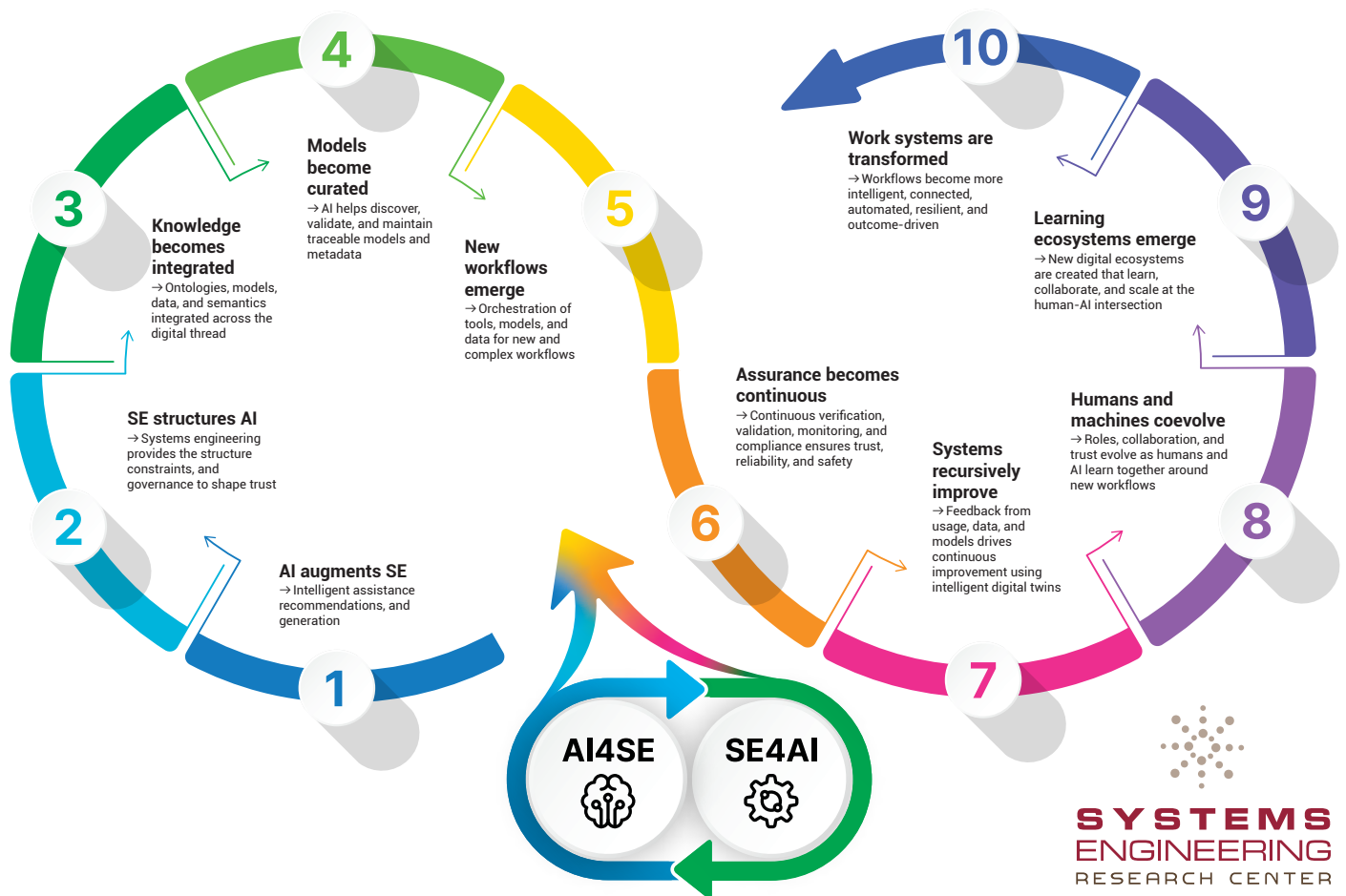


INSIGHT

This Issue's Feature: A Decade of AI and Systems Engineering: Looking Back and Forward



JUNE 2026
VOLUME 29 / ISSUE 3

A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING





EMEASEC 2026

Europe, Middle East, Africa

29 Aug - 2 Sep 2026

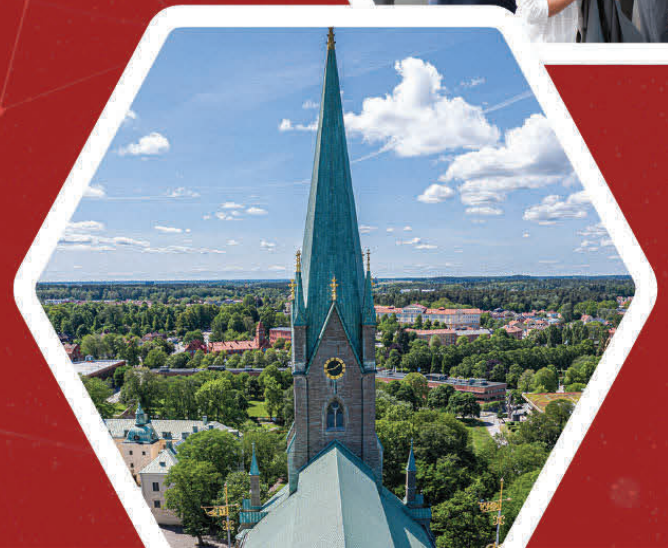
Linköping, Sweden

SAVE THE DATE FOR EMEASEC2026

Join us at EMEASEC2026, the Systems Engineering Conference for the Europe, Middle East, and Africa (EMEA) Sector of INCOSE.

 **29 AUG – 2 SEP 2026**

 **LINKÖPING, SWEDEN**



Inside this issue

FROM THE EDITOR-IN-CHIEF	7
SPECIAL FEATURE	8
Decadal Section	
A Decade of AI and Systems Engineering: Looking Back and Forward	8
Challenges and Opportunities for Systems Engineering in the Age of AI: SERC Perspective	10
A Review of the SERC AI4SE & SE4AI Workshop Series, and a Lookahead	15
From Rules to Agentic Swarms: A Systems Engineering Journey Through the Evolution of AI	19
AI4SE Section	
Twenty Tips for Using Generative AI in Systems Engineering	29
Automated Curation and Execution of Engineering Models	32
Trustworthy LLM-Augmented Data Reconciliation for Deployment-Constrained Environments	37
On (Human) Learning in Design Space Exploration and the Impact of AI Assistants	43
STPD – Operational Design as a Model-Based Extension of STPA	48
SE4AI Section	
Trustworthy AI for Quantitative Systems Engineering via Deterministic Delegation to Verified Tools	58
Engineering Trust in AI-Enabled Systems: Lessons from the Trusted AI Challenge	67
Designing Human-AI Architectural Robustness to Inherently Brittle AI	75
A Systems Architecture for Data-Constrained Environments: Leveraging the Astronaut Digital Twin to Improve Spaceflight Adaptation in Humans	83
From Models to Systems: The Evolution of T&E for Artificial Intelligence from 2020 to 2030	90
VIEWPOINT	
Who Gets to Decide? An Organizational Decision-Making Lens for AI Governance	95

About This Publication

INFORMATION ABOUT INCOSE

INCOSE's membership extends to over 27,000 members and CAB associates 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 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](http://www.incose.org)
(www.incose.org)

INSIGHT is the magazine of the International Council on Systems Engineering. It is published six 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, digital engineering, artificial intelligence, 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
Tia Gracey tia.gracey@incose.net

Theme Editors

Tom McDermott tmcdermo@stevens.edu
Zoe Szajnfarder zszajnf@gwu.edu

Layout and Design

Chuck Eng chuck.eng@comcast.net

Member Services

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

Officers

President: Michael D. Watson, *Leidos*
President-Elect: Stephen Cook, *Shoal*

Directors

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: Quoc Do, *ESEP, Frazer-Nash Consultancy*
Technical Operations Director: Tami Katz, *Ball Aerospace Services Director: Chris Browne, *CSEP, The Australian National University**

Secretary: Stueti Gupta, *BlueKei Solutions*
Treasurer: Alice Squires, *ESEP, University of Arkansas*

Director at Large: Jeff Anderson, *Boeing*
Director at Large: Annabel Fraga, *Universidad Carlos III de Madrid*
Director at Large: Annika Meijer-Henriksson, *Saab Aeronautics*
Director at Large: Robert Wirthlin, *Ford Motor Company*
Executive Director*: Steve Records, *INCOSE*

*Non voting

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://onlineibrary.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 customer@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 six times per year. Issue and article submission deadlines are as follows:

- August 2026 issue – 1 May 2026
- October 2026 issue – 1 July 2026
- December 2026 issue – 1 September 2026

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

© 2026 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 27,000 members and CAB associates and uncounted employees and students of more than 130 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
2026, Vol 29, 6 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

June Volume 29-3

EMEASEC2026	inside front cover
Systems Engineering Call for Papers	page 5
SPEC Innovations Innoslate – webinar	page 97
Caltech	back inside cover
MBSE with SysML Dassault Systèmes*	back cover

CORPORATE ADVISORY BOARD – MEMBER COMPANIES

3DSE Management Consultants
Advanced Systems Engineering, LLC
Aerospace Corporation, The
Airbus
Albers Aerospace
AM General LLC
Analog Devices, Inc.
ANSYS, Inc
Arcfield
Auburn University
Australian National University
AVIAGE SYSTEMS
Aviation Industry Corporation of China, LTD
BAE Systems
Bechtel
Becton Dickinson
Belcan Engineering Group LLC
BlueHalo Labs An AV Company
BMT Canada
Boeing Company, The
Bombardier
Booz Allen Hamilton Inc.
Boston Scientific Corporation
BTS Software Solutions
California State University Dominguez Hills
Caltech California Institute of Technology
Caggemini Engineering
Carnegie Mellon Univ. Software Engineering Institute
Change Vision, Inc.
Colorado State Univ. Systems Engineering Programs
Commercial Aircraft Corporation of China, Ltd (COMAC)
Cornell University
Cranfield University
C.S. Draper Laboratory, Inc.
CT Engineering Group
Cubic Corporation
Cummins, Inc.
Dassault Systèmes
Deloitte Consulting, LLC
Denso Create Inc
DENTSU SOKEN INC
Digiflight, Inc.
Drexel University
Eaton
EMBRAER
FAMU-FSU College of Engineering
Federal Aviation Administration (U.S.)
Florida Institute of Technology
Ford Motor Company
GE Aerospace
General Dynamics
General Motors
George Mason University
Georgia Institute of Technology
Hitachi Energy
Honeywell Aerospace Technologies
Huawei Technologies Co. Ltd
Idaho National Laboratory

IQNOX, LLC
ISAE - Supaero
ISDEFE
IVECO Group
Jama Software
Japan Aerospace Exploration Agency
Jet Propulsion Laboratory
John Deere & Company
Johns Hopkins University
KBR, Inc.
KEIO University
L3Harris Technologies
Lawrence Livermore National Laboratory
Leidos
LEONARDO
Lockheed Martin Corporation
Los Alamos National Laboratory
Loyola Marymount University
Magna
ManTech International Corporation
Marquette University
Massachusetts Institute of Technology
MBDA (UK) Ltd
Medtronic
MetaTech Consulting Inc.
Missouri University of Science & Technology
MITRE Corporation, The
Mitsubishi Electric Corporation
Mitsubishi Heavy Industries, Ltd
Modern Technology Solutions Inc
National Aeronautics and Space Administration (NASA)
National Reconnaissance Office (NRO)
Naval Postgraduate School
Nissan Motor Co, Ltd
Northrop Grumman Corporation
ONE-SYS Srl
Pacific Northwest National Laboratory
PARAMETRIC TECHNOLOGY GMBH PTC
Pennsylvania State University
Petronas International Corporation Limited
Prime Solutions Group, Inc
Project Performance International (PPI)
Purdue University
RealmOne
Redwire Space
Rolls-Royce
RTX
Saab AB
SAFRAN
SAIC
Sandia National Laboratories
Saudi Railway Company
Shanghai Formal-Tech Information Technology Co., Ltd
Shell
Siemens
Sierra Nevada Corporation
Singapore Institute of Technology
Southern Methodist University

Space Dynamics Laboratory
SPEC Innovations
Stevens Institute of Technology
Strategic Technical Services LLC
Studio SE, Ltd.
Suranaree University of Technology
Swedish Defence Materiel Administration (FMV)
Systems Planning and Analysis
System Strategy, Inc (SSI)
Taiwan Space Agency
Tata Consultancy Services
Terumo BCT Inc
Thales
The Afeka Academic College of Engineering
The George Washington University
The University of Arizona
The University of Texas at Arlington
The University of Utah
Torch Technologies
TOSHIBA Corporation
Trane Technologies
Tsinghua University
UK MoD
UNCOMN
Universidade Federal De Minas Gerais
University of Alabama in Huntsville
University of Arkansas
University of California San Diego
University of Central Florida
University of Connecticut
University of Maryland
University of Maryland, Baltimore County
University of Maryland Global Campus
University of Michigan, Ann Arbor
University of New South Wales, The, Canberra
University of South Alabama
University of South-Eastern Norway (USN)
University of Southern California
University of Texas at Austin
University of Texas at El Paso (UTEP)
US Department of Defense
Vector Informatik GmbH
Veoneer US Safety Systems, LLC
Virginia Tech
Volvo Cars Corporation
Volvo Construction Equipment
Wabtec Corporation
Warfighting Acquisition University
Wayne State University
Weber State University
Wichita State University College of Engineering
Woodward Inc
Worcester Polytechnic Institute (WPI)
Woven by Toyota, Inc.
Yulista Services, Inc.
Zuken, Inc

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.

FROM THE EDITOR-IN-CHIEF

William Miller, insight@incose.net

We are pleased to publish the June 2026 issue of *INSIGHT* published in cooperation with John Wiley & Sons as a magazine for systems engineering practitioners. The *INSIGHT* mission is to provide informative articles for advancing the state of the practice of systems engineering. The intent is to accelerate the dissemination of knowledge to close the gap between the state of practice and the state of the art as captured in *Systems Engineering*, the Journal of INCOSE, also published by Wiley.

The June issue of *INSIGHT* features “A Decade of AI and Systems Engineering: Looking Back and Forward.” We thank the Systems Engineering Research Center (SERC), especially theme editors Tom McDermott, SERC chief technology officer (CTO), and Zoe Szajnarfarber, SERC Chief Scientist and Research Council Chair, as well as the other researchers who contributed articles. Tom’s introductory article provides the context for AI and systems engineering and sets the stage for what follows. The June *INSIGHT* also includes an “op ed” by Jimmie McEver, principal scientist at the Johns Hopkins University Applied Physics Laboratory (JHU-APL) and current (2025-2026) INCOSE Deputy Technical Director, on the critical topic of AI governance: “Who Gets to Decide? An Organizational Decision-Making Lens for AI Governance.”

Looking back, Tom McDermott cites the 2019 INCOSE International Workshop where the Future of Systems Engineering (FuSE) team including Tom and your editor-in-chief spun up the AI and systems engineering working group, projects, and community outreach to support realizing the forthcoming *Systems Engineering Vision 2035: Engineering Solutions for a Better*

World (SEV2035) published in 2021. The March 2020 *INSIGHT* sponsored by the SERC addressed *augmented intelligence (AI) for systems engineering (AI4SE) and systems engineering for augmented intelligence (SE4AI)* per their research roadmap. SE4AI addresses the transformation we need in methods, procedures, and tools (MPTs) to engineer systems with embedded AI to be fit for purpose and doing no (unintended) harm. AI4SE addresses challenges that must be overcome to leverage AI in the practice of systems engineering much as a lever or pulley provides mechanical advantage to perform work in Newtonian mechanics.

