning problems, including vision, speech recognition, social networks, board games, and medical diagnosis. In a neural network, the first layer is termed the input layer since it is connected to the external input data. The last layer is termed the output layer since it provides the outputs of the total neural network. All the other intermediate layers are termed hidden layers. Each node unit processes the signal via an activation function. Each input has a weight that can be modified. Each unit computes the activation function f of the weighted sum of its inputs.

Recently, there is an explosion in the adoption of neural network-based machine learning models in various systems and are increasingly being used to control many physical systems, such as cars and drones. A good starting point to get the context of the work discussed in this paper is a comprehensive survey paper that provides a detailed look at the field with a review of over 150 odd papers (Xiang et al. 2018) that discusses the use of neural network-based machine learning techniques in safety control systems, and the formal methods/ verification used to validate the networks. The verification of NN is a hard task as it is said to be an NP complete problem. Most of the difficulties arise from the presence of activation functions and the complex structure of the neural network. Nevertheless, neural networks-based machine learning techniques have been used in some of the safety critical systems: F-15B intelligent flight control system (William-Hayes 2005) and intelligent autopilot system (Baomar and Bentley 2017). Neural networks are however susceptible to small changes in their inputs, and therefore ensuring their correct behavior under various conditions is very important. The Reluplex algorithm, which stands of ReLU with Simplex caters to the activation function Relu using the simplex algorithm, is evaluated on a set of 45 real world NN problems (Katz 2017). In this paper, we have used MathWorks® MATLAB R2020b Statistics and Machine Learning Toolbox™.

Formal Methods

Formal methods are mathematics-based techniques for the specification, development, and verification of digital systems (RTCA 2011). The mathematical basis of formal methods consists of formal logic, discrete mathematics, and computer-readable languages. The use of formal methods is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analyses can contribute to establishing the correctness and robustness of a design. Formal methods can be used to model complex systems as mathematical entities. The complex system behavior is broken down into smaller units and each one of these is defined as mathematical equations. Defining systems formally enables system validation (mathematically correct behavior – mostly safety criteria) using means other than testing – like a proof of correctness. The mathematical techniques are used to prove the correctness of the assumptions and theory using property proving. There are many tools that can be used for formal methods in the systems development V-Model (Nanda et al. 2018). Another branch of formal verification is called model checking, which involves a model of the system and a way to define the property of the system. The model checking tool then explores the possible states the model can be in and checks for violations of the property. A violation of the property yields a counter example that is used for debugging the model. It may give concrete evidence of the correctness of the property and this proves that the property can never be violated for any combination of states and within the overriding assumptions. There is a possibility of the formal methods tool to provide an outcome stating that the property cannot be proved due to the limitation of the tool. This usually happens due to the large state space that is created and makes the proving impossible given the memory limitation of the computing platform. In such cases one must slice the model or limit the input space to reduce the bloat-up of the state space. In this work, we have used Simulink Design Verifier, a tool from MathWorks (SLDV 2020) that uses formal methods to generate test cases, find design errors and to prove the correctness of assertions or properties defined as Simulink blocks or MATLAB code. We have successfully demonstrated the use of SLDV in our earlier work (Raman and Jeppu 2019). In this paper, we also look at another tool called CBMC (2020). This tool is a bounded model checker that looks at properties in a small defined region and bound and argues on its correctness. CBMC works on the C code, and it has additional features like MC/DC testing, array checks, branch coverage etc. that can be used on the generated code. We explore the use of CBMC in the current problem statement to look at the NN correctness and the SoS behavior.

PROPOSED APPROACH

This section discusses the proposed framework towards application of machine learning based classifiers and formal methods for analyzing and evaluating emergent behavior of complex system-of-systems. Figure 3 provides an overview of the proposed approach. The complex SoS has a set of defined MOEs. The SoS comprises independent constituent systems, with each having their own corresponding system MOEs. The proposed approach involves

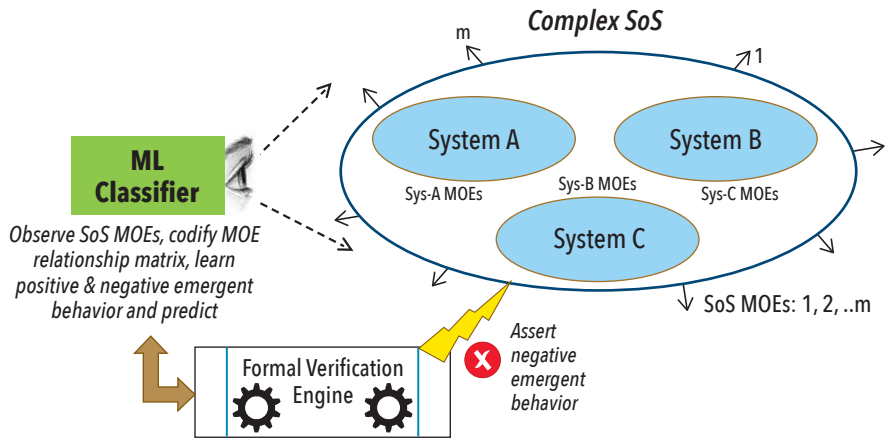


Figure 3. Overview of proposed approach

building a machine learning (ML) classifier that observes the various MOEs at SoS level and constituent system level, leverages the MOE relationship matrix (Figure 2), and learns the emergent behavior. The formal verification engine is used to assert the occurrence of negative emergent behavior. Figure 4 provides details of the proposed approach. To illustrate the proposed approach, the generic case of a swarm of UAVs is taken from publicly available literature (Tahir et al. 2019; X. Dong et al, 2019; and Hassanien and Emary 2016). This is specifically done to avoid any restrictions that may come in sharing the work with the community. The example, though generic, is sufficiently complex and is of significant relevance to the aerospace community and can further be scaled up to serve more complex use cases.

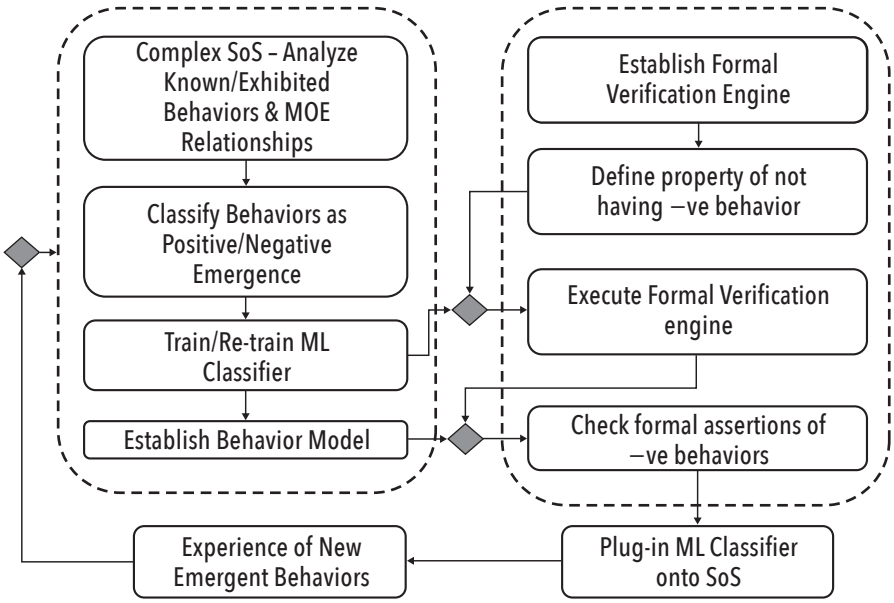


Figure 4. Proposed framework

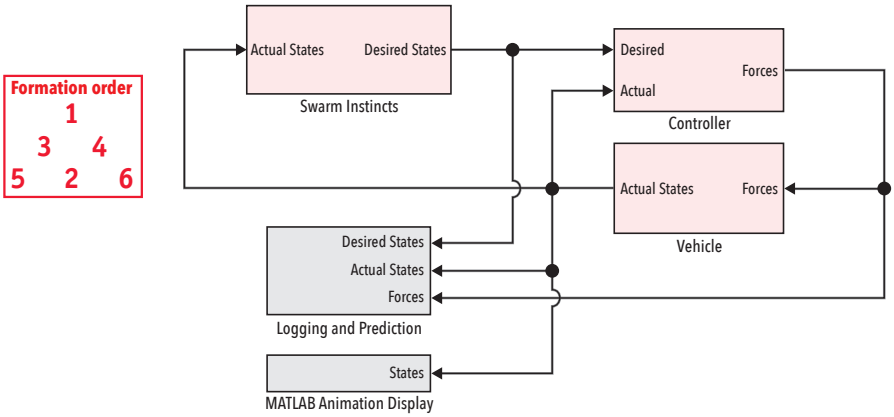


Figure 5. SoS model – with 6 constituent systems (autonomous UAVs)