The SERC and the U.S. Army have together sponsored a series of annual workshops to engage the wider community on AI and systems engineering beginning in 2020. This series has matured from conceptual exploration to practical implementation, with emphasis on lifecycle operation and assurance. The annual reports out of these workshops are published open access (www.sercuarc.org).

The decadal advancements since the March 2020 *INSIGHT* point to unprecedented acceleration changes. See Langdon Morris’ 2025 INCOSE International Symposium keynote, Ottawa, CA “AI and the Future of Systems Engineering: It’s a Revolution” https://www.youtube.com/watch?v=tEFv_VR-BhM (accessed 6 June 2026). Morris posed the question: “How will AI affect systems engineering?” His response: “... AI is indeed changing their work, and in fundamental ways, and that within 18 months they had better be prepared for a much different way of working.” Morris has coauthored *The AI Future: Cascading Waves of Change, Quantum Disruptions,*

and Humanity’s New Era, Future Lab, 2025 (<https://futurelabconsulting.com/>).

The key SERC takeaway in the June 2026 *INSIGHT* is that “AI augments systems engineering” and “systems engineering structures AI” as a recursive feedback process transforming our work systems, signaling a shift from AI4SE and SE4AI as separate concepts. The rapid shift in software workflows points to the anticipated fast change in systems engineering workflows illustrated on the *INSIGHT* cover and described in McDermott’s lead article. These workflow changes significantly impact and couple all five categories in the SEV2035 Chapter 4 Realizing the Vision (pages 55-61) that prioritize the FuSE roadmap: competencies, research, tools & environment, practices, and applications! The quote from former INCOSE president John Thomas (2012–2013) holds: “It’s a great time to be a systems engineer!”

We thank the contributing authors. We hope you find *INSIGHT*, the practitioners’ magazine for systems engineers, informative and relevant. ■

Introduction to the special issue

A Decade of AI and Systems Engineering: Looking Back and Forward

Tom McDermott, tmcdermo@stevens.edu

Copyright ©2026 by Tom McDermott. Permission granted to INCOSE to publish and use.

In early 2020, INCOSE and the Systems Engineering Research Center (SERC) joined together to publish a special issue of *INSIGHT* magazine entitled “AI and Systems Engineering.” This year we decided another special issue was warranted – one that looked across the decade of the evolution of AI and systems engineering. Never has a technology changed our discipline of systems engineering more quickly. But a whole decade? It’s only 2026. If we were to try to predict what AI will look like in 2030, we would be wrong. But after 6 years of workshops, conferences, and publications, we can spot the trends and make some definitive statements on where systems engineering and AI are going. Expect significant change.

Let’s start the story a bit before the decade. We will reflect in this special issue how AI and systems engineering have been tightly connected for many years. But AI4SE, defined as the application of AI in support of systems engineering practice and outcomes, is a rapidly accelerating trend. AI and systems engineering have been connected since the 1970’s. Many of the technologies that make up today’s generation of machine learning, generative AI, and large language models (LLMs) were in use by systems engineers as far back as the 1990’s. Wide adoption of

generative AI (GenAI) did not start until just before the decade of the 2020’s. This became a priority for INCOSE in 2019—a future of systems engineering (FuSE) workshop at the INCOSE International Workshop in January 2019 (IW2019) defined some of the initial vision and interaction for systems engineering and AI, and first noted the application areas “SE4AI” and “AI4SE” as transformational opportunities for systems engineering practice. The INCOSE AI working group was initially formed in 2019 and created a set of relationships between INCOSE, the SERC, and the Association for Advancement of AI (AAAI) to survey ongoing research and explore research needs. Version 1 of the SERC AI and Autonomy research roadmap was published in the first quarter 2020 special *INSIGHT* issue on Systems Engineering and AI (McDermott, DeLaurentis, Beling, Blackburn, and Bone 2020). This special issue set the stage for what we have experienced in the last six years.

In 2020, INCOSE and the SERC collaborated with the Association for Advancement of AI (AAAI) to execute two workshops entitled “AI meets Systems Engineering.” These workshops explored both the explosion of machine learning (ML) applications happening today, and

how these will evolve to more dynamic human-machine interactions in teams. Notably, these workshops introduced the concept of a chatbot to serve as a cognitive assistant in systems engineering activities. A book, *Systems Engineering and Artificial Intelligence*, joined the *INSIGHT* special issue, as the first definitive statements on what this decade may bring.

In 2020, the SERC hosted the first of a continuing a series of “SE4AI/AI4SE” workshops with the U.S. Army that have further informed research and application at the intersection of AI and systems engineering. The evolution of these workshops allows us to trace the trends in AI and systems engineering and make some statements about the rest of this decade. This issue’s cover art provides some clues on where this evolution is going, starting with “AI augments systems engineering” and “systems engineering structures AI.” This signals a shift halfway through the decade from AI4SE and SE4AI as separate concepts to these as a recursive feedback process. The initial article written by Zoe Szajnfarber, Tom McDermott, and Valerie Sitterle, “Challenges and Opportunities for Systems Engineering in the Age of AI: SERC Perspective,” notes this recursion as the primary shift at the mid-point of the decade.

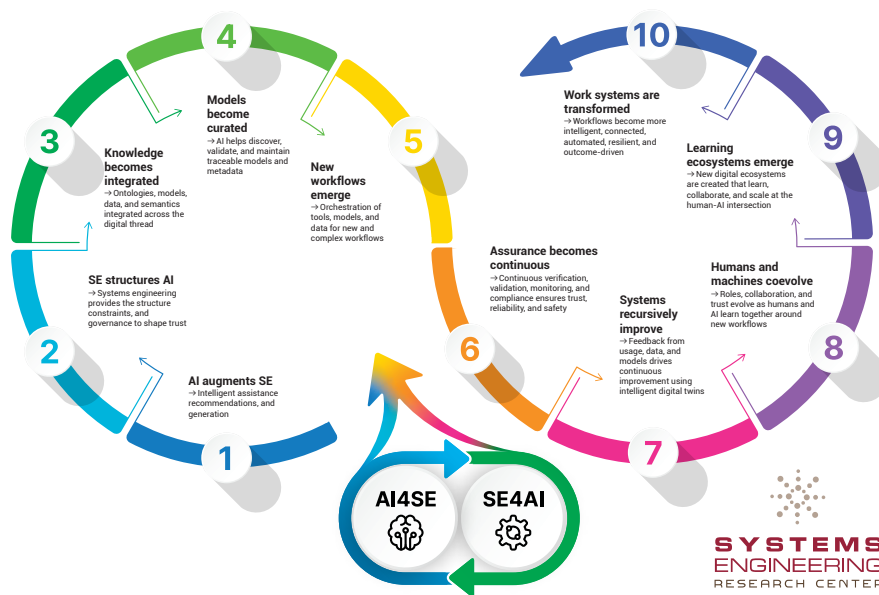


Figure 1. AI4SE and SE4AI as a recursive relationship that will transform our work systems

Take some time to study the cover art on this issue (Figure 1) which perhaps traces this recursion from today through its outcome in 2030: “transformation of the work systems we know today.” A human-AI team collaborated to extract themes from the 14 articles in this special issue to create this cover. You will find hints of the ten steps in this progression across the many articles herein. Expect that by 2030 the workflows and digital ecosystems systems we use for systems engineering will be transformed. That is only 3.5 years away, and yes, we do expect systems engineering work will change that fast (look what happened to software workflows in that short period).

We have organized this second INCOSE *INSIGHT* special issue around three sections. The first is a “decadal” section that contains three articles focused on trends. After the first article, Megan Clifford reflects on the growth of and content of the six SERC/Army AI4SE/ SE4AI workshops from 2020-2025 in “A Review of the SERC AI4SE & SE4AI Workshop Series, and a Lookahead.” The growth in interest, attendance, and presentations is one signal of the importance of this transformation, but a deeper look at the themes and content contributions as they progressed provide some solid trends. Following that, the team of Paul Wach, Alejandro Salado, and Brad Phillipbar contend that the AI4SE theme actually extends back five decades, but more importantly is accelerating rapidly and our discipline is lagging.

The “AI4SE” section focuses on the impact AI is having on systems engineering methods, processes, and tools. All of the articles in this section reflect this recursive relationship between “AI augments systems

engineering” and “systems engineering structures AI.” All should also give you time to reflect on human-AI coevolution and work system transformation – what will our workflows look like in 2030? We start with “Twenty Tips for Using Generative AI in Systems Engineering.” This article is a tribute to Dr. Barclay Brown, previous chair of the INCOSE AI Systems Working Group and tireless AI4SE advocate, who passed away earlier this year. Authors Ray Madachy, Dan O’Leary, Tom McDermott, and Sinan Bank will convince you that even working with a Generative AI requires a systems engineering process. Next, Jaya Kambhampaty, Olivia Fischer, and Dimitri Mavris present “Automated Curation and Execution of Engineering Models.” This is followed by Dan O’Leary and Allison Ledford who contribute “Trustworthy LLM-Augmented Data Reconciliation for Deployment-Constrained Environments.” We can call these the bookends of the systems engineering workflow in the age of AI—how we develop models and how we connect the underlying data. Both of these explore the concepts of human-AI workflows. Daniel Selva writes “On (Human) Learning in Design Space Exploration and the Impact of AI Assistants.” Dr. Selva raises an important question: what is lost when engineers offload cognitive effort to AI, and how do we preserve human learning? Finally, Avi Harel introduces “Systems Theoretic Process Design (STPD)—Operational Design as a Model-Based Extension of STPA.” We encourage you to reflect when reading this article how new leaning ecosystems will emerge in system design that help us deal with increasing system complexity, and how many of our work systems will be transformed using GenAI to overcome

limitations of cognitive scale.

An “SE4AI” section focuses on how systems engineering is and will need to support the evolution of “AI-enabled systems” (AIES). Ray Madachy offers “Trustworthy AI for Quantitative Systems Engineering via Deterministic Delegation to Verified Tools,” in which he presents a trustworthy AI pattern based on constraining the LLM outputs to deterministic outputs. This article discusses the critical link between the probabilistic nature of LLMs and the deterministic nature of quantitative engineering tools. In the second article, the U.S. Army and the SERC sponsored a student competition to directly explore human-AI trust relationships. “Engineering Trust in AI-Enabled Systems: Lessons from the Trusted AI Challenge” explores the results of that challenge. Aditya Singh and Zoe Szajnfarber contributed “Designing Human-AI Architectural Robustness to Inherently Brittle AI.” This article expands on concepts of trust when changes to scale and environment affect humans and AI differently and show that the ways in which humans and AI are paired together can affect the robustness of their combined system. Caleb Schmidt and Steve Simske contributed “A Systems Engineering Framework for Real-Time Biological Modeling and Monitoring in Data-Constrained Environments.” This article discusses this human-AI coevolution through the case study of an “astronaut digital twin,” formalizing expert knowledge into a testable human-machine co-learning system. In our last article “From Models to Systems: The Evolution of T&E for Artificial Intelligence from 2020 to 2030,” Tyler Cody and Peter Beling provide a follow-up look at the decadal trends and this recursion between AI4SE and SE4AI stating “we are moving from AI-enabled systems to AI-generated systems.” As generative AI effects not only the systems that we are deploying, but also how we create those systems, we significantly shift how we must evaluate future systems.

We decided to organize this special issue around the theme “A Decade of AI and Systems Engineering: Looking Back and Forward,” recognizing that our views on AI and systems engineering have significantly changed since the 2020 special issue, and that this change is only accelerating half-way through the decade. We now see the evolution of AI and the future of systems engineering as a recursive process between (as noted in the final article) “AI-enabled systems” and “AI-generated systems”. AI4SE and SE4AI are no longer separable concepts. It is the relationship between the two that will transform future systems engineering work systems. And we expect that transformation to continue to accelerate. ■

Challenges and Opportunities for Systems Engineering in the Age of AI: SERC Perspective

Zoe Szajnarber, zszajnfa@gwu.edu; Tom McDermott, tcmcdermo@stevens.edu; and Valerie Sitterle, vsitterl@stevens.edu
Copyright ©2026 by Zoe Szajnarber, Tom McDermott, and Valerie Sitterle. Permission granted to INCOSE to publish and use.

■ ABSTRACT

This perspective summarizes recent work across the SERC network in terms of five major buckets: 1) AI4SE – AI tools developed to support systems engineering work; 2) SE4AI—leveraging systems principles to assure AI-enabled systems; 3) feedback to AI4SE – where system architecting principles are leveraged to re-map what systems engineers do as AISE progresses; 4) feedback to SE4AI—where novel AI tools are developed to support challenges arise when assuring AIES; and 5) supporting the broader workforce transformation that will be needed for systems engineering to evolve with AI. In each section we highlight ongoing work and point out emerging trends. We conclude with a roadmap, suggesting key needs and trajectories for the next decade.

1.0 CONTEXTUALIZING SE + AI

Artificial intelligence enabled systems (AIES) are a key enabler of both 1) future system capabilities and 2) efficiency and effectiveness in the lifecycle of defining, engineering, acquiring, manufacturing, and operations & sustainment of future systems. While there is broad recognition that those who fail to embrace the potential of AI will fall behind, at the same time, there are serious concerns that the power of current technology is being over-hyped and that we are not prepared to ensure that when deployed, AIES will work as needed, in accordance with ethical principles and without introducing new sources of vulnerability.

While attention is often focused on developing innovative AI algorithms, most real-world AI is deployed as part of a socio-technical system, and, in that context,

value is as much a matter of how the AI is deployed within systems as it is about what the deployed AI itself can do. We contend that this creates a critical role for systems engineers to contribute, focused on the “how” of design integration and evaluation.

To that end, the Systems Engineering Research Center (SERC) has led the framing of the role of systems engineering in the AI revolution. Figure 1 defines this space. The left block AI4SE (artificial intelligence for systems engineering) focuses on how AI models and AI-based tools can be leveraged to make systems engineers more effective in their work, by enhancing analysis and decision making, and making most tasks more efficient. The right block SE4AI (systems engineering for artificial intelligence) focuses on how established systems engineering principles can improve the design, development,



Figure 1. Role of systems engineering in the AI revolution

testing and sustainment of AIES.

While these two buckets have been the primary focus of SE + AI for the first five years, more recently, as the field has

matured, more effort has been directed towards the feedback arrows. The feedback from SE4AI to AI4SE signals that once AI is introduced into systems engineering work, there is an important opportunity to rethink what work is done. That feedback loop therefore applies systems architecting principles to the task of remapping systems engineering work to be more efficient in the age of AI. This is particularly important as digital engineering transforms processes in ways that require AI integration to make it feasible. The feedback from AI4SE to SE4AI recognizes the extent to which AI can transform novel systems engineering tasks that are coming into existence because of the needs that arise in the design, development, test and sustainment of AIES.

Finally, since many of these changes are fundamentally shifting what systems engineers need to know and do, workforce development for operators and decision makers become fundamental to taking advantage of the opportunities that AI creates.

The remainder of this paper is structured around each of these five buckets, further defining them and then outlining what we see as the current state of future needs from the research and practitioner communities. We conclude summarizing some of the trends we expect to see throughout the rest of this decade.

2.0 CURRENT RESEARCH AND OPEN QUESTIONS

This section overviews ongoing research across our community. For clarity, we start with AI4SE and then explain the evolution of SE4AI feedback in that space, then shift to SE4AI and discuss the emergence of AI4SE feedback. The last section ties the threads together in terms of their impact on workforce.

2.1 AI4SE: AI Tools to Support Systems Engineering Work

Following the launch of ChatGPT, significant attention has been focused on the potential of GenAI-based tools to replace “white collar” work. Multiple tools have been developed to summarize vast corpuses of text and/or answer questions about it, cross-check facts, generate content in domain-specific styles, and transform software development. Especially as the capability and reliability of these tools continue to improve, they will play an important role in relieving many forms of mundane work.

The work that systems engineers do includes some aspects that lend themselves to the strengths of large language model (LLM) based systems, while others are challenged. Across the SERC network and broader community, we have seen several

themes of tasks where AI4SE has been particularly productive.

First, many of the analysis tasks that systems engineers have always done had already transitioned to ML and in some cases, modern AI makes this more possible. We have seen multiple studies develop AI-driven analytical tools to rapidly process larger and more unstructured data sets. Second, we have seen significant work directed towards tasks that involve generating formal models of the system (for example, in SysML) or making sense-of and translating existing models. Here the focus is often on translating natural language instructions or sketches into formal models, while making it possible to interact with the model through a chatbot-like interface. This can improve cycle times by orders of magnitudes and also enable junior employees to get up to speed much quicker. Third, we have seen many cognitive assistants, designed to help navigate complex datasets and or keep decision-maker focus on the most important pieces. Here, studies have aimed to develop interfaces that automate repetitive systems engineering tasks or workflows. Fourth, AI has been used to evaluate textual or numerical outputs of systems engineering activities such as requirements quality or verification of numerical results, ideally improving quality or just speeding up work traditionally done by human systems engineers.

We have also seen tasks where LLM-based tools seemed promising, but studies revealed the limits of current tools (or put another way, reveal the core value of what systems engineers do). These studies have tended to be more on the generative side. For example, studies where tools are developed to generate requirements or ConOps rather than validate them, or support the development of design alternatives. These studies have generally shown that while first drafts can be helpful, they often lack the context and specificity to be useful without significant rework, which requires more (not less) expertise to do well. More recent projects have attempted to provide useful knowledge bases as part of the prompt or through fine tuning, or variants of RAG (retrieval-augmented generation). While these strategies can help, in some ways, the process of doing AI4SE clarifies where engineering judgement and experience is most important.

One must not ignore the rapid evolution of the software coding disciplines. Because of the large corpuses of software code, LLMs have in a few years completely shifted the software discipline from manual coding toward AI-assisted software architecting. While the long-term effectiveness of AI-generated software is not yet known,

the markets for hiring and even training software coders has already been radically impacted. One must ask, as the quantity of public learning samples increases, if traditional systems engineering tasks like requirements generation and model-building will also rapidly change their nature. We would predict yes, and sooner than you think.

While most of the early studies have focused on replacing or augmenting tasks, as most of the projects progressed, many noticed an opportunity to rethink the overall workflow, building iterations into how the tools are used. This notion will be elaborated on in the feedback section.

Looking ahead, we see the need for research that:

- Characterizes the set of tasks where AI4SE is most helpful, and just as importantly identifies the ones where there is more risk (or less opportunity) for adoption.
- Continues to develop the tools to support systems engineers.
- Develops robust frameworks to evaluate these systems in context (SE4AI)
- Evolves the work systems in which they are deployed (via feedback).

2.2 Feedback: Leveraging Systems Engineering Principles to Evolve AI-Supported Systems Engineering Work

Today, LLM-based autonomous agents use multi-step reasoning, memory, and tool invocation to autonomously execute complex workflows. As AI deeply integrates into systems engineering, work systems must shift from human-driven, linear processes to modular, system-architected frameworks that emphasize context management, auditing, and automated verification. An emerging thrust within this community is to consider work as a system that can be architected using new tools such as Agentic AI. This can happen at the workflow level, or the work systems level.

To illustrate the difference, consider the workflow of a doctor who sees sick patients in their clinic. During each meeting, they review a chart, examine vitals and listen to the set of symptoms that prompted the visit. Based on that information they make a diagnosis, potentially ordering more tests to confirm, and then prescribe treatment. After the visit, they write up their notes for the patient’s file. Within this “workflow” there are multiple places where an AI tool could help, summarizing information, checking diagnoses, recording, transcribing visit notes. Each one of these can be done as more contained or open-ended task, for example verbatim transcription vs. summarization, with access to the full patient record or only a restricted dataset. In this context,

the workflow can be evaluated in terms of how well this patient was served.

Now, instead consider the work system of the whole clinic. At the clinic level we must consider serving many patients and likely trading resources across them. Tradeoffs occur at the level of how many regular appointments vs. slots reserved to deal with emergent issues, how to prioritize tests if too many are requested simultaneously, and whether individuals always see the same provider or if they have a care team. Again, AI could fulfill multiple functions, including scheduling and prioritizing, or potentially triaging in real time, supporting patient handoffs across a team, and much more. Some of this will change the tradeoffs significantly. For example, it may make sense for a level of triage to be done by nurses supported by AI, with doctors only coming in for difficult cases, or the pooling approach may feel more personal with better transitions, supported by AIs.

Within this thrust, several studies have examined this architecting problem. So far, much of the work is descriptive, looking at past contexts where this transition has happened. Other studies have attempted to simulate or design experiments to envision the future. This work is in the fairly early stage, but it is an area with significant potential. In the broader AI space, there is increased recognition that despite wide adoption of tools, we are still seeing limited value generation at the organizational level, because this will require a re-architecting of the work system. In terms of systems engineering and acquisition, we see advancing these ideas as a critical enabler of digital transformation, as much of the value will require core processes to be re-architected as tools are adopted. In systems engineering, because new capabilities emerge from component and system interactions, we expect that these workflows intentionally may never be fully automated. There is too much need for human input in the values that drive new systems.

As a community we appear to be working towards a few key directions:

- Novel work system architecting patterns to take advantage of the relative strengths of human and machine cognition, and their ability to work together.
- Strategies for evaluating the impact of these tools from a risk-performance tradeoff perspective.
- Support for the human transition, be it workforce, or decision support tools for these new roles.

2.3 SE4AI: Leveraging Systems Engineering Principles to Assure AIES, and Deliver Capabilities at Speed

While there has been significant hand-

wringing about the novel risks imposed by AI, many in the systems engineering community have felt that these concerns are overblown, or more precisely misdirected. After all, non-deterministic, algorithmic decision-making has been a part of aerospace and defense systems for decades. A core function of systems engineering as a discipline in that context has been to ensure that systems, when deployed, behave as intended, continue to deliver value over their lifecycles, and when they fail, do so in non-catastrophic ways.

Most of the systems that systems engineers have historically worked on are defined by long lifecycles and significant operational uncertainty. So, an important question in this thrust has been, is AI different, and if so when and how? While the community has not come up with a definitive answer, the body of work has enabled a useful framing of the types of systems.

Purpose built AI-enabled systems, that leverage AI components whose parameters are frozen before delivery, might be treatable similarly to traditional systems at the stage of acceptance testing, where they may differ during design, and post deployment. Consider the example of a drone, whose control system leveraged reinforcement learning on a large simulation training set, or its targeting system leveraged a classification model. Both types of AI can be highly performant but are known to be fragile to changes in context. This makes choices about how the systems were developed highly contingent on where and how it will be used, and requires new ideas about how systems will be monitored in the field. Several studies have looked at what would be required to observe drift it becomes an issue, or develop guidance based on safety frameworks.

In contrast, generative AI tools deployed for direct interaction by users with widely ranging experience are quite different from a systems engineering perspective. Where in the AIES context, physics tends to dampen unpredictable behavior, and the system's control surface is generally only exposed to professionalized operators, with GenAI, the interaction is both more direct, with less experienced users, and lacks the surrounding "physics" to smooth out behaviors. As a result, the challenge of designing and evaluating these systems in ways that can be assured has introduced new challenges. Several researchers within this community have developed risk management frameworks or explored guardrails and controls to manage the more serious downsides. Others have developed novel testing schemes to translate traditional concepts like combinatorial coverage to

these new spaces. However, there remain multiple open challenges about how to evolve systems engineering practice to meet these needs.

Various application areas are now developing AIES that can "learn on the fly" using pattern recognition and more traditional adaptive learning algorithms such as Kalman filters, but at the task level. In fact, "visual language models" are being evaluated in self-driving car applications, combining visual reasoning with LLMs that can update their knowledge base in real-time as new scenes are encountered. In the near future we may be closer to the "human-machine co-learning" integrative outcome envisioned by the SERC AI and Autonomy roadmap.

Looking ahead, we see the need for research that:

- Creates new tools and strategies to monitor when AIES need to be re-engineered or updated.
- Develops new architecting patterns to design AIES for this new kind of flexibility and robustness.
- Evolves systems engineering principles to adapt to novel forms of AIES including generative AI.
- Develops new test and evaluation strategies to be able to consider the complex ways AIES change when fielded and engaged with diverse operators and users.

2.4 Feedback: Leveraging AI Tools to Support the Evolution of Systems Engineering Principles for AIES

As the community has delved deeper into the challenges of design and test of AIES, new opportunities to leverage AI to support the associated evolution are being developed. In this respect we are starting to see work across three broad categories:

Tools to generate diverse content including tests and designs. One of the challenges of AIES is their performance (and attack) surfaces. Particularly as AIES engage with adversarial AIES, traditional practices for testing may be insufficient. The generative capacity of AI makes it possible to rapidly deploy much larger (and potentially more diverse) tests. This idea leverages similar underlying technology as described in the design section of SE4AI and will need to continue ensuring that the rapid generation is also grounded in appropriate context.

Test harnesses. With the new capacity to rapidly generate a large number of independent test cases, or design alternatives, there is a need to automate their consideration and application. The notion of a test harness captures the idea of a structured way to manage this process, with reusable content for future tests.

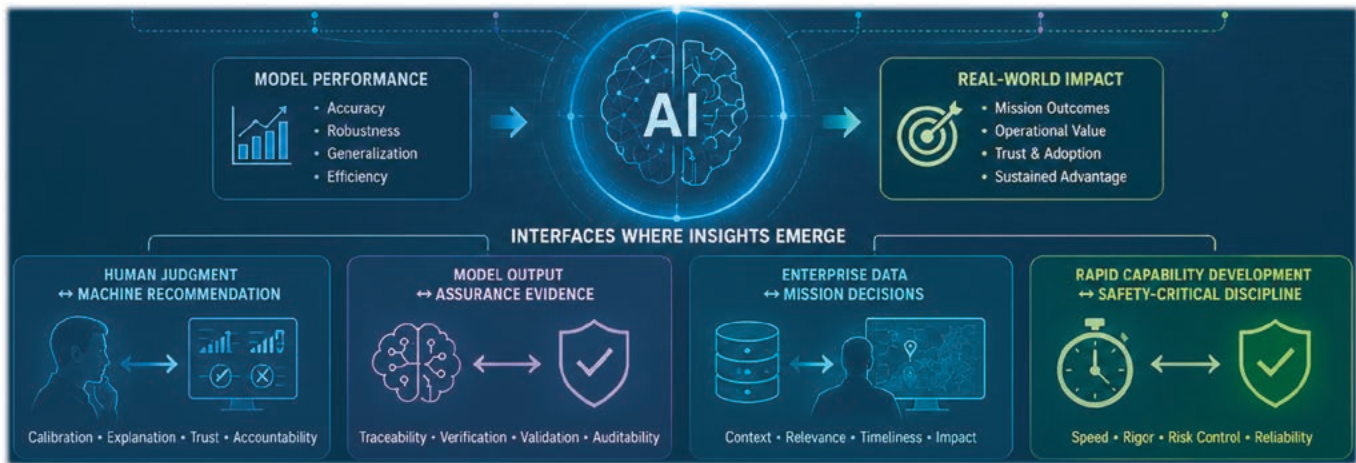


Figure 2. Interfaces where the insights emerge

Similarly, researchers are developing tools to apply architecture patterns and generated alternatives to support the concept selection phase of the design process.

Sandboxes for testing and training. One of the emerging challenges of cognitive integration of automation (not just automating functions, but also cognition and decision-making) is the need for more of the system to be represented as part of test and evaluation. There is a need to bring together notions from model validation and the types of testing that have been developed for truly novel environments (like Mars rovers). Here discovery is an important part of test, which is important when AIES, particularly those with complex operator integration, often generate surprise interaction effects. To that end, there has been a drive within the community to define what it means to expand the unit under test, to include human-human, human-AI, AI-AI interactions at relevant scale.

Consolidating these ideas, we see a need for research that:

- Develops tools that generate diverse inputs to testing and design.
- Automate their integration over multiple aspects of the system lifecycle, from design to testing and sustainment.
- Characterize what is needed to evaluate system behavior when there are multiple emergent interactions (AI-AI, H-AI, H-H etc.).
- New “-ilities” to capture behavior over the lifecycle.

2.5 Workforce Development

The trajectory described above will result in a fundamental shift in the workflows that systems engineering entails and what systems engineers need to know to do them effectively. At the same time, AIES will likely embody a much more complex

integration with their operators, meaning that system design will need to include notions of training and co-development much earlier in the process.

With respect to the systems engineering workforce, research in this community has begun to address the question of capability at three levels. First, we don't think that the skills that make system engineers effective will change significantly, in fact they will become more valuable. These are skills like critical thinking, systems thinking, systems architecting, problem solving, managing stakeholders, communicating, and decision-making under uncertainty. These skills were highlighted in the Helix study of effective system engineers. In fact, in the current evolution of AI generated software, they are exactly the skills employers are emphasizing. However, many lower-level systems engineering tasks like requirements and model-building will likely shift. Second, the SERC developed digital engineering competency framework (DECF) outlined a shift in core systems engineering skills toward information technologies, data management and analytics, and software engineering. These are necessary skills for the AI transformation. Third, researchers have started to explore what AI explainability means for systems engineers and senior decision-makers. This thread focuses on how information can/should be presented to audiences with different backgrounds, based on their needs. System engineers should be highly knowledgeable of AI as a technology. Finally, researchers have been developing dashboards that leverage AI to support different types of decision-makers learn and act in this data-intensive world, ideally assisting systems engineers (and program managers) to better manage the complexity of large systems.

With respect to operators, it's important to realize that in most practical imple-

mentations of AIES, there is still a human decision-maker at least “on” the loop, with the power to turn the system off if it isn't working as expected. Particularly when we're considering new capabilities deployed to the edge, making sure that the warfighter is prepared to use it effectively and within their context is critical to successful fielding. This requires both early training, and consideration of their capabilities, and learning curve, during the design process. Testbeds are one way to achieve that. Research has shown that many systems designed to take advantage of what humans and AI do well separately, fails when those are integrated, and we are at a state of understanding that still requires this to be investigated. More broadly, we see a need for systems engineering to expand the scope of the “system” to include the operator(s) and potentially user(s), which will require all of the aspects described so far to make feasible.

Looking ahead, we see a need for:

- All systems engineers to be knowledgeable on the capabilities and limitations of different forms of AI, at the technological level. Think of this like the way a mechanical engineer must be highly knowledgeable in material and thermal properties and underlying physics.
- Training should be developed for system engineers on SE4AI as it evolves, focused on issues such as agility, explainability, assurance, resilience, and qualification/certification.
- Human systems integration should become a core systems engineering process and skill domain, focused on human-AI handoffs particularly for information to support effective decision making.
- Tools need to be developed to support upskilling and training of future systems engineers.

- Finally, we need a rethinking of when training happens in the system development process to ensure the overall human-AI system can work well.