Swarm Formation Flying

As discussed in the earlier section, emergence refers to the ability of a system to produce a highly structured collective behavior over time, from the interaction of individual subsystems. A typical example pertains to a flock of birds flying in a specific formation, which is supposed to give many benefits including reduced energy requirements and safety. Similar approaches have been adopted for swarm of UAVs too, wherein specific formation shapes are expected to provide specific benefits – including fuel savings and redundancy in mission coverage. The scenario being simulated is a system-of-systems comprising six autonomous UAVs as the constituent systems. The autonomous UAVs collaborate with each other during the required situations and fly in a formation to leverage the desired benefits as and when required. Various scenarios pertain to the different formation shapes as required for the mission. For an individual constituent system UAV, the MOEs would pertain to parameters such as whether the required speed constraints are adhered to or not, and whether the required space constraints with adjacent autonomous UAVs are adhered to or not. For the SoS comprising the various autonomous UAVs, MOEs would pertain to aspects such as the time duration to transition from one formation shape to another, the safety and separation constraints being adhered to by all the autonomous UAVs, and the formation shape being maintained without distortion. For experiments, simulations were done to study various scenarios that would be encountered during the autonomous UAV formation flying. MathWorks® MATLAB R2020a Aerospace & Control System Toolbox was used to build the models for the same.

The high-level SoS model is illustrated in Figure 5. The formation shape and indexing for the six autonomous UAVs is indicated in the figure (in "formation order"). For the specific formation, UAV-1 is considered as the leader of the formation. Each signal line depicted in the model represents data from all the six UAVs. The states of each individual UAV include the inertial position and velocities. The states are initialized in the vehicle block with random initial positions with respect to the leader of the formation.

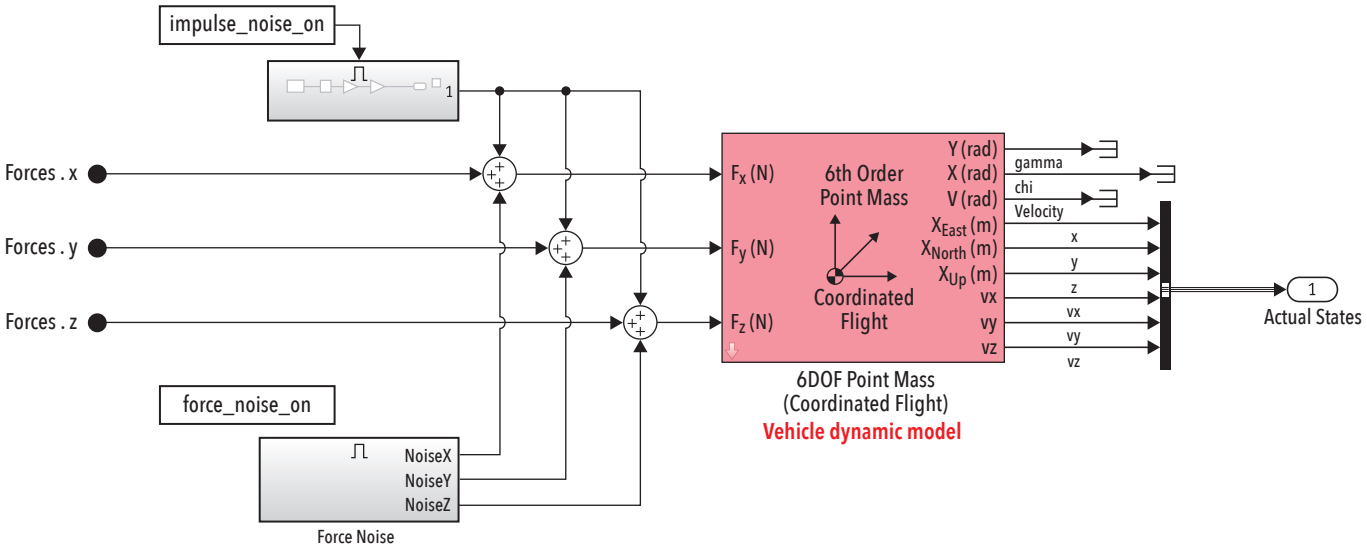


Figure 6. Constituent system model – an autonomous UAV

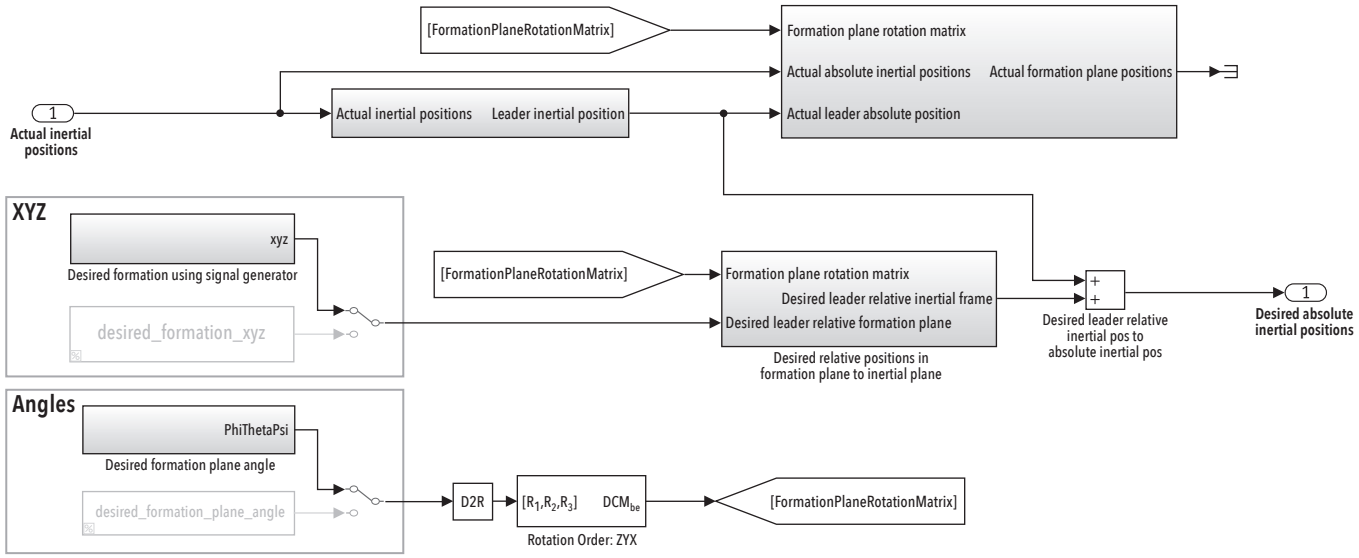


Figure 7. SoS – autonomous UAVs formation

The states are consumed by the swarm instincts block to generate desired states to maintain or change the formation. The desired states are passed onto the controller which generates required forces to be applied on the vehicle.