3.0 CONCLUSION AND WAY FORWARD

AI has the potential to change many aspects of how we work and learn, and what systems are capable of. We see the role of systems engineering in this transformation at multiple interfaces where insights emerge between both system components and system development and support processes (see Figure 2). As a community, we will be the ones to develop AI tools to support systems engineering work (AI4SE), particularly at the intersection of AI model performance and real-world performance of AIES. We must take advantage of the speed and efficiency of AI4SE but also be the SE4AI guardians of development rigor and assurance. We must advance systems engineering methods, processes, and tools to assure AIES (SE4AI) and we must develop new AI to support the new challenges presented by AI adoption (feedback to AI4SE). We also must take advantage of the opportunity to rethink what work should be done and how it should be done. This will impact systems engineering but also how it is integrated across the enterprise to support mission-related decisions. What will emerge are

Individual

- How can AI make this task or subsystem perform better?

Workflow

- How can AI integration change my work or system for the better?

System

- How can my organization's work be remapped to take advantage of relative strengths of different contributors (AI, non-cognitive technology, humans of different expertise)?

Workforce

- How does all of the above change what people need to know/do to do their jobs? (And can AI help with that).

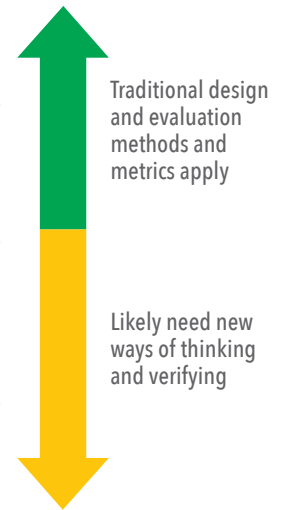


Figure 3. Challenges across four enterprise levels

new human-AI architectures and systems engineering must take a lead role in how these emerge. To navigate this transition, workforce training is critical and core, and will involve both education and new systems to support real-time evolution.

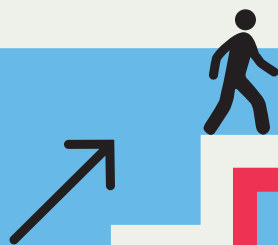
As a community we have already made significant progress along most of these directions, but there is much more work to do. The challenge spans the four enterprise

levels as shown in Figure 3: individual work, redefined workflows, transformation of the work system, and preparation of the workforce.

The SERC pledges to continue to provide academic research, convening of communities, communication, and leadership to this evolution at all enterprise levels. We hope that the community will continue to engage with us. ■

THE INCOSE CAREER COMPASS

TAKE THE NEXT STEP IN YOUR CAREER!



A Review of the SERC AI4SE & SE4AI Workshop Series, and a Lookahead

Megan Clifford, mcliffor@stevens.edu

Copyright ©2026 by Megan Clifford. Permission granted to INCOSE to publish and use.

■ ABSTRACT

This technical review synthesizes the evolution of the AI4SE & SE4AI workshop series from 2020 through 2025, with emphasis on the event's technical maturation, recurring themes, emerging patterns, and future relevance. The review considers the workshop as both an event series and a signal of disciplinary development: it reflects how the AI and systems engineering communities are defining the engineering practices required to transform AI from an emerging capability into an assured, operationally credible component of complex systems.



1. EXECUTIVE SUMMARY

The artificial intelligence for systems engineering (AI4SE) and systems engineering for AI (SE4AI) workshop series has progressed from an exploratory convening on the intersection of artificial intelligence and systems engineering into a technically mature research and application forum focused on the engineering, assurance, governance, and operationalization of AI-enabled systems. The series now reflects a decisive technical reality: artificial intelligence and systems engineering are no longer separable disciplines. AI-enabled capabilities require systems engineering to become trustworthy, testable, governable, resilient, and mission-relevant; conversely, systems engineering requires AI-enabled methods to manage increasing system complexity, accelerate engineering workflows, and sup-

port decision-making across the lifecycle.

The inaugural 2020 workshop established the foundational problem space. It recognized that digital engineering, AI/ML-enabled automation, and the emergence of AI-intensive systems would require new systems engineering methods, tools, processes, and workforce models. Initial discussions centered on defining the AI4SE/SE4AI landscape, identifying research priorities, and building a shared agenda across government, academia, and industry. By 2025, the workshop had shifted into a markedly more operational and assurance-oriented posture. The sixth

workshop was sold out, included more than 250 in-person and virtual participants, and featured two keynotes, two plenary panels, 60 submitted technical presentations, and two in-person interactive sessions. Its theme, “systems engineering AI that works: assuring transformative capabilities and enabling a digital transformation,” reflects the series’ maturation from conceptual exploration to practical implementation. Figure 1 summarizes the events and growth in interest.

Across the six-year arc, the event’s strengthening is evident in three dimensions: increased technical scope, more

WORKSHOP HISTORY

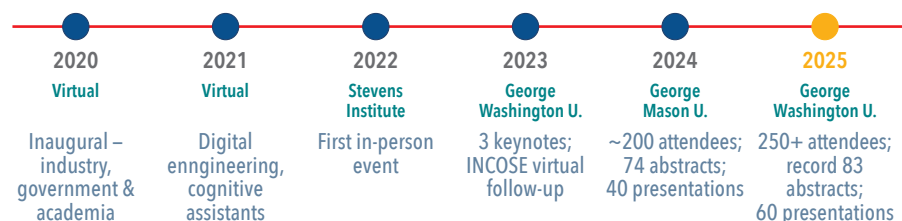


Figure 1. AI4SE and SE4AI workshop history

refined thematic structure, and a clearer focus on lifecycle assurance. The series has moved from broad questions of how AI and systems engineering relate to one another toward more rigorous treatment of trustworthiness, human-AI teaming, continuous test and evaluation, model-based validation, data governance, cognitive assistants, digital engineering infrastructure, workflow integration, resilience, and workforce transformation. The strongest recurring conclusion is that AI must not be treated as a standalone tool or isolated model. It must be engineered as a system element embedded within technical architectures, sociotechnical workflows, organizational governance, acquisition pathways, and mission outcomes.

This review article focuses on four analytical questions:

How has the workshop evolved in structure, content, and technical maturity?

1. What recurring technical themes have strengthened across the series?
2. What patterns indicate the event's increasing relevance to mission, acquisition, and engineering practice?
3. What future drivers should shape the next phase of AI4SE and SE4AI research and application?

2. EVOLUTION OF THE WORKSHOP SERIES

2.1 2020: Establishing the Technical Problem Space

The 2020 workshop established the central premise that AI and systems engineering are mutually reinforcing but also mutually disruptive. AI can augment systems engineering by automating routine engineering tasks, assisting with knowledge extraction, accelerating design analysis, and supporting decision-making. However, AI-enabled systems also expose limitations in traditional systems engineering assumptions, particularly around determinism, lifecycle stability, verification, validation, explainability, and trust.

The 2020 roadmap anticipated the major dimensions that would continue to shape the series: AI/ML technology evolution, automation and human-machine teaming, augmented engineering, digital engineering, and workforce and culture. These categories provided the conceptual architecture for subsequent workshops and remain visible in later discussions on AI assurance, digital thread development, model-based engineering, data infrastructure, and human-centered AI adoption.

2.2 2021: Transition from Broad Framing to Technical Specialization

The 2021 agenda served as a bridge between broad problem definition and specialized technical inquiry. The event em-

phasized “verification and validation (V&V) of AI systems” and “lifecycle adaptation & resilience,” signaling early recognition that AI-enabled systems require new approaches to verification, validation, reliability, adaptability, and operational monitoring.

The agenda included topics such as reinforcement learning test and evaluation, reliability and prognostics for machine learning, systems-theoretic test and evaluation (T&E) frameworks, validation arguments, U.S. Army software suitability criteria, cyber-attack impacts, NLP-based science traceability, cognitive assistance, and digital twins. In retrospect, the 2021 workshop mapped many of the themes that later became core pillars of the series: assurance, resilience, digital traceability, human cognition, model-based approaches, and lifecycle adaptation.

2.3 2022: Structuring AI4SE and SE4AI as Co-Evolving Domains

The 2022 workshop moved the series into a more deliberate structure. It included 17 presentations, two panels, and tracks organized around AI4SE, SE4AI, AI4SE and development, and SE4AI and verification and validation. This structure sharpened the distinction between using AI to improve systems engineering and using systems engineering to govern, design, and assure AI-enabled systems.

Several enduring conclusions emerged. First, AI must be treated as a system, not as an isolated algorithmic component. Second, testing must occur in system context because AI behavior is shaped by data, operating environment, user interaction, and system integration. Third, digital twins and model-based representations are critical for lifecycle analysis and experimentation. Fourth, workforce development must extend beyond data scientists and software engineers to include systems engineers, acquisition professionals, operators, and decision-makers who must understand the role of AI within larger mission ecosystems.

2.4 2023: Balancing Opportunity, Risk, and Trustworthiness

The 2023 workshop expanded the event's technical sophistication. With 24 presentations, three panels, and three keynotes, it introduced tracks on trustworthy AI, AI4SE, human/AI teaming, and SE4AI. The theme, “balancing opportunity and risk,” captured the central transition of the series: AI was no longer framed simply as an opportunity for engineering acceleration, but as a capability requiring disciplined assurance, governance, ethics, user trust, and continuous evaluation.

The 2023 discussions emphasized human-centered design, calibrated trust,

explainability, data quality, ethical frameworks, and ongoing evaluation in complex environments. This marked a significant maturation point. The series began to treat trust not as a desirable attribute but as an engineered property requiring measurable evidence, operational context, stakeholder alignment, and lifecycle verification.

2.5 2024: Strengthening SE4AI and Responsible Deployment

The 2024 workshop represented another step in the series' maturation. It gathered nearly 200 participants and included 40 presentations selected from 74 submitted abstracts, nearly doubling the previous year's presentation count. Its theme, “safer AI-enabled complex systems: responsible deployment of AI through systems engineering,” placed stronger emphasis on SE4AI and the disciplined application of systems engineering principles to AI-enabled systems.

The 2024 workshop clarified that AI-enabled systems require extensions to traditional systems engineering practice because their behavior may evolve based on data, context, operational experience, and human interaction. Discussions emphasized model-based systems engineering (MBSE) artifacts, validation infrastructure, sociotechnical testbeds, responsible deployment, and the continued need for human operators to manage uncertainty, contextual mismatch, and emergent behavior. This positioned SE4AI as a core mechanism for building trust in autonomous and AI-enabled systems, not merely as an adjacent research area.

2.6 2025: Operationalizing AI-Enabled Systems Engineering

By 2025, the workshop had matured into a research and application forum oriented toward operationalization. The event was sold out, hybrid, and attended by more than 250 participants across academia, industry, government, military, and U.S. federally funded research and development centers (FFRDCs). The agenda included two keynotes, two plenary panels, 60 submitted technical presentations, and 20 interactive sessions, including hands-on work in AI-driven SysML v2 model generation and AI-enabled mission engineering.

The 2025 technical framing is especially important. AI4SE was organized around improving systems engineering, managing complexity, cognitive assistants, AI tools and workflows, training, and digital engineering. SE4AI focused on trust and bidirectionality, lifecycle design processes, safety, reliability, ethics, digital engineering, and test and evaluation. This structure demonstrates that the field has moved

from conceptual alignment to operational integration. The workshop is no longer only asking how AI and SE relate; it is asking how AI-enabled engineering workflows can be made traceable, trustworthy, auditable, scalable, and mission-effective.

3. CROSS-SERIES TECHNICAL THEMES

3.1 Trust as an Engineered System Property

Trust emerged as one of the most persistent and technically consequential themes across the series. Early discussions connected trust to transparency, robustness, reliability, explainability, and user confidence. Over time, the treatment of trust became more operational: trust requires evidence, metrics, validation artifacts, assurance cases, testbeds, governance mechanisms, and human-in-the-loop decision structures.

The workshop series increasingly frames trust as a system-level property produced through architecture, data quality, model behavior, human interaction, operational context, and governance. This is a significant technical shift. Trustworthy AI is not achieved through model accuracy alone; it requires evidence that the system behaves reliably under mission-relevant conditions, communicates uncertainty appropriately, supports accountable human judgment, and can be monitored and adapted across the lifecycle.

3.2 Continuous Test and Evaluation

A second major theme is the movement from static verification and validation toward continuous test and evaluation. Traditional verification & validation (V&V) approaches are insufficient for AI-enabled systems that learn, adapt, behave probabilistically, or operate in dynamic environments. Across the workshop series, test & evaluation (T&E) increasingly becomes a lifecycle function that supports development, deployment, monitoring, adaptation, and retirement.

The 2025 discussions extend this theme by emphasizing proactive stress testing, optimization-based falsification, and continuous safety evaluation. This reflects a broader technical requirement: AI-enabled systems must be tested not only against predefined requirements, but against evolving operational conditions, adversarial contexts, emergent behaviors, and system-level performance interactions.

3.3 Human-AI Teaming as a Systems Design Challenge

Human-AI teaming is repeatedly framed as more than a user-interface concern. It is a sociotechnical design problem involving cognition, accountability, cognitive load, trust calibration, operator authority, task allocation, and workflow integration.

Across the series, the role of the human shifts from passive overseer to active participant in a human-machine system whose performance depends on the quality of collaboration.

The 2025 report is particularly explicit that the systems engineer's role is evolving from model constructor to orchestrator of human-AI collaboration. AI tools function most effectively as co-pilots within modular, human-in-the-loop systems where cognitive checkpoints, human oversight, and accountability mechanisms are designed into the workflow rather than appended after deployment.

3.4 Data as Engineering and Assurance Infrastructure

Data is consistently treated as foundational infrastructure for AI4SE and SE4AI. The series links data acquisition, curation, analysis, maintenance, reliability, security, and representativeness to system performance and trust. By 2025, data management is framed as a critical aspect of AI governance, and AI's value is increasingly tied to its ability to transform unstructured engineering knowledge into structured, queryable, traceable knowledge systems.

This theme has direct implications for digital engineering. AI-enabled systems engineering requires structured data environments, semantic scaffolds, ontologies, model-based artifacts, traceable evidence chains, and retrieval-augmented knowledge architectures. Without these foundations, AI may accelerate document production but fail to improve engineering confidence, decision quality, or lifecycle assurance.

3.5 Workforce, Culture, and Organizational Adoption

The workshop series consistently resists a purely technical interpretation of AI transformation. Workforce and culture are treated as engineering enablers because AI adoption requires changes in skills, roles, workflows, organizational incentives, governance structures, and decision authority. People develop, validate, deploy, interpret, and govern AI-enabled systems; therefore, the workforce must understand both the technical capabilities and the operational limitations of AI.

This theme becomes increasingly important as AI tools move from research prototypes to embedded engineering workflows. The next phase of AI4SE and SE4AI will require training systems engineers to evaluate AI outputs, design human-AI workflows, manage model risk, interpret assurance evidence, and operate within data-centric digital engineering environments.

4. EMERGING PATTERNS AND EVENT STRENGTHENING

The most important pattern across the series is the shift from possibility to disciplined implementation. Early workshops asked what AI might do for systems engineering and what systems engineering might do for AI. Later workshops ask what evidence, architecture, governance, test infrastructure, and workforce capabilities are required to make AI-enabled systems operationally credible (see Figure 2).

A second pattern is the increasing bidirectionality of AI4SE and SE4AI. AI4SE supports requirements generation, architecture analysis, traceability, knowledge extraction,



Figure 2. AI4SE and SE4AI emerging patterns

design optimization, test generation, and decision support. SE4AI provides the methods for ensuring that those tools and the AI-enabled systems they support are safe, reliable, ethical, explainable, governable, and mission-relevant. The two domains now operate as a coupled feedback loop: better systems engineering enables better AI systems, while AI-enabled methods transform the practice of systems engineering.