The required forces are then sent to the vehicle, which integrates them to get the current states. MATLAB animation display was used for visualizing the formation in real time during the various simulation runs. The constituent system autonomous UAV vehicle block is illustrated in Figure 6. The forces from the controller block are integrated within the 6-degree of freedom (6DOF) point mass block to generate the inertial position and velocities in ENU (east north up) frame. Impulse noise block introduces an impulse force at a random instant and of random total magnitude. This was used to simulate various scenarios such as wind gust. The force noise block adds a Gaussian noise to all the forces to simulate real world conditions. Figure 7 illustrates the model that pertains to the various formation shapes as part of the simulation runs. There are three frames of relevance in the model: (a) leader relative formation plane frame, in short, referred to as the formation plane (b) leader relative inertial frame, and (c) absolute inertial frame. The desired shape of formation and formation plane angles are taken as input, either from a time series file or from a signal generator. The desired formation plane angle is defined by Euler angles, which is consumed as a time series file or from a signal generator. The angles are converted to DCM (direction cosine matrix) in ZYX (axis) order. The desired positions in formation plane are transformed to the leader relative inertial frame using the DCM of formation plane. Leader relative inertial positions are converted to absolute inertial by adding leader's inertial frame position. Figure 8 illustrates the individual controller model that resides in each of the autonomous UAVs. The controller generates the desired forces to achieve the desired states from the actual current states. The cascaded proportional–integral–derivative (PID) is used to control the outer loop positions and the inner loop velocity. Saturation is added to the outputs to limit the control corrections to realistic values. Some of the formations are illustrated in Figure 9, along the rolling, pitching and yawing planes.

Design of Experiments

An orthogonal array of experiments is devised to analyze the behavior of the SoS for different values of various state parameters, as indicated in Figure 10. Following

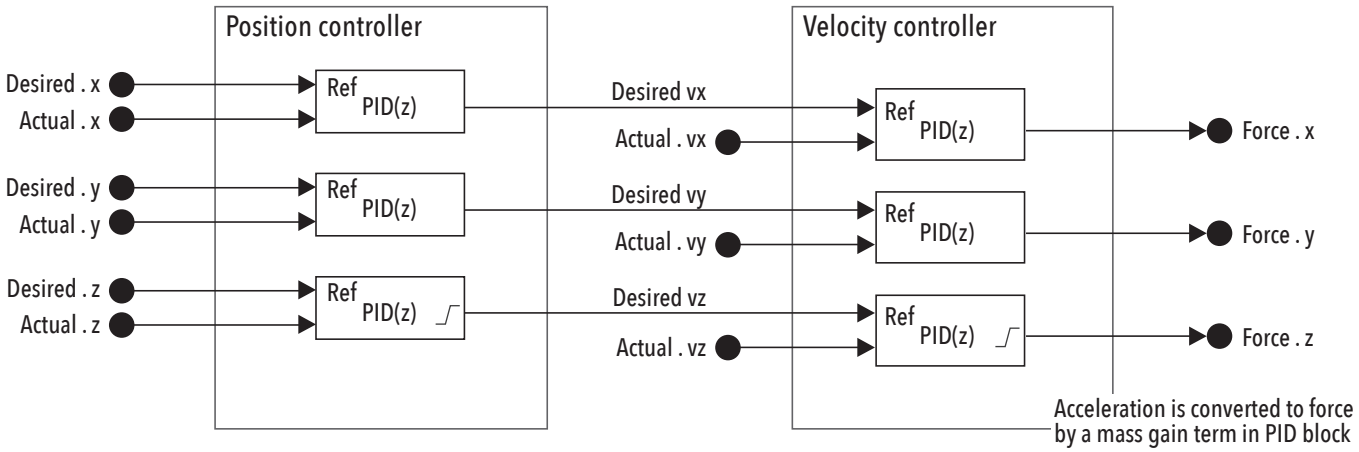


Figure 8. Constituent system: autonomous UAV – controller

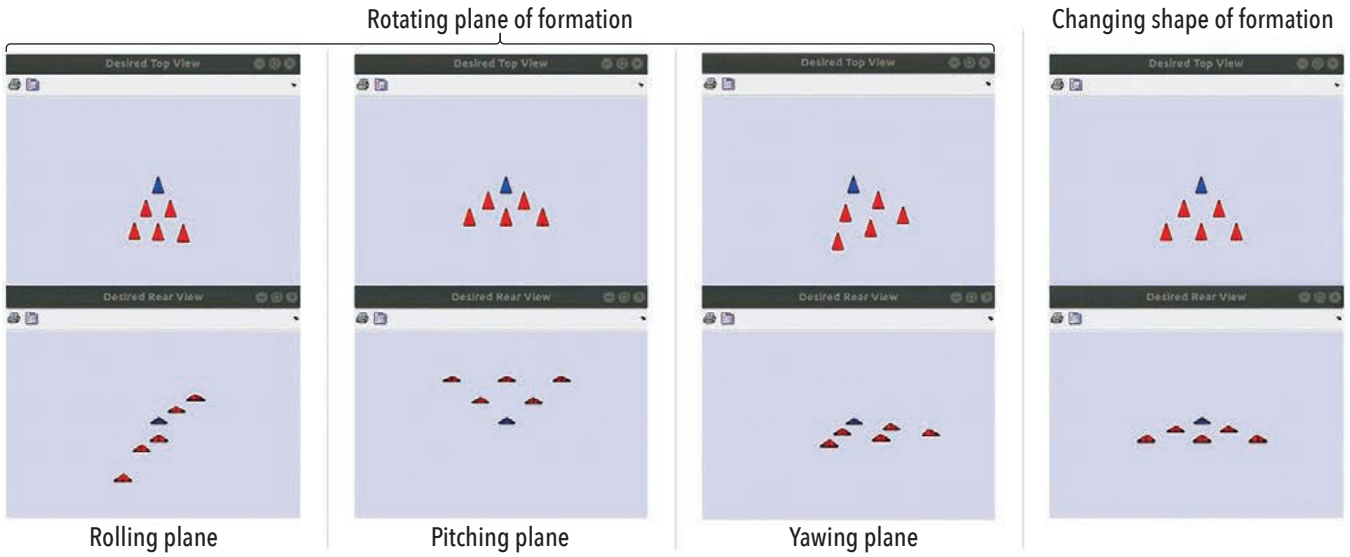


Figure 9. Various UAV formations in different planes

parameters are considered for behavior analysis: (1) initial / final formation shape, (2) angles of rotation of formation's plane of reference, as defined by Euler angles phi ϕ, theta θ, psi ψ, (3) vehicles facing wind gust, and the corresponding magnitudes on X,Y,Z axes, with/without forces.

Principal Component Analysis (PCA) and Zone Visualization

There are many factors that impact the emergent behavior of SoS, and visualization of the SoS state parameters would provide deeper insights. However, to understand the interplay of the various factors on SoS emergent behavior, a higher dimensional visualization is required which would be difficult to represent. Principal component analysis (PCA) (Martınez and Kak 2001) is used towards getting lower dimensional views. PCA comprises projection of an n-dimensional input data onto a reduced k-dimensional linear subspace, such that the reconstruction error is minimized. The lower-dimensional view is a projection of various points in the multi-dimensional space when viewed from its most informative viewpoint. PCA can be done by singular value decomposition of a data matrix, after mean centering, and normalizing the data matrix for each attribute.

Figure 11a illustrates the plot of PCA of UAV pair-wise distances (15 pairs between the five UAVs), against the PCA of UAVs pair-wise slopes (15 pairs between the five UAVs, with two slopes along XZ and YZ). The figure essentially represents the state of the SoS, reducing the multi-dimensional state parameters to lower dimensions, enabling identification of specific regions/ zones of positive (1) and negative (0) emergence. Further, the state space of a constituent system can be analyzed against the state space of the SoS, with respect to the emergent behavior and the MOEs. Figure 11b illustrates the plot of PCA of SoS versus PCA of UAV#3 MOEs. The following 3 scenarios are depicted in the figure: the zone of both SoS and UAV#3 exhibiting bad behavior (legend 0 in the plot, red); the scenario of UAV#3 meeting its own MOEs, but SoS is exhibiting negative behavior (legend 1 in the plot, yellow); and finally, the scenario of both UAV#3 and the SoS exhibiting positive behavior (legend 3 in the plot, green). Further the scenario of UAV#3 not meeting its MOEs while the SoS exhibiting positive behavior does not occur. This implies scenario wherein the constituent system is a key player in the SoS (scenario depicted in MOE relationship matrix in Figure 2). Three different supervised learning classification algorithms were tried (a) naïve Bayes classification, (b) fitted binary classification decision tree, and (c) KNN-nearest neighbor (Mitchell 1997). The decision surface of these different classification algorithms is illustrated in Figure 12, matching well with Figure 11b (legend used in Figure 12 is same as used in Figure 11b).

Machine Learning Model – ML Classifier

In the simulation runs of the complex SoS comprising autonomous UAVs, various state parameters of the formation are logged —

Experiment #	Initial Shape	Final Shape	Final psi	Final theta	Final phi	Gust UAVs	Gust X (kN)	Gust Y (kN)	Gust Z (kN)	Force on/off
1	Triangle	Triangle	-45	-45	-45	0	5	5	5	0
2	Triangle	Triangle	-45	-45	0	UAV3	2.5	2.5	2.5	1
3	Triangle	Triangle	-45	-45	-45	UAV 3 and 6	1.6	1.6	1.6	1
4	Triangle	Inv. Triangle	0	0	-45	0	5	2.5	2.5	1
5	Triangle	Inv. Triangle	0	0	0	UAV3	2.5	1.6	1.6	1
6	Triangle	Inv. Triangle	0	0	45	UAV 3 and 6	1.6	5	5	0
7	Triangle	Closed Loop	45	45	-45	0	5	1.6	1.6	1
8	Triangle	Closed Loop	45	45	0	UAV3	2.5	5	5	0
9	Triangle	Closed Loop	45	45	45	UAV 3 and 6	1.6	2.5	2.5	1
10	Inv. Triangle	Triangle	0	45	-45	UAV3	1.6	5	2.5	1
11	Inv. Triangle	Triangle	0	45	0	UAV 3 and 6	5	2.5	1.6	0
12	Inv. Triangle	Triangle	0	45	45	0	2.5	1.6	5	1
13	Inv. Triangle	Inv. Triangle	45	-45	-45	UAV3	1.6	2.5	1.6	0
14	Inv. Triangle	Inv. Triangle	45	-45	0	UAV 3 and 6	5	1.6	5	1
15	Inv. Triangle	Inv. Triangle	45	-45	45	0	2.5	5	2.5	1
16	Inv. Triangle	Closed Loop	-45	0	-45	UAV3	1.6	1.6	5	1
17	Inv. Triangle	Closed Loop	-45	0	0	UAV 3 and 6	5	5	2.5	1
18	Inv. Triangle	Closed Loop	-45	0	45	0	2.5	2.5	1.6	0
19	Closed Loop	Triangle	45	0	-45	UAV 3 and 6	2.5	5	1.6	1
20	Closed Loop	Triangle	45	0	0	0	1.6	2.5	5	1
21	Closed Loop	Triangle	45	0	45	UAV3	5	1.6	2.5	0
22	Closed Loop	Inv. Triangle	-45	45	-45	UAV 3 and 6	2.5	2.5	5	1
23	Closed Loop	Inv. Triangle	-45	45	0	0	1.6	1.6	2.5	0
24	Closed Loop	Inv. Triangle	-45	45	45	UAV3	5	5	1.6	1
25	Closed Loop	Closed Loop	0	-45	-45	UAV 3 and 6	2.5	1.6	2.5	0
26	Closed Loop	Closed Loop	0	-45	0	0	1.6	5	1.6	1
27	Closed Loop	Closed Loop	0	-45	45	UAV3	5	2.5	5	1

Figure 10. Design of experiments (DOE)

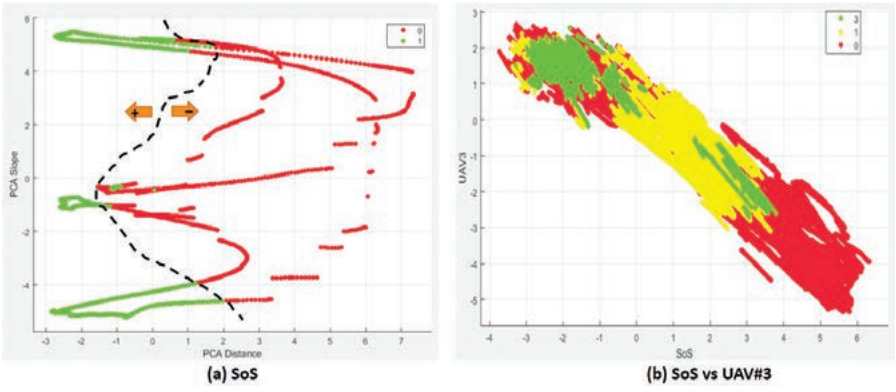


Figure 11. PCA analysis and visualization

including distances and bearings between each of the UAVs. The scenarios are labelled as "good"(1) or "bad"(0) based on the behavior seen at the SoS level. These labelled scenarios are fed into a neural network, and supervised learning algorithms were devised so that the network learns on the positive and negative emergent behaviors. Figure 13 illustrates the neural network-based ML model. The number of hidden layers, and number of units in the hidden layer defines the topology of the network. The inputs to the ML model are the various pairwise Euclidean distances, YX slopes (β) and ZX slopes (α) in the swarm, as illustrated in Figure 14. The trend of the parameters for a window of 4-time steps is provided as the learning data set. The data set is