A third pattern is the movement from presentation to participation. The addition of interactive sessions in 2025 indicates that the community has reached a stage where shared terminology and conceptual agreement are no longer sufficient. Participants need hands-on demonstrations, model-generation exercises, sociotechnical testbeds, mission-engineering workflows, and prototype environments that translate research concepts into reusable practices.

A fourth pattern is increasing alignment with mission and acquisition relevance. The workshop series repeatedly connects AI and systems engineering to the delivery of reliable capability to the warfighter at the speed of relevance. Later workshops make this connection more actionable by addressing assurance documentation, verification artifacts, acquisition reform, data foundries, policy-tool alignment, mission engineering, digital decision superiority, and traceable governance.

5. FUTURE TECHNICAL DRIVERS FROM THE LAST TWO SERIES

The next phase of the AI4SE & SE4AI workshop series will likely be shaped by several converging drivers.

First, AI-enabled systems are becoming too adaptive, interconnected, and mission-critical to be governed through traditional engineering or software development methods alone. This creates demand for lifecycle assurance methods that combine model-based systems engineering, continuous monitoring, resilience analysis, operational testbeds, and human-machine accountability (see Figure 3).

Second, generative AI is accelerating engineering work while introducing new risks. Requirements synthesis, architecture generation, test case creation, documentation support, and knowledge extraction can all be accelerated by AI. However, these gains introduce concerns around hallucination, traceability, model contamination, evaluation ownership, intellectual control, and overreliance. Future AI4SE workflows will therefore need structured knowledge bases, domain-specific grounding, semantic scaffolds, modular prompt architectures, and embedded governance mechanisms.

Third, the workshop will remain relevant by exposing the hidden work of AI adop-

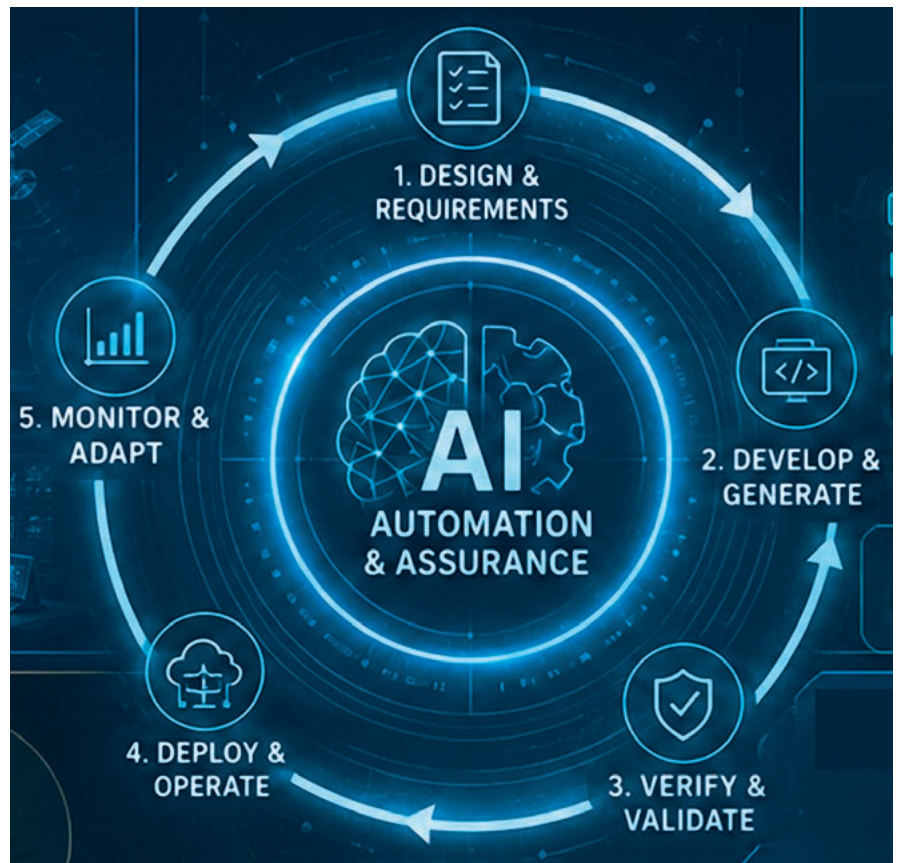


Figure 3. Technical drivers for future AI4SE & SE4AI workshops

tion. Successful AI integration depends not only on model performance, but also on data preparation, model governance, workforce retraining, infrastructure modernization, policy interpretation, cognitive integration, and organizational change. The most significant future insights may emerge at the interfaces between human judgment and machine recommendation, model output and assurance evidence, enterprise data and mission decisions, and rapid capability development and safety-critical engineering discipline.

Fourth, the series is positioned to define next-generation AI assurance practice. The 2025 discussions suggest that governance, trustworthiness, resilience, safety, and human accountability must be treated as formal systems properties. Assurance artifacts must be traceable, reusable, auditable, and embedded within engineering workflows. Adaptive agents and AI-enabled mission systems will require structured qualification processes before they can be trusted in operational environments.

6. CONCLUSION

The AI4SE & SE4AI workshop series has become more coherent, more technical, more applied, and more strategically necessary. Its early value was convening a multidisciplinary community around a

rapidly emerging problem. Its current value is helping that community define the methods, evidence structures, and engineering practices required to develop trustworthy AI-enabled systems and AI-augmented engineering workflows.

The central pattern is not merely growth in attendance, presentations, or institutional participation. The more important pattern is disciplinary maturation. Across six years, the series has moved from digital transformation to engineered trust; from AI opportunity to AI assurance; from isolated tools to integrated workflows; from human-in-the-loop rhetoric to human-AI teaming design; and from standalone research topics to a broader digital engineering ecosystem.

The workshop's future relevance will depend on its ability to continue revealing what responsible AI adoption actually requires. AI will not become operationally credible through scale, automation, or model performance alone. It will require architecture, assurance, evidence, governance, continuous test and evaluation, resilient workflows, trusted data, and a workforce prepared to engineer systems in which humans and machines reason, decide, and adapt together. The series is strongest when it treats AI not as a novelty, but as a systems engineering challenge — and as a systems engineering opportunity. ■

From Rules to Agentic Swarms: A Systems Engineering Journey Through the Evolution of AI

Paul Wach, PhD*¹, paulwach@arizona.edu; Alejandro Salado, PhD¹; and Brad Philipbar²

¹ University of Arizona | ² Philipbar Analytics LLC, IFC/USAF A

* Corresponding author: Paul Wach, PhD, Department of Systems and Industrial Engineering, University of Arizona.
Email: paulwach@arizona.edu

■ ABSTRACT

Artificial intelligence (AI) for systems engineering has evolved through six paradigms in five decades: expert systems, machine learning, deep learning, large language models, agentic AI, and agentic swarms, with each paradigm shorter than the last. This position article traces the arc through four systems the authors constructed across the paradigms, Houston (rule-based requirements), TurboArch (large-language-model architecture decisions), GenGroves (agentic model-based systems engineering), and MACQ (multi-agent acquisition support), supplemented by published work from researchers across the Systems Engineering Research Center (SERC) community for the machine-learning and deep-learning paradigms. A central observation runs through this history: each new paradigm has become a durable substrate for the next. Ontology and machine learning persist into later paradigms, and the article's portfolio of sixteen AI-for-SE systems shows how agentic swarms now compose those layers across the engineering lifecycle. The article does not attempt a comprehensive survey of AI; rather, the article identifies a critical inflection point through the lens of the authors and the SERC community. A critical observation from this article is that when organizational adoption lag exceeds paradigm duration, organizations risk investing in yesterday's frontier while missing the durable investment in the layers themselves. The systems engineering community must compress its own adoption cycles or accept permanent strategic lag.

1. INTRODUCTION

Systems engineering organizations face a widening capacity crisis. The systems they must develop, including satellites, aircraft, medical devices, and power grids, integrate more functionality, span more disciplines, and involve more stakeholders than ever before. Requirements number in the thousands, interfaces proliferate, and regulatory constraints grow more demanding. Yet engineering capacity has not grown proportionally. Talent remains scarce, budgets are constrained, and schedules compress (INCOSE 2023; INCOSE 2022). The result is an expanding gap between what must be engineered and the human capacity to engineer it.

Artificial intelligence (AI) offers a path to close this gap. Over the past five decades, AI has evolved through six distinct paradigms, each expanding what computational systems can contribute to engineering practice. This article traces that evolution through the lens of systems engineering application, from the earliest expert systems encoding human rules to the agentic AI swarms now emerging as collaborative engineering partners. Figure 1 provides a companion timeline of the six AI paradigms discussed in this article.

Two cross-cutting themes run through this history and carry urgent implications for practice:

Accelerating cycles: Each AI paradigm is shorter than the last. Expert systems and machine learning each spanned roughly twenty years. Deep learning (DL) compressed to eight. Large language models (LLMs) were available for roughly five years before agentic AI emerged. Agentic swarms are already arriving within a year of single-agent systems reaching maturity. Organizations that took five years to adopt the last wave may not have five years before the next one arrives.

Systems engineering practice adoption lag: Based on publicly available evidence, national security organizations have consistently adopted AI paradigms

Expert Systems	Machine Learning	Deep Learning	Large Language Models	Agentic AI	Agentic Swarms
1970s-1980s	1990s-2010s	2012-2020	2018-2023	2023-2024	2024-Present
Rule-based if-then logic Human-encoded knowledge Brittle, narrow, no learning	Learning patterns from data Statistical methods, early neural nets Feature engineering; one task per model	Multi-layer neural networks Auto feature learning Single-task; data-hungry	Transformers at scale Generation, not classification Hallucination; specialist trade-off	LLM + Tools + memory Goal-directed control loops Designer-specified pipelines	Multiple coordinating agents Shared memory; consensus Validation methods nascent

Figure 1. Six paradigms of AI evolution, from expert systems to agentic swarms

years after they emerge commercially. The Department of War (known as the Department of Defense (DoD) from 1949 to 2025—for consistency, this article uses the current designation throughout, including in historical context.) funded much of the original AI research in the 1960s through 1980s and adopted expert systems with relatively little lag. But as AI innovation shifted to the commercial sector, a gap opened. The transformer architecture emerged in 2017 (Vaswani et al. 2017); defense organizations did not begin serious LLM adoption until after the release of ChatGPT in late 2022, a five-year lag. As cycles compress, this lag becomes a strategic risk: organizations may find themselves investing heavily in one paradigm just as the next one arrives.

A third observation, less obvious but more consequential, threads through the history of AI: each new paradigm has become a durable substrate. Expert systems established the rules-and-ontology layer that persists in agentic systems today, encoded as guardrails, structured representations, and ontology-backed coordination. Machine learning established the statistical-learning layer that deep-learning models and LLMs still depend on. Deep learning, once it arrived, became the substrate for the language understanding that LLMs inherited. LLMs in turn became the cognitive surface that agentic systems now run on. Each paradigm is shorter than the last not because the previous one becomes obsolete, but because each new layer builds on the layers below rather than reinventing them. The frontier moves; the substrate accumulates. This compounding is what makes the adoption-lag problem strategic. Organizations that optimize only for the frontier-as-it-stands-today miss the more durable investment in the layers themselves.

This pattern also resolves a tension in how the field talks about paradigm transitions. Expert systems did not “lose” to machine learning; rule-based AI did not “lose” to learning-based AI; agentic systems are not pure generative AI. Many systems engineers reasonably prefer the simplest sufficient AI for a given task, on grounds of explainability and assurance. The accelerating-cycles claim is about which paradigm owns the frontier, not which paradigms remain in service. Prior paradigms remain in service, and most modern systems blend several.

This article grounds each era in concrete systems engineering practice through four systems the authors have constructed across the AI paradigms: Houston (Salado and Tan 2020; Wach and Salado 2019), an expert system leveraging hard-coded rules and true model-based requirements (TMBR) for requirements definition; TurboArch (Salado 2025), an LLM-based copilot for automating system architecture decisions; GenGroves (named for General Leslie Groves; Anderson et al. 2025; Wach, Nerayo, et al. 2025), an LLM-based agentic AI co-pilot bridging systems engineering modelers and domain subject matter experts (SMEs); and MACQ (modernized acquisition; authors’ work in progress), an agentic swarm co-pilot for acquisition assistance. For the machine-learning, deep-learning, and natural-language-processing (NLP) paradigms, the article points to published work by researchers across the Systems Engineering Research Center (SERC) community, including Daphne (Viros Martin and Selva 2019), digital-twin DL (Blackburn and Austin 2021; Hagedorn et al. 2020), and NLP for requirements engineering (Vierlboeck, Lipizzi, and Blackburn 2025; Topcu et al. 2025). Together, these systems and citations illustrate what each AI paradigm made

possible, where each reached its ceiling, and what each contributed as durable substrate.

2. EXPERT SYSTEMS (1970s-1980s)

Era span: roughly twenty years. **Defense adoption lag:** minimal, as DoW funded much of the original research.

The first generation of AI systems applied to engineering problems encoded human expert knowledge as explicit if-then rules. These expert systems represented domain expertise in symbolic form, enabling consistent application of established practices without requiring the human expert’s physical presence. Landmark systems included MYCIN for medical diagnosis (Shortliffe 1976) and XCON (R1) for computer configuration at Digital Equipment Corporation (McDermott 1982), among numerous other systems for spacecraft fault diagnosis and mission planning.

Expert systems offered clear strengths: they codified expertise into deterministic, auditable decision procedures. When the problem fell within the encoded rule base, performance was reliable and explainable. For systems engineering, this was a natural fit, as many systems engineering activities involve systematic application of standards, checklists, and established heuristics.

However, expert systems proved too brittle for disciplines that demand broad rulesets, like systems engineering. They performed well within their encoded knowledge but failed unpredictably when encountering situations outside their rule base. The knowledge acquisition bottleneck, the expensive and time-consuming process of eliciting and encoding expert knowledge, limited scalability (Feigenbaum 1977). As rule bases grew, the combinatorial explosion of rule interactions made maintenance increasingly difficult and introduced the risk of logical inconsistencies between rules. Expert systems could not learn from

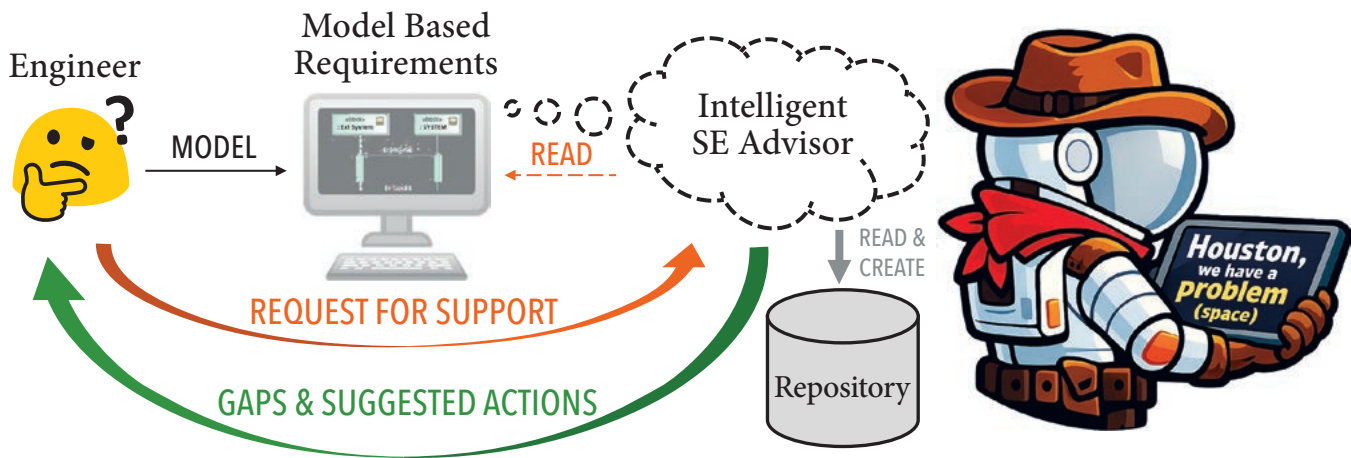


Figure 2. Houston architecture. An intelligent systems engineering advisor reads model-based requirements, identifies gaps in the requirement set, and suggests corrective actions to the engineer.

experience or adapt without manual rule updates. The limitation was scope, not the paradigm itself: expert systems remain effective for narrower, well-bounded tasks, and the rules-and-ontology layer they established persists in later paradigms.