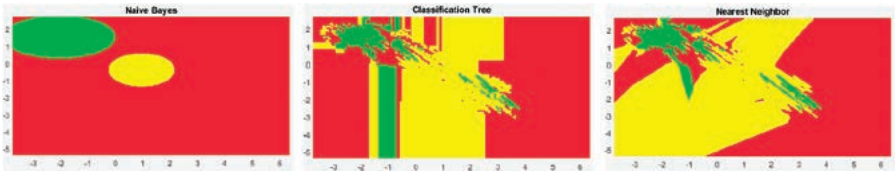


Figure 12. Classification of SoS versus UAV#3 behaviors

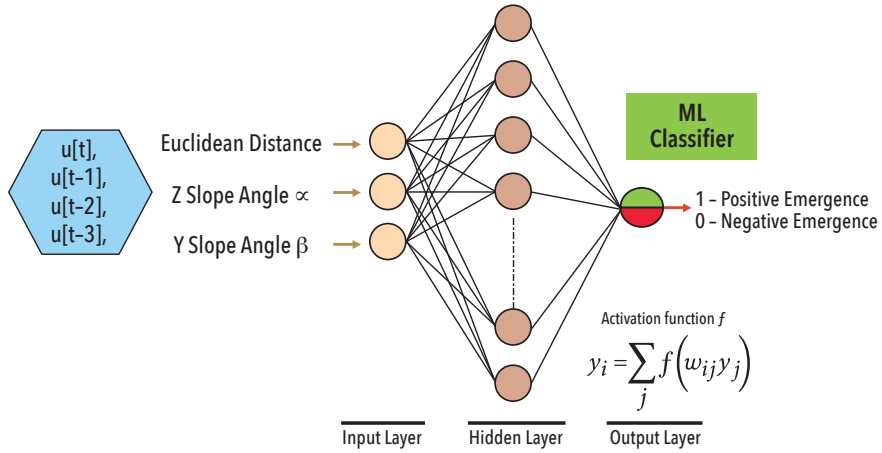


Figure 13: Machine learning ML classifier

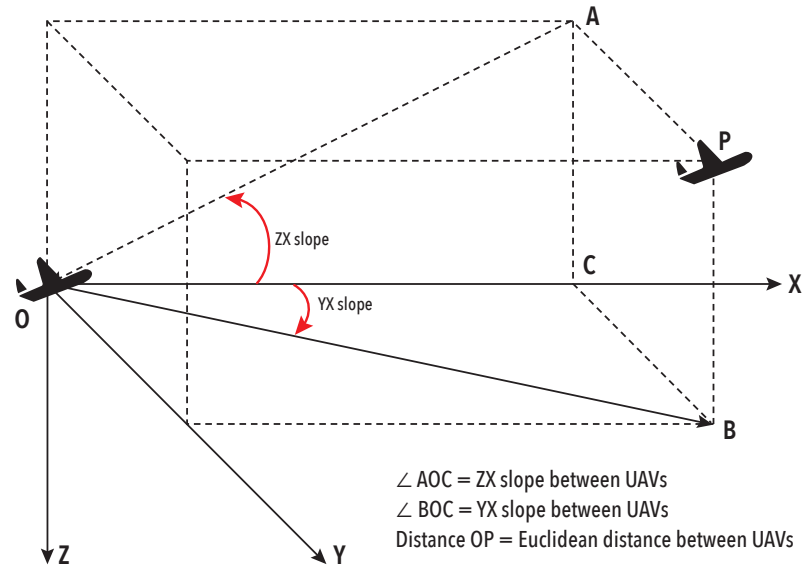


Figure 14. UAV formation – state parameters w.r.t pairs of constituent systems

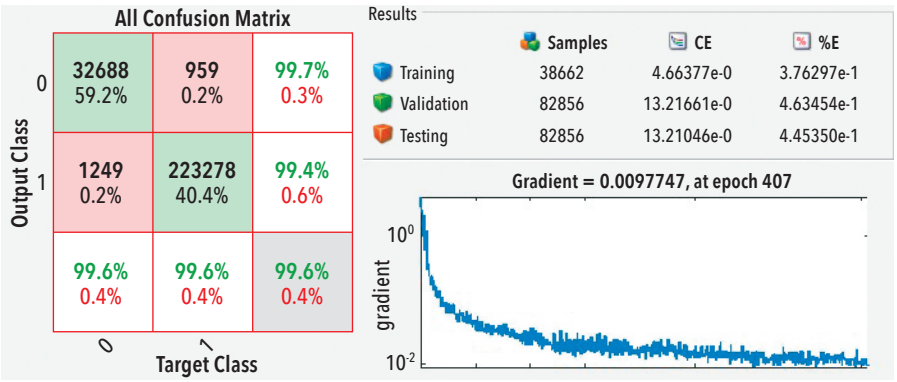


Figure 15. ML classifier – training performance

further split into training set, validation set and test sets. Typically, the training set is used to fit the model, while the validation set is used to estimate prediction error for model selection. The test set is then used for assessment of the generalization error of the final chosen model. Learning algorithms are devised that can automatically tune (and learn) the weights and biases so that the output produced by the network closely matches the desired output. Mathematically, this close matching involves an associated cost function that needs to be minimized. Hence, the training process is iterative, to minimize the cost function below a threshold, with each iteration fine tuning the parameters. The iteration concludes once the cost function reaches the minima, below the expected threshold. The weights/ biases thus learned by the neural network at the end of the iteration represents the parameters that can be used to directly transform the inputs to outputs. The data set comprising over half a million records is split between training (75%), testing (15%), and validation (15%) sets.

The performance of the learning is assessed in terms of cross entropy function, wherein minimizing the cross-entropy (CE) leads to better classifiers. Figure 15 illustrates the ML classifier training performance, for the scaled conjugate algorithm. The confusion matrix indicated in the figure illustrates the accurate and inaccurate classifications. The rows correspond to the predicted class (output class) and the columns correspond to the true class (target class). The diagonal cells (green color) indicate the correctly classified observations. The off-diagonal cells (light rose color) are the incorrectly classified observations. Both the number of observations and the percentage of the total number of observations are shown in each cell. The column on the far right of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified. These metrics are the precision (or positive predictive value) and false discovery rate, respectively. The row at the bottom of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified. These metrics are often called the recall (or true positive rate) and false negative rate, respectively. The cell in the bottom right of the plot shows the overall accuracy. As seen, overall, the prediction accuracy performance achieved is 99.6%.

ML Classifier – Behavior Predictions

The trained ML classifier is used to observe the SoS behavior and predict positive and negative emergence. The ML classifier is plugged onto the SoS model as

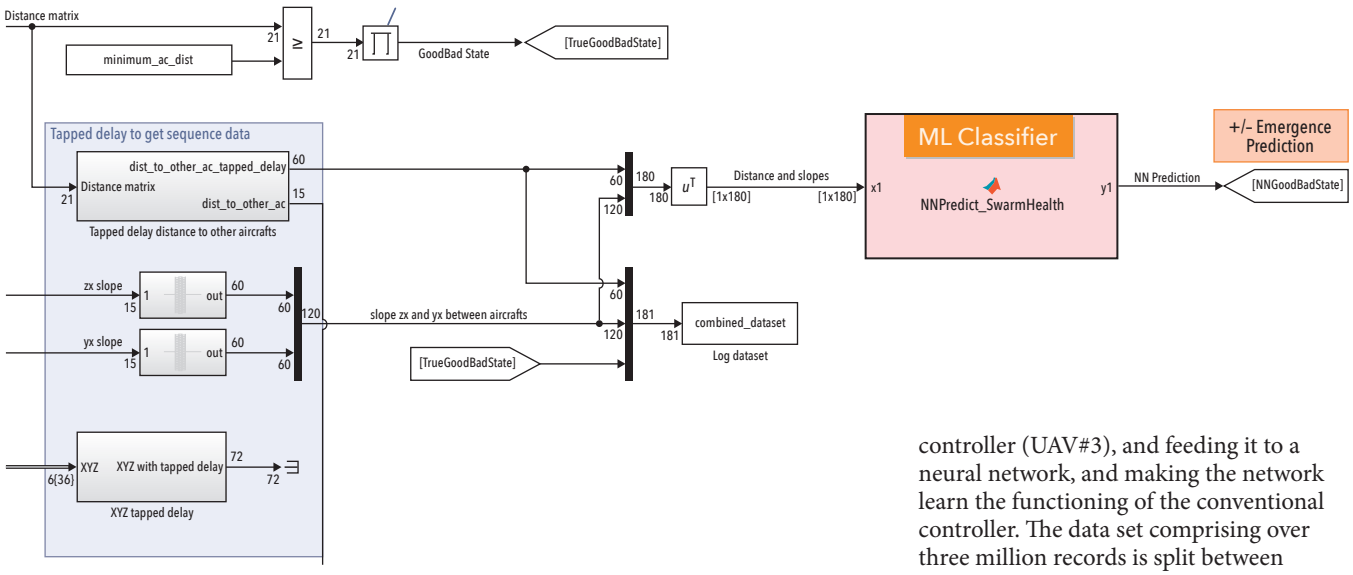


Figure 16. ML-classifier integrated into SoS model

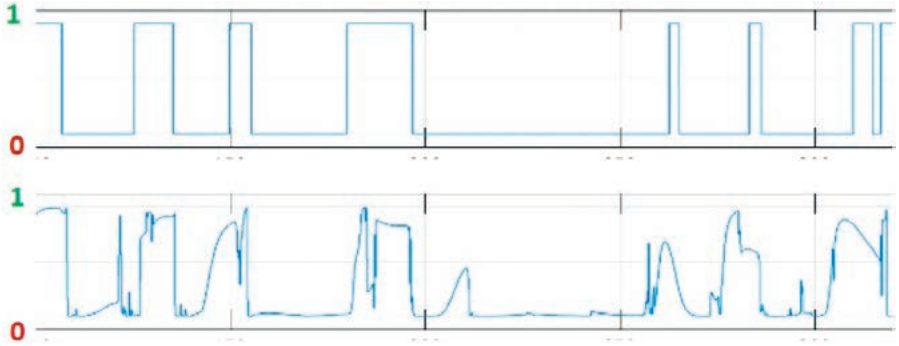


Figure 17. Machine learning model predictions of emergent behavior. (top) Sos Behavior Labelled on tiime steps as "good" (1) or "bad" (0); (bottom) ML Classifier predicted behavior probabilities, between 1 ("positive emergence) and 0 ("negative emergence")