Case Study: Houston. The expert system referred to as Houston was focused on requirements definition, leveraging hard-coded rules and true model-based requirements (TMBR) to assist systems engineering practitioners in producing consistent, standards-compliant requirements artifacts (Salado and Tan 2020; Wach and Salado 2019). Figure 2 shows the Houston architecture: an intelligent advisor reading model-based requirements, identifying gaps, and suggesting corrective actions. Houston demonstrated the value proposition of rule-based AI for systems engineering: encoding TMBR patterns enabled repeatable, template-driven requirements generation that reduced manual effort and enforced consistency. These well-documented strengths of the expert systems paradigm, including deterministic behavior, auditability, and domain-specific codification, translated directly to requirements engineering practice. Equally, the well-documented limitations of expert systems applied: the requirements engineering domain demands contextual judgment, stakeholder negotiation, and cross-domain reasoning that static rules have a difficult time capturing. As an expert system, Houston inherited these constraints. Every novel project context demanded new rules, and the system could not generalize beyond what was explicitly encoded.

What this paradigm enabled that manual practice alone could not: deterministic, auditable encoding of domain expertise into reusable decision procedures; template-driven requirements generation;

first systematic mechanization of design heuristics.

Where this paradigm reached its ceiling: contextual judgment, scalability of rule maintenance, and adaptation to novel project contexts; validation was straightforward because the rule base was inspectable, but the rule base could not grow with experience.

Houston proved the value proposition of AI-assisted systems engineering while also demonstrating why the expert systems paradigm alone, with its inherent brittleness and inability to learn, was insufficient to address the broader capacity crisis. Yet the rule-encoding pattern Houston established did not retire. It re-emerges, in hybrid form, in the agentic systems described in Sections 6 and 7, where explicit rules are encoded into generative agents to bound their behavior. It also persists in the cited works that follow: Daphne's knowledge bases and ontologies (Viros Martin and Selva 2019), Blackburn's ontology-anchored digital twins (Blackburn and Austin 2021), and the ontology-backed structured-communication protocols of GenGroves and MACQ.

3. MACHINE LEARNING (1990s–2010s)

Era span: roughly twenty years. **Defense adoption:** gradual, primarily in intelligence and signals analysis.

The second paradigm shifted from hand-coded rules to learning patterns from data. Machine learning (ML) algorithms, including support vector machines (SVMs), random forests, decision trees, and early neural networks, could generalize from training examples to new inputs, fundamentally reducing dependence on manual knowledge engineering (Mitchell 1997). In systems engineering practice, early neural networks were adopted in the early 2000s as surrogate models for

expensive simulations and as data-driven companions to physics-based models (Forrester, Sobester, and Keane 2008), an application thread that continued forward into the deep-learning-era graph autoencoders described in Section 4.

For systems engineering more broadly, machine learning enabled cost estimation from historical project data, defect prediction in software-intensive systems, and risk assessment based on project characteristics. The approach proved particularly valuable where explicit rules were difficult to articulate but historical data was available.

Case Study: Daphne (Selva and colleagues). Daphne (Bang et al. 2018; Viros Martin and Selva 2019) is an intelligent assistant for designing Earth-observing satellite systems, developed by Selva's group within the SERC community. Daphne combined a knowledge base, a rule-based reasoning engine, and classical ML over historical mission archives to support early-formulation tradespace decisions. The architecture is instructive for two reasons. First, it shows ML applied to a concrete systems engineering activity (architecture trade-space exploration), not as a generic data-mining exercise. Second, it embeds the rules-and-ontology layer from the expert-systems paradigm directly inside an ML pipeline. The expert-systems layer did not retire; it became Daphne's substrate.

However, machine learning required careful feature engineering; human experts still needed to identify and select relevant input variables. Models were typically narrow, trained for a single task, and struggled to transfer knowledge across domains. Many models proved fragile against small variations in their inputs, with minor perturbations producing disproportionately different outputs. The black-box nature of many algorithms

limited adoption in safety-critical systems engineering contexts where traceability and justification are essential. Most critically, the systems engineering domain suffered from a chronic shortage of labeled training data, constraining what could be learned.

What this paradigm enabled that the prior could not: generalization from data without manual rule encoding; learned cost and risk estimates from historical project archives; data-driven surrogates for expensive physics-based simulations.

Where this paradigm reached its ceiling: narrow per-task models, brittleness to distribution shift, the labeled-data shortage in systems engineering, and limited transparency for safety-critical use; validation required held-out test sets and, when available, agreement with physics-based reference models.

The statistical-learning techniques introduced here did not vanish when deep learning arrived; they became part of the substrate for it. Random forests still anchor cost estimation; SVMs and tree ensembles still serve as baselines and as components of larger ML pipelines.

4. DEEP LEARNING (2012–2020), WITH NATURAL LANGUAGE PROCESSING AS THE BRIDGE TO LARGE LANGUAGE MODELS

Era span: roughly eight years; cycle compression begins. Defense adoption: DoW invested (Project Maven, Joint AI Center) but systems engineering-specific adoption remained minimal.

Deep learning extended machine learning through multi-layer neural networks capable of automatically learning hierarchical feature representations from raw data (Krizhevsky, Sutskever, and Hinton 2012). The 2012 AlexNet breakthrough in image classification demonstrated that deep networks could match or exceed human performance on perception tasks, triggering rapid adoption across computer vision, speech recognition, and game-playing (Silver et al. 2016).

Architecturally, deep learning introduced convolutional neural networks (CNNs) for spatial data, recurrent neural networks (RNNs) for sequential data, and eventually attention mechanisms that would seed the next revolution. The key advance was automatic feature learning: rather than requiring human engineers to specify what to look for, networks learned relevant representations directly from data.

For systems engineering, deep learning supported image-based inspection in manufacturing and test, anomaly detection in operational systems, and ontology-anchored decision support for digital engineering.

Case Study: Blackburn and Austin’s Digital Twin Graph Autoencoders. Within the SERC community, Blackburn and Austin (2021), prepared as SERC technical report SERC-2021-TR-007 under research task WRT-1025, applies graph autoencoders, a deep-learning method, to learn low-dimensional embeddings of urban distribution-network graphs, then couples those embeddings with downstream ML for digital-twin decision support. A complementary positioning by Hagedorn, Bone, Kruse, Grosse, and Blackburn (2020), titled “Knowledge Representation with Ontologies and Semantic Web Technologies to Promote Augmented and Artificial Intelligence in Systems Engineering,” makes the persistence of the ontology layer explicit: deep learning is most useful in systems engineering when it is grounded in formal knowledge representations. Together, these works show DL applied to a concrete systems engineering activity (digital-twin decision support) on top of the rules-and-ontology layer that expert systems established.

Deep learning also re-anchored natural language processing (NLP) for systems engineering. Prior to neural NLP, requirements and specifications work relied on dependency parsing, entity extraction, and other classical statistical-NLP methods. Neural sequence-to-sequence models and contextual embeddings then extended what could be done with engineering text. Within the SERC community, Vierlboeck, Lipizzi, and Blackburn (2025) developed an NLP for requirements engineering (NLP4RE) pipeline that applies dependency parsing and entity extraction to assess the structural complexity of system requirements, an example of statistical NLP applied to a concrete systems engineering activity. As neural NLP matured into transformer-based language models, the path led directly to LLMs. The NLP layer did not retire; the LLMs of Section 5 inherited it. The bridge is most visible in work like Topcu, Husain, Ofsa, and Wach (2025), which uses transformer-based LLMs to generate systems engineering artifacts and uses NLP similarity algorithms to evaluate them, the same NLP techniques that statistical-NLP work like Vierlboeck et al. (2025) applies for structural assessment.

What this paradigm enabled that the prior could not: automatic feature learning from raw data; hierarchical representations that flexibly support perception, sequence, and graph problems; statistical-to-neural NLP that opened structured engineering text to learned models.

Where this paradigm reached its ceiling: single-task per trained model, voracious appetite for labeled data that systems engi-

neering rarely provides, classification and prediction without reasoning or generation, and (until transformers) limited ability to capture long-range structure; validation required held-out test sets and, increasingly, agreement with physics-based or ontology-grounded reference models.

The deep representations introduced here became part of the substrate for what followed. Transformer-based LLMs, modern agents, and swarm controllers depend on representations whose techniques were developed in the deep-learning era. The NLP techniques developed here continue to be used for evaluation even when LLMs do the generation.

5. LARGE LANGUAGE MODELS (2018–2023)

Era span: roughly five years. **Adoption note:** the transformer architecture emerged in 2017 (Vaswani et al. 2017), but defense and systems engineering adoption did not begin until after the November 2022 release of ChatGPT, a five-year lag.

The transformer architecture (Vaswani et al. 2017) and the scaling laws it enabled produced LLMs trained on internet-scale text corpora. Beginning with the bidirectional encoder representations from transformers model (BERT 2018) and accelerating through the generative pre-trained transformer (GPT) series and Claude, LLMs demonstrated emergent capabilities qualitatively beyond their predecessors: zero-shot and few-shot learning, general-purpose reasoning, code generation, and multi-step problem decomposition (Devlin et al. 2019; Brown et al. 2020).

For systems engineering, LLMs introduced the possibility of generating, not merely classifying, engineering artifacts. Requirements documents, design descriptions, trade study analyses, and code could be drafted from natural-language instructions. A single model could address many systems engineering tasks without task-specific training, dramatically lowering the barrier to AI-assisted engineering.

Three lines of empirical work collectively map the ceiling of LLMs operating alone. Topcu, Husain, Ofsa, and Wach (2025) showed that LLMs can produce systems engineering artifacts whose surface features pass for expert work but whose qualitative behavior resembles novice practice (premature requirements definition, unsubstantiated numerical estimates, propensity to overspecify). Wach et al. (2025a) showed that specialization on a systems engineering subdomain (SysMLv2) actually degraded general systems engineering knowledge on the SysEngBench benchmark of 1,144 multiple-choice questions, an apparently counterintuitive result: unmodified ChatGPT-4o

outperformed fine-tuned and fine-tuned-plus-retrieval-augmented-generation (RAG) variants. Wach et al. (2025b) framed this as a generalist-versus-specialist trade-off that scaling alone does not resolve. Together these works identify the LLM ceiling precisely: it is not a missing capability of any single LLM, it is the limit of what one model can do without orchestration, persistent memory, and validation feedback. This is the ceiling that motivates the move to agentic AI in Section 6.

Case Study: TurboArch. TurboArch (Salado 2025) is a proof-of-concept LLM-based copilot that automates system architecture decision-making. Named by analogy to TurboTax (making complex evidence synthesis accessible to practitioners), TurboArch retrieves and synthesizes evidence from diverse data pools, including heuristic lists and empirical studies, to predict how architectural choices impact system qualities (ilities). Where Houston required hand-coded TMBR rules for requirements, TurboArch uses LLM-based evidence synthesis for architecture; the same goal of augmenting systems engineering practitioners, achieved through a radically different mechanism that illustrates the paradigm shift from rules to learning. Later iterations of TurboArch reach toward agentic patterns; that thread is taken up in Section 6.

What this paradigm enabled that the prior could not: generation rather than only classification; general-purpose reasoning across many systems engineering tasks from a single model; rapid prototyping of evidence-synthesis copilots like TurboArch; the LLM acted as a cognitive surface that earlier paradigms could not provide.

Where this paradigm reached its ceiling: hallucination, overconfidence, the specialist-versus-generalist trade-off identified by Wach et al. (2025a), and the validation problem made acute by perceived ease-of-use; outputs that feel authoritative encourage adoption with less rigorous validation than earlier paradigms received. Validation here required human review of synthesis provenance and traceability of evidence.

The LLM layer, once established, became the cognitive surface for the agentic and swarm paradigms that followed. Modern agentic systems depend on a generative reasoning layer of this kind.

6. AGENTIC AI (2023–2024)

Era span: roughly one to two years; cycle compression accelerating rapidly. **Defense adoption:** beginning now, but most organizations are still in the LLM adoption phase.

Agentic AI addressed the fundamental limitation of standalone LLMs: a model responding to a single prompt is reactive,

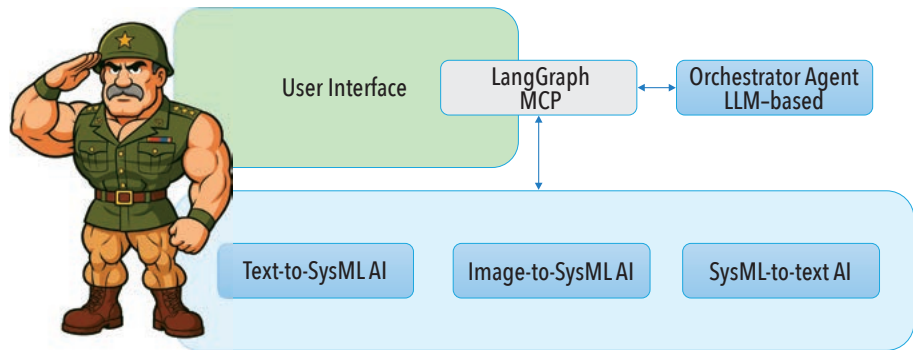


Figure 3. GenGroves architecture. An LLM-based orchestrator agent coordinates three specialized models, text-to-SysML, image-to-SysML, and SysML-to-text, through LangGraph and model context protocol pipelines.

cannot interact with external systems, and forgets between conversations. Agentic systems augment LLMs with three capabilities that change what they can do.

Tool calling. The agent invokes external functions during a task. Rather than generating only text, the agent reads files, queries databases, executes code, retrieves documents, calls validators, or invokes simulators, then incorporates the result into its reasoning. Standards such as the model context protocol (MCP) make these interactions structured and auditable. This grounds the LLM in real engineering systems.

Persistent memory. State retained across interactions. The context that would be lost between LLM prompts can be recalled and reasoned over.

Goal-directed control loops. The agent interleaves reasoning with action: think about what to do, call a tool, observe the result, decide what to do next, continue until the goal is achieved or it determines it cannot proceed. The ReAct framework (Yao et al. 2023) demonstrated that interleaving reasoning traces with tool execution substantially improved task completion on complex problems.

For systems engineering, agentic AI opened the door to automated trade study execution, design space exploration, and continuous verification workflows. An agent can be tasked with evaluating architectural alternatives, gathering relevant data, performing analyses, and synthesizing recommendations, with reduced step-by-step human guidance.