illustrated in Figure 16, to understand and predict the emergent behaviors as positive and negative behaviors. Figure 17 indicates snapshots of the tool scope monitor, wherein values closer to 1 indicates positive emergent behavior being exhibited, while values closer to 0 indicates negative emergent behavior.

SoS with Hybrid (conventional + machine learning) Constituent Systems

The scenario of the SoS comprising a mix of conventional constituent systems and machine-learning model-based constituent systems is then studied. Towards this, the conventional controller (Figure 8) is replaced with a machine learning based NN controller for UAV numbered #3 (Figure 5). The NN controller (Figure 18) is built by tapping the data from the conventional controller (UAV#3), and feeding it to a neural network, and making the network learn the functioning of the conventional controller. The data set comprising over three million records is split between training (75%), testing (15%), and validation (15%) sets. The learning is stopped at epoch of 1000. The R value in this case has reached only about 0.4 after completion of 1000 epochs (a robust learning would imply an R value very close to 1). This learning is stopped to serve the purpose of dealing with a system that can exhibit negative emergent behavior at times. Figure 19 illustrates the scenario of the formation flying with the NN controller managing UAV #3 included in the formation. As indicated, UAV#3 misbehaves (that is, exhibits an oscillating behavior). This misbehavior is predicted by the ML classifier, indicating a negative emergence for the SoS.

Formal Methods

We have used formal methods for model checking in the earlier work (Raman and Jeppu 2020). We could optimize for time by changing the sampling time and abstracting the behavior of collision in the collision avoidance problem. We tried the same approach of using the Simulink design verifier on the 6 UAVs maneuvering together. The assertion was that once the maneuver occurs the UAVs will not collide while they maneuver. During the maneuver, a wind gust can

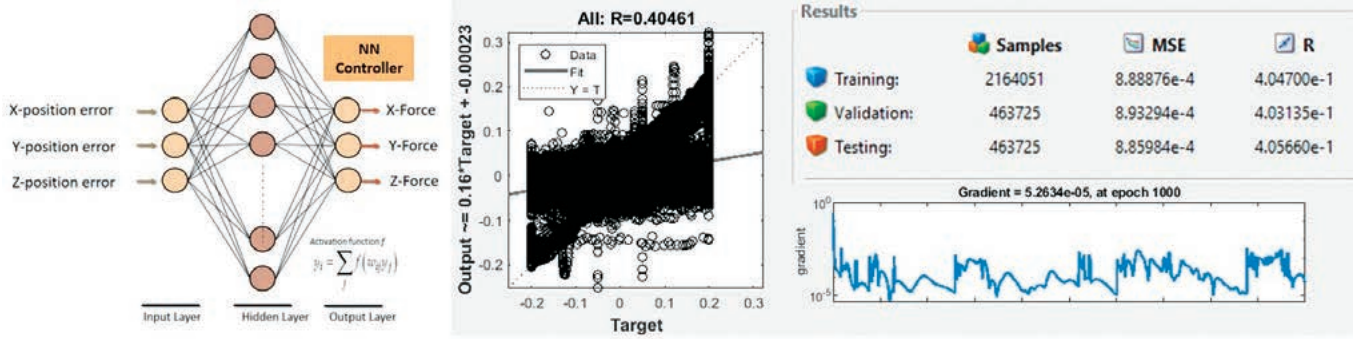


Figure 18. NN controller

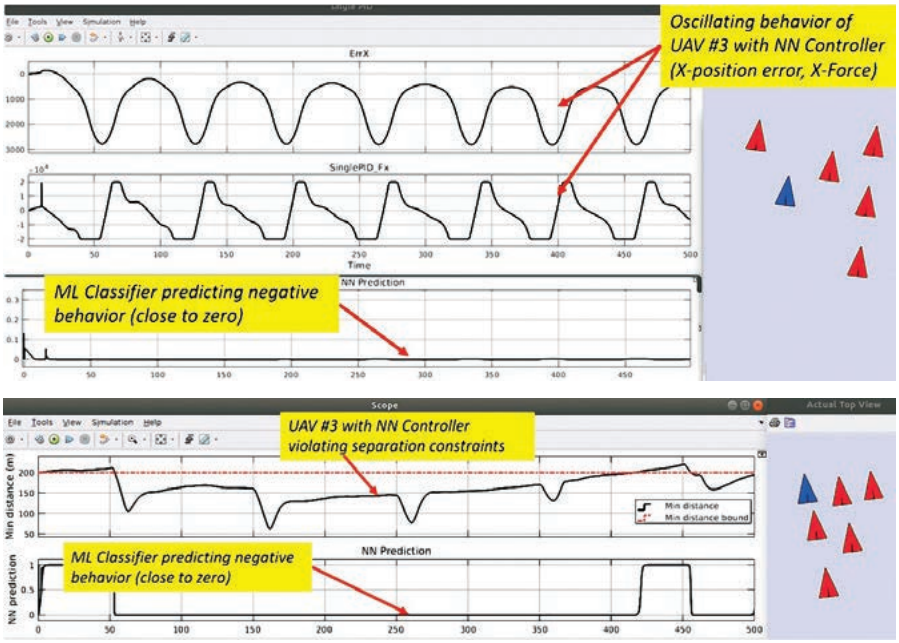


Figure 19. SoS behavior with machine learning based UAV #3

shift location of the UAVs. The individual controller behavior was approximated to a proportional control. The sample time was increased, and the direction cosine matrix eliminated by considering the maneuver in a plane of operation. These assumptions were made based on our earlier experience of using formal methods.

The Simulink design verifier took about eight hours to indicate that there could be a collision if there was a gust. We had to look at a small zone around the gust region for the model checking. Bringing in a variable in the gust time could not provide a result as it hit the limits of the computing resources used. We explored a C bounded model checker CBMC on the C language implementation of the neural network. The 6 UAV engagement scenario was coded in C and the minimum distance computed between the various UAV as a measure of asserting the behavior that collision could not happen. CBMC is a bounded model checker, and we can look at the zone where the maneuver occurs for proving the correctness. CBMC unwinds the "while loop" for the time of execution and "for loops" for the 6 UAV. The ML classifier in loop in CBMC hit the limits of the computing resources used. Other features of CBMC like array bound checks and divide by zero checks worked well with the neural network code. Subsequently, C program simulation of UAVs was tried – the simulation of the 6 UAV in the C code is shown in Figure 20. The 6 UAVs move in a 2-dimensional space and carry of a maneuver to invert the V shape. A wind gust disturbance is applied that shifts the position of all UAVs during the maneuver. The UAV realign to the new shape as seen in the plot. The simulation works well as seen. We now need to prove that it is always the case. We define the problem statement for the formal correctness as ***Definition 1*** (see below).

CBMC can prove that the condition is satisfied but to really ensure that the behavior is correct we have changed the threshold for minimum distance to a larger value to ensure that it fails and provides a counter example. CBMC can provide us this solution with a failure. It takes about 700 seconds to solve the problem. We provide the complexity of the problem as the number of variable and clauses the CBMC generates from the UAV SoS C code. The number of variables and clauses define the complexity of the problem. CBMC converts the C code into a conjunctive normal form (CNF). The CNF is a conjunction of one or more clauses, where each clause is a disjunction of variables. The C code gets translated to approximately 2,350,198 variables which are combined into around 9,416,265 clauses. The approximation is because these number change with the change in the range looked for the time of disturbance t_d. In view of the complexity of the neural network problem and the large time taken by the formal methods tool, we have explored another way of looking at the neural network performance in terms of the PCA explained above. The PCA defines the plot of the principal components of the slope or angle between each UAV and the distance measures the range between the UAVs. It is possible to define a region in the plot of PCA that defines a safe behavior

Definition 1: The distance between UAV_i and UAV_j at time t is given by $d_t(i,j) = \sqrt[2]{(x_i - x_j)^2 + (y_i - y_j)^2}$ where $(x_i - x_j)$ and $(y_i - y_j)$ represent the coordinates of UAV_i and UAV_j respectively.

Given start time maneuver t_m, end time for maneuver t_c and a disturbance at time t_d where $t_m \le t_d \le t_c$ then property $\phi = \min_{1 \le i,\, j \le 6 \wedge i \ne j} d_t(i,j) \ge \Delta$, $\forall t: t_m \le t \le t_c$ where Δ is maximum unsafe distance, holds. We look at this property in CBMC using the assert statement and assume the time of disturbance as a variable. We look at 20 seconds after maneuver as t_c

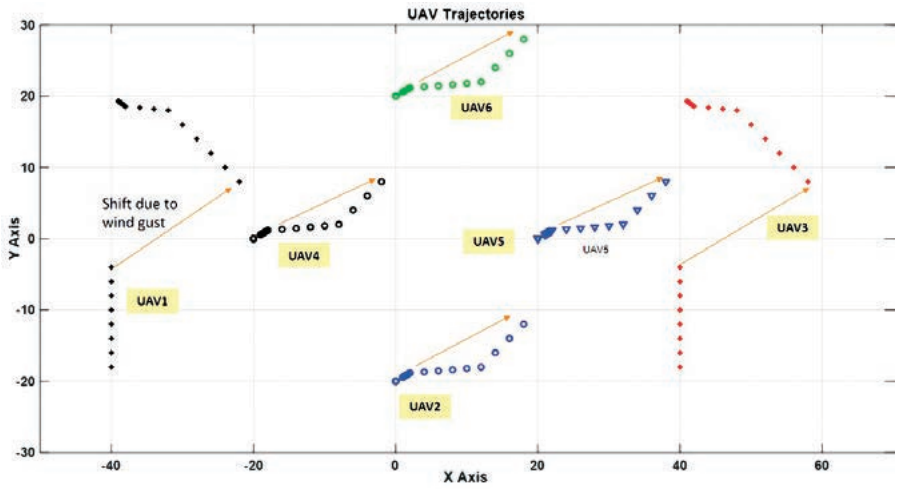


Figure 20. SoS – trajectories simulation in C programming language

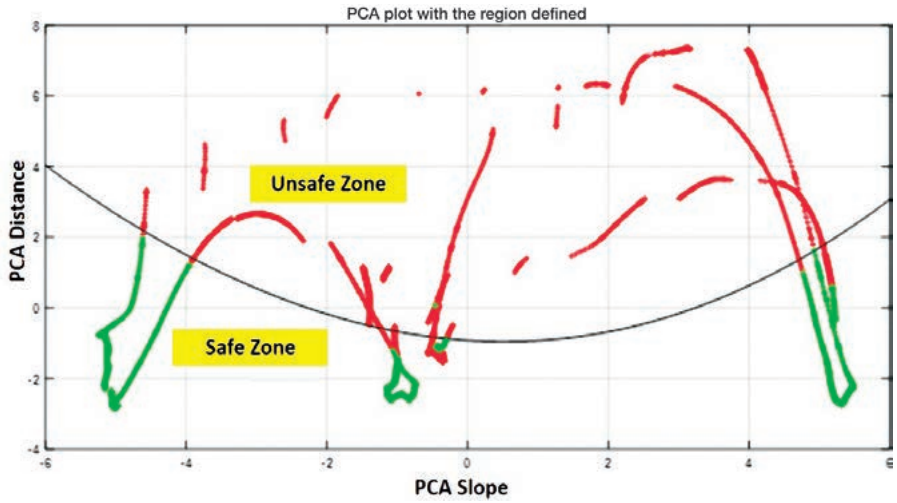


Figure 21. PCA and the region defined by the polynomial

for the swarm of 6 UAV. The region can be defined in this case a polynomial in X (the PCA slope) providing a Y (PCA distance) as shown in Figure 21. If the PCA values are in the safe zone and the neural network indicates so then we can say that neural network is correctly predicting the MOE of collision or unsafe swarm behavior. In the initial study on this given a small zone where there was a collision the formal method could provide an indication that the neural network was indeed providing an error. As the space increases, we again hit the tool and memory limitations. The results are preliminary but this, we feel, is a good direction to explore in future.

CONCLUSIONS & FUTURE WORK

This paper presented a novel approach towards application of machine learning based classifiers and formal methods for analyzing and evaluating emergent behavior of complex system-of-systems. The various elements of the framework were illustrated through a case of a swarm of autonomous UAVs flying in a formation, and dynamically changing the shape of the formation, to support varying mission scenarios. PCA based zone classification for analyzing constituent system versus SoS behaviors, machine learning classifier models for predicting positive and negative emergent behaviors, and formal verification models were presented, including their effectiveness and performance. In the future, we plan to enhance the framework to address multiple scenarios pertaining to evolution of the SoS. Over a period, different changes can happen in constituent systems, such as new functions getting added, obsolete functions getting removed, efficient means of realizing some of existing functions being incorporated and other structural changes. These changes could cause changes in the emergent behavior of the SoS. ■