Case Study: GenGroves. GenGroves (named for General Leslie Groves) is an agentic AI co-pilot designed to bridge the communication gap between systems engineering modelers and domain SMEs (Anderson et al. 2025; Wach, Nerayo, et al. 2025; Wach, Jugan, and Lucero 2024). The system architecture comprises four specialized language models: an orchestrator

that manages workflow, a text-to-SysMLv2 generator, an image-to-SysMLv2 generator, and a SysMLv2-to-text translator. These models coordinate through LangGraph and model context protocol pipelines, with verification and validation (V&V) feedback loops ensuring generated SysMLv2 artifacts are validated before delivery. Figure 3 shows the architecture.

In practice, GenGroves operates through a sequence of orchestrated tool calls. A systems engineer asks the system to translate a domain SME’s natural-language description (for instance, an electrical engineer’s account of a power-distribution subsystem) into SysMLv2 model fragments. The orchestrator agent calls the text-to-SysMLv2 generator (a specialized fine-tuned model invoked as a tool) to draft a model fragment. If the SME provided sketches or diagrams, the orchestrator calls the image-to-SysMLv2 generator to parse those into graph fragments. Both fragments are merged. The orchestrator then calls the SysMLv2 grammar validator (a non-LLM tool) to check syntactic validity. If validation fails, the orchestrator observes the error, may re-invoke the text-to-SysMLv2 generator with the error context, and re-validates. This is the observe-and-correct loop in concrete form. Once the artifact is valid, the orchestrator calls the SysMLv2-to-text translator to render the result back into a domain-appropriate natural-language summary the SME can review and confirm. Throughout, MCP provides the structured interface for each tool call, making the orchestrator’s behavior auditable. Anderson et al. (2025) demonstrated that fine-tuned “translator” models achieved 91.37% BERTScore (an embedding-based similarity metric) accuracy in converting SysMLv2 to natural-language descriptions, establishing a bidirectional bridge that spans the full spectrum of stakeholders, from executives requiring high-level summaries to electrical engineers needing component-level specifications.

Table 1. AI paradigm durations and the compression of innovation cycles

AI Paradigm	Approx. Span	Duration	Approx. SE Adoption Lag
Expert Systems	1970s–1980s	~20 years	Minimal (DoW-funded)
Machine Learning	1990s–2010s	~20 years	Moderate (intel/signals)
Deep Learning	2012–2020	~8 years	Moderate (Project Maven, JAIC)
Large Language Models	2018–2023	~5 years	~5 years (2017 → 2022/23)
Agentic AI	2023–2024	~1–2 years	Beginning now
Agentic Swarms	2024–present	~1 year	Not yet started

What this paradigm enabled that the prior could not: multi-step, goal-directed workflows; tool use and persistent memory; orchestrated specialization across multiple models within a single pipeline; bidirectional translation between formal models and natural-language summaries.

Where this paradigm reached its ceiling: validation methods are still emerging; pipelines remain brittle to off-distribution inputs; the orchestration assumes a designer-specified workflow rather than autonomous coordination.

GenGroves represents the transition from LLMs as text generators to agentic systems that coordinate multiple specialized models and tools in pursuit of an engineering goal. It also illustrates a key architectural lesson: rather than building one monolithic systems engineering agent, decomposing the problem into specialized models with clearly defined interfaces produces more reliable results.

7. AGENTIC SWARMS (2024–PRESENT)

Era span: roughly one year and emerging. The lag problem intensifies, as organizations are still adopting LLMs while swarms are already here.

The current frontier extends agentic AI from single-agent systems to coordinated multi-agent swarms, teams of specialized agents working collectively on problems too complex for any individual agent.

It is worth distinguishing agentic swarms from the orchestrated multi-model pipelines of Section 6. **Orchestrated agents** (such as GenGroves) are pre-planned: a central planner dispatches specialized models along a fixed workflow with structured handoffs. The graph of who-talks-to-who is designed in advance. **Agentic swarms** are emergent: multiple autonomous agents with specialized roles, shared memory, and consensus protocols coordinate through peer-to-peer communication, with collective behaviors that the designer did not script in advance. Orchestration is graph-shaped and pre-planned; swarms are emergent. Orchestration is easier to validate; swarms

are more capable, but harder to assess.

Key characteristics distinguish swarms from single-agent systems:

- **Agent specialization:** individual agents optimized for specific roles (requirements analysis, architecture, domain expertise, verification).
- **Parallel execution:** multiple agents working simultaneously on subtasks, whether independent or coordinated.
- **Shared memory:** common knowledge repositories enabling information exchange.
- **Coordination mechanisms:** protocols for managing dependencies, resolving conflicts, and achieving consensus.
- **Emergent capabilities:** collective behaviors exceeding the sum of individual agent capabilities.

This architecture mirrors the multi-disciplinary, collaborative nature of systems engineering itself. Systems engineering is not a solo activity; it requires, as examples, requirements engineers, systems architects, domain specialists, integration engineers, and V&V engineers working in concert. Agentic swarms can operate the same way.

Case Study: MACQ (modernized acquisition). MACQ is an agentic swarm co-pilot designed for acquisition processes across multiple organizational frameworks, such as covering both DoW and the National Aeronautics and Space Administration (NASA). Multiple specialized agents coordinate across acquisition lifecycle phases, with current capabilities including milestone review support and systems engineering plan (SEP) generation.

MACQ's architecture assigns distinct agents to distinct acquisition concerns, including regulatory compliance, technical assessment, schedule analysis, and stakeholder coordination, and enables them to share context through shared memory and structured communication protocols. The structured protocols are themselves an ontology layer. The regulatory-compliance agent, for example, works against an explicit representation of Federal Acquisition

Regulation (FAR) and Defense Federal Acquisition Regulation Supplement (DFARS) clauses rather than free-text retrieval. When conducting a milestone review, agents independently assess their respective domains and then synthesize findings into a coordinated review package that identifies cross-cutting risks no single agent would catch in isolation.

What this paradigm enabled that the prior could not: emergent collective behavior across autonomous agents; parallelism across the full lifecycle; identification of cross-cutting concerns by composition rather than by central design; consensus-based coordination at scale.

Where this paradigm reached its ceiling: validation methods are nascent; emergent behavior is hard to bound; failure modes can be subtle and arise only at scale; auditability across many agents is an open problem.

The progression from Houston (single expert system, one domain), through the ML and DL examples cited above (Daphne, Blackburn's digital-twin DL, NLP4RE), to TurboArch (LLM copilot, architecture decisions), GenGroves (coordinated specialized models, one workflow), and MACQ (autonomous specialized agents, full acquisition lifecycle) traces the expanding frontier of AI-assisted systems engineering. *Prior layers persist:* rules and ontologies in MACQ's structured protocols; classical ML in cost and risk components; deep learning in the LLMs that power its agents; NLP in the artifact-evaluation pipelines.

8. THE COMPRESSION PROBLEM

The two cross-cutting themes of this article, accelerating cycles and defense adoption lag, converge into a strategic problem that demands attention. Table 1 summarizes how AI paradigm durations have compressed alongside defense adoption lag.

The pattern is unmistakable: paradigm durations have compressed from decades to a single year. Meanwhile, defense and national security organizations have seen their adoption lag remain roughly con-

stant or grow. The critical inflection point occurs when the adoption lag exceeds the paradigm duration. At that point, organizations risk investing heavily in a paradigm (for example, standing up LLM programs) just as the next paradigm (agentic swarms) arrives, or skipping an era entirely without ever realizing its benefits.

The persistence framing of Section 1 sharpens this argument. Each new paradigm is shorter than the last in part because it builds on accumulated layers rather than reinventing them. That same compounding makes the strategic stakes higher. Organizations that optimize only for the frontier-as-it-stands-today miss the more durable investment in the layers themselves: the ontology layer that persists from expert systems through later paradigms, the statistical-learning layer that anchors most modern models, and the deep-representation layer that LLMs and most agents depend on. Investing in those layers buys leverage across paradigm transitions; investing only in the current top layer leaves an organization exposed to obsolescence at the next transition.

This is not merely an academic observation. Organizations today are establishing LLM centers of excellence, developing LLM governance frameworks, and training their workforce on LLM tools. These are valuable investments. But if those programs take three to five years to mature, the AI landscape will have shifted beneath them twice over, and the organizations that prepared only for the LLM layer will need to re-prepare for the agentic and swarm layers as separate efforts, rather than as natural extensions of an investment already in place.

Across the six paradigms, a recurring evolutionary pattern emerges: increasing autonomy (from static rules to self-directed agents), decreasing brittleness (from rigid rule bases to adaptive learned behaviors), and growing coordination (from isolated systems to collaborative swarms). The throughline from Houston, through the cited SERC-community ML, DL, and NLP

work, to TurboArch, GenGroves, and MACQ illustrates that each AI paradigm shift expanded what was possible in systems engineering practice, while preserving the need for human engineering judgment.

What appears constant across the paradigms covered in this article:

- **Human judgment:** value-laden trade-off decisions that reflect stakeholder priorities and engineering experience.
- **Accountability:** when systems fail, humans bear responsibility; AI systems are tools, however sophisticated.
- **Stakeholder relationships:** the interpersonal dimensions of engineering that resist computational mediation.
- **Creative insight:** redefining the problem space itself, recognizing when the wrong question is being answered.

AI augments engineering capacity in the systems engineering practice we describe. It does not replace engineering judgment.

Validation, in particular, is the throughline that makes the persistence framing operationally meaningful. Each paradigm imposed its own validation regime: rule-base inspection for expert systems; held-out test sets and physics-agreement for ML and DL; provenance review for LLMs; and emerging methods for agentic and swarm systems. Because paradigms persist as substrate, validation in modern systems must operate layer-by-layer. A swarm built on agents built on LLMs built on transform-

er-era NLP built on classical ML built on ontological rules cannot be validated only at the swarm level. Validation regimes accumulate alongside the substrates they test.

9. LOOKING FORWARD

Several convergence points define the near-term frontier.

Agentic swarms and digital engineering. As organizations transition to model-based systems engineering (MBSE), digital threads, and digital twins (INCOSE 2023; INCOSE 2022; NASA 2016), they are building the infrastructure on which AI swarms will operate. MBSE provides formal, machine-readable representations of system architecture. Digital threads maintain traceability across the lifecycle. Digital twins bridge development and operations. AI swarms can consume, produce, and validate engineering artifacts across this entire digital ecosystem.

The strategic imperative. The compression problem described in Section 8 means that organizations cannot afford multi-year adoption cycles for each new AI paradigm. Instead, they must develop the organizational agility to evaluate, pilot, and scale AI capabilities on compressed timescales, or accept permanent lag as a strategic risk. This requires investment not only in technology but in workforce AI literacy, governance frameworks, organizational learning capacity, and (per the persistence framing) deliberate investment in the durable sub-



Figure 4. The authors' AI-for-systems-engineering research timeline

Table 2. AI and systems engineering research portfolio. Names are working titles subject to change as the research program matures.

Names	AI Paradigm	Activity	Key Capability
Houston	Expert Systems	Requirements definition	Template-driven, standards-compliant requirements generation
Problem Definition Workbench	Agentic AI v1, swarms expected v2	Problem definition	Streamlined requirements assessment
TurboArch	LLM (proto-agentic)	Architecture decisions	Evidence synthesis for ility trade-offs
GenGroves	Agentic AI	MBSE / SysMLv2 translation	Bidirectional bridge between modelers and SMEs
SysMLv2 Hive	Agentic Swarms	SysMLv2 grammar and modeling	GenGroves expansion: grammar development, library integration, semantic analysis
MACQ	Agentic Swarms	Acquisition lifecycle	Multi-agent milestone review, SEP generation, FAR/DFARS compliance
COSYSMOS	Agentic Swarms	Cost estimation	Parametric SE cost modeling with size drivers and effort multipliers
GI-JOE	Agentic Swarms	Knowledge representation	OWL/RDF/OML ontology engineering, BFO/CCO-aligned, SHACL validation
SCAR	Agentic Swarms	Multiple	Contested supply-chain analysis and remediation
SEAD	Agentic Swarms	Software engineering, DevSecOps	Codebase analysis, infrastructure-as-code, monitoring, security
PLMr	Agentic Swarms	Product-lifecycle management	BOM management, change control, compliance tracking
Systems Theoretic Assistant	Agentic Swarms	Cross-cutting, Systems Engineering theory	Neuro-symbolic agentic swarm; cross-domain isomorphism/homomorphism library (Wach, Sandman, and Iyer 2026; Philipbar and Wach 2026; Wach, Philipbar, and Gregory 2026)
FortWachs	Agentic Swarms	Security	Resilience, cyberphysical systems
ZynWorld	Agentic Swarms	Mission engineering, modeling and simulation	Gamification, killchain analysis, mission-driven scenario evaluation
HOS	Hive of Hives	Cross-cutting	Hive-based operating system, enabling infrastructure

Abbreviations used in Table 2: BFO = Basic Formal Ontology; BOM = Bill of Materials; CCO = Common Core Ontologies; COSYSMOS = Constructive Systems Engineering Cost Model Swarm, based on COSYSMO (Valerdi 2005); GI-JOE = Generalized Information, Judicially and Ontologically Engineered (aka, semantic hero); HOS = Hive Operating System; OML = Ontology Modeling Language; OWL = Web Ontology Language; PLMr = Product Lifecycle Manager; RDF = Resource Description Framework; SCAR = Supply Chain Analysis and Remediation; SEAD = Software Engineering, Architecture, Development, Deployment, Operations, and Security; SHACL = Shapes Constraint Language.

strate layers, not only the frontier layer of the moment. Figure 4 is an illustration of the rapid pace, based on samples from our portfolio.

To summarize the authors' perspective: from a single expert system (Houston, 2019) through agentic AI (GenGroves, 2023–2025), the portfolio expanded rapidly into a hive of hives comprising sixteen specialized agentic swarms by early 2026

(Figure 4). The next horizon is integration. **STOIC (systems theoretic information coherence)**, a math-first ontology approach building on the authors' earlier systems theoretic co-pilot work (Wach et al. 2025c), is becoming the core of the next-generation hive operating system (HOS).

A broader AI-for-systems-engineering research portfolio. The four case studies in this article (Houston, TurboArch,

GenGroves, and MACQ) and the cited SERC-community ML, DL, and NLP work are part of a larger research program investigating AI across the full systems engineering lifecycle. Table 2 summarizes the current portfolio of sixteen systems spanning requirements engineering, problem definition, architecture decisions, model-based systems engineering, acquisition management, cost estimation,

ontology engineering, supply-chain analysis, software-engineering infrastructure, product-lifecycle management, security, mission engineering, and cross-cutting hive coordination. Each tool addresses a different systems engineering activity, but most share a common design philosophy: AI augments practitioner judgment through structured workflows, not black-box automation. Together, they form a testbed for studying how AI paradigms translate into engineering practice, and where they fall short.

From swarms to hives of hives. The portfolio in Table 2 already exhibits a pattern beyond individual swarms: multiple agentic-swarm systems sharing ontologies, memory, and coordination protocols across systems engineering domains. This “hive of hives” architecture, where specialized swarms collaborate at the enterprise level through neuro-symbolic reasoning, represents the next frontier. One of the authors participates in NATO STO exploratory team SAS-ET-GI, “neuro-symbolic AI-augmented capability gap analysis and acquisition decision support,” which is investigating precisely this class of system: AI-augmented methods that combine symbolic reasoning with neural capabilities to identify, test, and justify future military capabilities under conditions of uncertainty (NATO STO 2026). In parallel, the authors’ SERC-sponsored research program “systems engineering beyond the horizon” (Wach, Topcu, and Esser 2025; SERC 2025) has developed a living transformation roadmap that positions neuro-symbolic agentic architectures as the next capability horizon for systems engineering practice. The convergence is clear: the systems engineering community is not merely adopting AI tools; it is approaching a threshold where interconnected AI ecosystems become integral to how engineering organizations think, decide, and act.