REFERENCES

- Aiguier, M., P. L. Gall, and M. Mabrouki. 2008. "A Formal Definition of Complex Software." Proceedings of the 3rd International Conference on Software Engineering Advances, ICSEA, MT.
- Baomar, H., and P. J. Bentley. 2017. "Autonomous Landing and Go-Around of Airliners Under Severe Weather Conditions Using Artificial Neural Networks." Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS) Linköping, SE.
- CBMC. 2020 Bounded Model Checking for Software. https://www.cprover.org/cbmc/.
- Dong, X., Y. Li, C. Lu, G. Hu, Q. Li and Z. Ren. 2019. "Time-Varying Formation Tracking for UAV Swarm Systems With Switching Directed Topologies." *IEEE Transactions on Neural Networks and Learning Systems* 30 (12): 3674-3685.
- Giammarco, K. 2017. "Practical Modeling Concepts for Engineering Emergence in Systems of Systems." IEEE System of Systems Engineering Conference (SoSE), Waikoloa, US-HI, 18-21 June.
- Hassanien, A.E. and E. Emary. 2016, *Swarm Intelligence – Principles, Advances, Applications*. CRC Press.
- INCOSE. 2005. Technical Measurement Guide. INCOSE TP-2003-020-01.
- ———, 2015. *Systems Engineering Handbook*. 4th Edition. Hoboken, US-NJ: Wiley.
- ———, 2016. Systems of Systems, *INSIGHT 19* (3).
- Jamshidi, M. 2008. *Systems of Systems Engineering: Principles and Applications*. CRC Press.
- Kang, E., and L. Huang. 2018. "Formal Specification and Analysis of Autonomous Systems in PrCCSL/Simulink Design Verifier." Cornell University Library arXiv:1806.07702. https://arxiv.org/abs/1806.07702/.
- Katz, G., C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. 2017. "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks." Cornell University Library arXiv:1702.01135.
- Kinsner, W. 2008. "Complexity and its Measures in Cognitive and Other Complex Systems." 7th IEEE International Conference on Cognitive Informatics, Stanford University, US-CA, 14-16 August.
- Ladyman, J., J. Lambert, and K. Wiesner. 2013. "What is a Complex System?" *European Journal for Philosophy of Science* 3: 33-67.
- Lane, J. A. 2013. "What is a System of Systems and Why Should I Care?" USC-CSSE-2013-001.
- Martınez, A. M., and A. C. Kak. 2001. "PCA versus LDA." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2: 228–233.
- Mitchell, T. 1997. *Machine Learning*. McGraw Hill.
- Mohri, M., A. Rostamizadeh, and A. Talwalkar. 2012. *Foundations of Machine Learning*. MIT Press, Cambridge US-MA.
- Nanda M., J. Jayanthi, and Y. Jeppu. 2018. "Formal Methods—A Need for Practical Applications." eds. M. Nanda and Y. Jeppu. *Formal Methods for Safety and Security*. Springer.
- Nielsen, C.B. et al. 2015. "SoS Engineering: Basic Concepts, Model-Based Techniques and Research Directions." ACM Computing Surveys (18).
- PVS. 2018. Specification and Verification System. http://pvs.csl.sri.com/
- Raman, R., and M. D'Souza. 2017. "Knowledge Based Decision Model for Architecting and Evolving Complex Systems-of-Systems." INCOSE International Symposium 27: 30-44.

- Raman, R., and M. D'Souza. 2018. "Learning Framework for Maturing Architecture Design Decisions For Evolving Complex SoS." 13th IEEE Conference on System of Systems Engineering (SoSE), Paris, FR, 19-22 June, pp. 350-357.
- Raman, R., and M. D'Souza. 2019. "Decision Learning Framework for Architecture Design Decisions of Complex Systems and Systems-of-Systems." *Systems Engineering* 22: 538– 560.
- Raman, R., and Y. Jeppu. 2019. "An Approach for Formal Verification of Machine Learning based Complex Systems." INCOSE International Symposium 29: 544-559.
- Raman, R., and Y. Jeppu. 2020. "Formal Validation of Emergent Behavior in a Machine Learning Based Collision Avoidance System." IEEE International Systems Conference (SysCon), CA, 24-27 June.
- RTCA. 2011. DO-333 Formal Methods Supplement to DO-178C and DO-278A.
- SLDV. 2020. Simulink Design Verifier. http://www.mathworks.com/products/sldesignverifier.html.
- Smith, N., and T. Clark. 2006. "A Framework to Model and Measure System Effectiveness." 11th ICCRTS Coalition Command and Control in The Network Area Conference.
- Tahir, A., J. Böling, M. Haghbayan, H. T. Toivonen, and J. Plosila. 2019. "Swarms of Unmanned Aerial Vehicles — A Survey." *Journal of Industrial Information Integration* 16, https://doi.org/10.1016/j.jii.2019.100106.
- Williams-Hayes, P. 2005. "Flight Test Implementation of a Second Generation Intelligent Flight Control System." Infotech@Aerospace Conference, Arlington, US-VA, 26-29 September http://doi.org/10.2514/6.2005-6995.
- Xiang, W., P. Musau, A. W. Ayana, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson. 2018. "Verification for Machine Learning, Autonomy, and Neural Networks Survey." Cornell University Library arXiv:1810.01989 http://arxiv.org/abs/1810.01989.

ABOUT THE AUTHORS

Dr. Ramakrishnan Raman received B.Tech and MS degrees from IIT Madras, and PhD from IIIT-Bangalore. He is a certified Six Sigma Black Belt and is an INCOSE certified Expert Systems Engineering Professional – ESEP. He has extensive systems and software engineering experience in domains of building/industrial automation, and aerospace. He has been the lead systems engineer and architect for the design of many complex systems globally over the years. He is currently a principal systems engineer at Honeywell Technology Solutions, Bangalore. (ORCID:0000-0002-8471-9172)

Dr. Yogananda Jeppu is a BE in electronics and communication, from Mangalore University, a post graduate in missile guidance and controls from Pune University. He has a PhD in certification of safety critical control systems using model based techniques. He has been working in the field of control system design and implementation, simulation of aerospace systems, and verification and validation for aircrafts and missiles for the past 32 years. He started his career in 1987, working on missiles and the Indian Light Combat Aircraft program with Defense R&D Organization. He is currently working in Honeywell Technology Solutions as a principal systems engineer. (ORCID: 0000-0003-1401-6348)

Nikhil Gupta completed his BTech and MTech in aerospace engineering from IIT Madras in 2018 with a specialization in guidance, navigation, and control of unmanned aerial vehicles. He has participated in several aeromodelling competitions and loves piloting RC aircraft and multirotors. He holds a patent to his name and is certified as a SolidWorks Associate and in Six Sigma Green Belt for software. Currently, he is working in Honeywell Technology Solutions as a senior embedded engineer in the field of alternate guidance and navigation methods primarily using computer vision and neural networks.

FuSE

Future of Systems Engineering

OUR MISSION

FuSE refines and evolves the SE Vision 2035 across competencies, research, tools & environment, practices, and applications

FuSE identifies critical gaps towards the vision realization and initiates & supports relevant actions

FuSE fosters involvement and collaboration within and outside of INCOSE

FuSE educates, shares success, and expands

JOIN US

INCOSE.ORG/FUSE

INCOSE

International Council on Systems Engineering

A better world through a systems approach

33rd Annual INCOSE international symposium

hybrid event

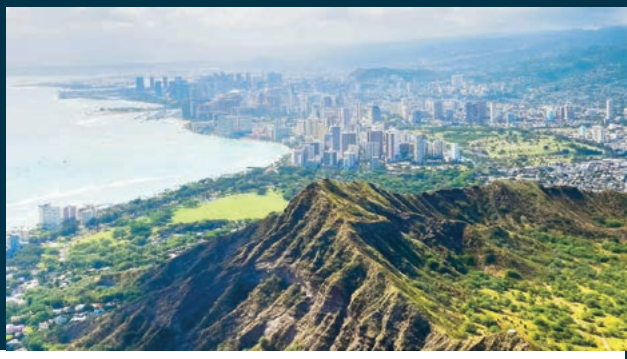
Honolulu HI USA

The Venue

Hawaii Convention Center
1801 Kalākaua Ave
Honolulu, HI 96815 - USA

Dates

Saturday 15 July, 2023
Thursday 20 July, 2023



- From now: Make your hotel reservation
- December 2022: Sponsorship registration open
- March 2023: Registration fees available
- April 2023: Registration open Final program on-line
- June 2023: Virtual platform open
- July 2023: Event (15 - 20 July 2023)

Sponsorship Opportunities

Why become a sponsor?

VISIBILITY

Unique brand of recognition and visibility for your organization

PRACTICE

Access to the latest thinking relevant to the practice of Systems Engineering

SPOTLIGHT

Put a spotlight on your organization's competency in Systems Engineering

ASSOCIATION

Be associated with the highest culture of professionalism and innovation

SUPPORT

Demonstrate organizational support to INCOSE's mission

CONNECTIONS

Put a spotlight on your organization's competency in Systems Engineering

INCOSE's Impact

18000+ Members
65+ Chapters
77+ Countries
120+ Corporate Advisory Board Members

Social media

1,950 members | 3,750 members | 470 members | 21,500 members | 3,220 members

https://www.incose.org/symp2023