Validation in the age of rapid adoption. A critical difference between earlier AI paradigms and LLMs is the perceived ease of use. Expert systems and ML models demanded explicit validation before deployment; their limitations were visible and well-understood. LLMs, by contrast, produce outputs that feel authoritative, encouraging adoption with less rigorous validation. Early research confirms this concern: García Alarcia, Golkar, and Salado (2026) found that LLM-generated requirements, while superficially similar to human-authored ones, require careful scrutiny to identify subtle quality differences. As the systems engineering community adopts LLM-based and agentic tools at compressed timescales, research into validation methods for AI-generated engineering artifacts becomes increasingly urgent. Under the persistence framing of this article, that research must explicitly address layer-by-layer validation: how does one validate a swarm that runs on LLMs that ride on transformer-era NLP that depends on classical ML that anchors to formal ontology? Each layer needs its own validation regime, and the regimes need to compose. This is a research program in itself, and an open one.

A call to action. The systems engineering community faces a choice. The community can react to AI advancement as it unfolds, adapting after the fact to capabilities others develop for other purposes. Or the community can actively shape how AI is applied to engineering practice, ensuring that the resulting systems serve engineering needs, preserve engineering values, and amplify the capacity that the discipline so urgently requires (INCOSE 2022).

Three concrete openings define what “actively shape” looks like in the next eighteen months. **First**, INCOSE working groups can formalize the layer-by-layer

validation discipline this article argues for, producing a community standard for validating AI-augmented systems engineering artifacts that addresses each substrate layer (ontology, statistical learning, deep representations, language, agency) rather than only the top-of-stack output. **Second**, funded research can develop persistence-aware adoption decision support. Organizations transitioning between AI paradigms face a real decision-support problem: what to keep from the prior paradigm as durable substrate, what to layer onto next, and how to validate the composition. This is itself a systems engineering problem, and one the systems engineering research community is positioned to study with rigor. **Third**, the community can establish a pre-validation program for agentic systems engineering tools, modeled on early-phase clinical trial discipline, so that organizations adopting agentic and swarm systems do so on the basis of evidence rather than impression.

The durable investment is at the layers, not the frontier. The frontier moves; the layers accumulate. Organizations and communities that treat each new paradigm as a replacement of the last will keep paying the full cost of every transition. Organizations and communities that recognize the compounding will build platforms that survive the transitions.

The time between paradigm shifts is no longer measured in decades. The window to shape the next paradigm is open now. ■

ACKNOWLEDGEMENT

This work was supported by SERC project WRT-2516, “Systems Engineering Beyond the Horizon.” The authors used PostWach in the creation of this article. PostWach is an agentic hive developed by Dr. Wach based on rUv (2026).

REFERENCES

- Anderson, T., G. Topcu, K. Jugan, S. Nerayo, and P. F. Wach. 2025. “LLM-Enabled Knowledge Transfer: Modeler to SME.” In Proceedings of CSER, Long Beach, CA.
- Bang, H., A. Viros Martin, A. Prat, and D. Selva. 2018. “Daphne: An Intelligent Assistant for Architecting Earth Observing Satellite Systems.” 2018 AIAA Information Systems-AIAA Infotech@ Aerospace, AIAA SciTech Forum.
- Blackburn, M. R., and M. A. Austin. 2021. Architecting for Digital Twins and Model-Centric Engineering with AI/ML, Part II. SERC Technical Report SERC-2021-TR-007 (research task WRT-1025). Systems Engineering Research Center.
- Brown, T., et al. 2020. “Language Models Are Few-Shot Learners.” *Advances in Neural Information Processing Systems* 33.
- Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. 2019. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of NAACL-HLT.
- Feigenbaum, E. A. 1977. “The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering.” In Proceedings of the International Joint Conference on Artificial Intelligence.
- Forrester, A. I. J., A. Sobester, and A. J. Keane. 2008. *Engineering Design via Surrogate Modelling: A Practical Guide*. Chichester, UK: Wiley.
- García Alarcia, R. M., A. Golkar, and A. Salado. 2026. “Exploring Similarity Assessments of LLM-generated Requirements.” In Proceedings of CSER.
- Hagedorn, T., M. Bone, B. Kruse, I. Grosse, and M. Blackburn. 2020. “Knowledge Representation with Ontologies and Semantic Web Technologies to Promote Augmented and Artificial Intelligence in Systems Engineering.” *INSIGHT* 23 (1). DOI: 10.1002/inst.12279.

- Hamilton, K., and M. I. Ali. 2026. “Neuro-symbolic AI for Predictive Maintenance (PdM): Review and Recommendations.” arXiv:2602.00731.
- INCOSE. 2022. *Systems Engineering Vision 2035*.
- INCOSE. 2023. *Systems Engineering Handbook*. 5th ed. Wiley.
- Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. “Image Net Classification with Deep Convolutional Neural Networks.” *Advances in Neural Information Processing Systems* 25.
- McDermott, J. 1982. “R1: A Rule-Based Configurer of Computer Systems.” *Artificial Intelligence* 19 (1): 39–88.
- Mitchell, T. M. 1997. *Machine Learning*. New York: McGraw-Hill.
- NASA. 2016. *Systems Engineering Handbook*. Rev. 2. NASA/SP-2016-6105.
- NATO STO. 2026. “Neuro-Symbolic AI-Augmented Capability Gap Analysis and Acquisition Decision Support (SAS-ET-GI).” *SAS Newsletter*, January 2026.
- Philipbar, B., and P. F. Wach. 2026. “DEVS-Based Agentic AI Swarms: System Entity Structure of Queens for Rapid Systems Development and Verification.” INCOSE International Symposium (accepted).
- rUv. 2026. ruflo (Version 3.6.27). Agent orchestration platform for Claude. <https://github.com/ruvnet/ruflo>.
- Salado, A. 2025. “TurboArch: Towards Automating System Architecture Decisions with a CoPilot.” INCOSE International Symposium 35 (1): 20–36. DOI: 10.1002/iis2.70074.
- Salado, A., and R. M. Tan. 2020. “Structural Rules for an Intelligent Advisor to Identify Requirements Gaps using Model-Based Requirements.” In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 920–927. DOI: 10.1109/SMC42975.2020.9283466.
- SERC. 2025. “Research Highlight: Optimized Portfolio Digital Engineering Transformation.” Systems Engineering Research Center, December 15, 2025. <https://sercuarc.org/news/research-highlight-optimized-portfolio-digital-engineering-transformation/>.
- Shortliffe, E. H. 1976. *Computer-Based Medical Consultations: MYCIN*. New York: Elsevier.
- Silver, D., et al. 2016. “Mastering the Game of Go with Deep Neural Networks and Tree Search.” *Nature* 529: 484–489.
- Topcu, T. G., M. Husain, C. Ofsa, and P. F. Wach. 2025. “Trust at Your Own Peril: A Mixed Methods Exploration of the Ability of Large Language Models to Generate Expert-Like Systems Engineering Artifacts and a Characterization of Failure Modes.” *Systems Engineering*, Wiley. DOI: 10.1002/sys.21810.
- Valerdi, R. 2005. “The Constructive Systems Engineering Cost Model (COSYSMO).” Ph.D. dissertation, University of Southern California.
- Vaswani, A., et al. 2017. “Attention Is All You Need.” *Advances in Neural Information Processing Systems* 30.
- Vierlboeck, M., C. Lipizzi, and M. R. Blackburn. 2025. “Natural Language Processing to Assess Structure and Complexity of System Requirements.” *Systems Engineering*. DOI: 10.1002/sys.21784.
- Viros Martin, A., and D. Selva. 2019. “Daphne: A Virtual Assistant for Designing Earth Observation Distributed Spacecraft Missions.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.
- Wach, P. F., G. Bell, K. Jugan, E. Longshore, and R. Madachy. 2025a. “The Cost of Expertise: Performance Trade-Offs in LLMs for Systems Engineering.” In Proceedings of the INCOSE International Symposium, Ottawa.
- Wach, P. F., A. Iyer, B. Sandman, et al. 2025c. “Systems Theoretic Co-Pilot MVP.” In Proceedings of the Conference on Systems Engineering Research (CSER), Long Beach, CA.
- Wach, P. F., K. Jugan, and J. Lucero. 2024. “Using Large Language Models to Accelerate Development of Complex Systems.” SERC AI Workshop.
- Wach, P. F., S. Nerayo, K. Jugan, T. Anderson, P. Beling, and G. Topcu. 2025b. “Cautions of Leveraging LLMs for Systems Engineering: Generalist versus Specialist.” In Proceedings of CESUN.
- Wach, P. F., S. Nerayo, P. Beling, K. Jugan, T. Anderson, and V. Anand. 2025. “GenGroves: A Bridge between Systems Engineers and Domain Experts.” SERC AI4SE Workshop.
- Wach, P. F., B. Philipbar, and J. Gregory. 2026. “Math-Based Data Structures and Analysis for Mission Engineering.” INCOSE International Symposium (accepted).
- Wach, P. F., and A. Salado. 2019. “Template/Model-Based Requirements: A Framework for Structured Requirements Definition.” In Proceedings of the INCOSE International Symposium.
- Wach, P. F., B. Sandman, and A. Iyer. 2026. “Toward a Library of Isomorphic Patterns for Systems Engineering.” In Proceedings of the Conference on Systems Engineering Research (CSER).
- Wach, P. F., T. Topcu, and K. Esser. 2025. Optimized Portfolio Digital Engineering Transformation. SERC Technical Report WRT-2406. Systems Engineering Research Center.
- Yao, S., et al. 2023. “ReAct: Synergizing Reasoning and Acting in Language Models.” In Proceedings of ICLR.

ABOUT THE AUTHORS

Paul Wach, PhD is a systems engineering and AI researcher in the Department of Systems and Industrial Engineering at the University of Arizona. He is developing a hive-of-hives platform comprising the majority of the AI systems listed in Table 2, including HOS, MACQ, GI-JOE, SEAD, COSYSMOS, PLMr, SysMLv2 Hive, and Problem Definition Workbench. He leads research on SE4AI/AI4SE and participates in the NATO STO SAS-ET-GI on neuro-symbolic AI for defense capability analysis. He is a member of INCOSE.

Alejandro Salado, PhD is an associate professor in the Department of Systems and Industrial Engineering at the University of Arizona. His research focuses on systems engineering theory, requirements engineering, and system architecture decision-making. He is the developer of TurboArch and co-developer of the TMBR framework. He is an INCOSE Fellow.

Brad M. Philipbar is principal at Philipbar Analytics LLC and the AFAF Fox Exercise and Wargame Senior Fellow at the U.S. Air Force Academy’s Institute for Future Conflict, where he conceives and directs AI-driven exercises and analytic wargames utilizing the Academy’s AI and Multi-Domain Laboratories. A former GS-15 DoW senior civilian leader, he has led digital engineering frameworks, integrated space-based sensing into joint operations, and directed cross-sector coalitions spanning government and industry, with earlier work in computational physics, high-performance computing, and particle-detector instrumentation. He holds an M.S. in mechanical engineering and a B.S. in physics, with publications in *Physical Review Letters*, *Nuclear Instruments and Methods in Physics Research Section A*, and *Numerical Heat Transfer*, and is a member of INCOSE, AIAA, ASME, NDIA, and SISO.

Twenty Tips for Using Generative AI in Systems Engineering

Ray Madachy, rjmadach@nps.edu; Dan O'Leary, djo0008@auburn.edu; Tom McDermott, tmcdermo@stevens.edu, and Sinan Bank, sinan.bank@colostate.edu

Copyright ©2026 by Ray Madachy, Dan O'Leary, Sinan Bank, and Tom McDermott. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Generative AI (GenAI) is rapidly moving from experimentation to routine use in systems engineering practice. We are already using these tools across virtually all process activities. The potential benefits are substantial, but the risks are equally real including inconsistent outputs, hidden assumptions, unverifiable calculations, weak traceability, context loss across sessions, and avoidable rework caused by poorly framed prompts and informal review habits. Current use is often ad hoc, whereas we need to move to disciplined individual and team processes that are measurable and thus manageable.

This article is dedicated to Dr. Barclay Brown, previous chair of the INCOSE AI Systems Working Group. Barclay recently passed away and was a strong advocate in our community for applying AI in systems engineering with practical engineering discipline. He could have written this article as well, and this contribution aims to carry his spirit forward by offering concrete guidance that engineers can apply immediately while preserving rigor, transparency, and accountability.

■ A SHORT TRIBUTE TO DR. BARCLAY BROWN

Barclay was very fond of Star Trek references. He would say about GenAI “It’s life, Jim, but not as we know it.” He liked to describe GenAI like it was a new kind of being: “Neither human nor sentient; no soul, no goals; but with vast knowledge and information; creativity of a sort; little (or no) judgment; great speed; ‘eager’ to please; and happy to re-do things many times.” Why would we not want to converse with an entity like this?

He would note “Large Language Models (LLMs) have no interests, desires, ambitions, understandings, willingness, emotions, or intentions; and no capacity for these, though they can TALK like they do. But treating AIs like beings is the most productive way to relate to them. They know all about human behavior, so they respond better when addressed in that same way. So have them act in any way that serves you.” This makes them extremely easy to use but also makes it easy to accept what they tell you as correct, complete, verifiable, and immediately usable. Easy to use works for short information summaries, but not for complex systems engineering tasks.

Barclay enjoyed presenting tips on using GenAI in INCOSE and other conferences. In his spirit we have distilled some essential tips for systems engineers using GenAI. Use it but use it right.

TWENTY TIPS FOR USING GENERATIVE AI IN SYSTEMS ENGINEERING

Effective use of GenAI in systems engineering depends on treating AI interaction with the same rigor applied to other processes that produce engineering artifacts. This means stating purpose clearly, defining constraints and acceptance criteria, exposing ambiguity early, requesting structured outputs,

and designing reviewable workflows. It also means delegating quantitative tasks to deterministic tools when correctness and auditability matter. At the team level, it means institutionalizing reusable context through curated assistants and retrieval mechanisms, benchmarking models and configurations against representative tasks,

measuring rework, AI defects by category, cycle time, and cost impacts so AI usage improves over time.

Systems engineers should go further than simply entering explorations. Specifications should be used as the shared contract between human designers and AI agents, and should be iterated through

experiments to validate alignment. Before asking an AI tool to generate a design, draft requirements, or evaluate alternatives, engineers should invest the time to write down what they actually want: the constraints, objectives, and acceptance criteria, in a structured specification. The specification should not be treated as a one-shot handoff. Engineers should run intermediate experiments and tests to check whether the AI's outputs align with their intent, then refine the specification based on what they learn. In this view, AI tools are far more useful when the problem is well framed and the human-AI loop includes explicit checkpoints for validation. Specifications make the human's intent explicit, give the AI a concrete target to work toward, and create a traceable record of what was asked versus what was produced. Intermediate experiments close the gap between what was intended and what was generated. This aligns with the augmented intelligence paradigm: the human frames the problem, the AI drafts solutions, experiments reveal misalignment, and the specification evolves as the shared contract that keeps both aligned.

So here are the tips:

- 1. Start with a clear purpose and a way to measure success.** Don't start by asking, "How can I use AI?" Like a good systems engineer, start by asking, "What problem am I trying to solve?" Then ask, "How will I structure the investigation?" Treat this as a structured problem-solving exploration. Experiment and learn practical, high-value tasks where AI can help immediately. For each task you have GenAI help with, be clear about the decision you are trying to support, the product or artifact you want to create, and who will review or use the output.
- 2. Use AI as part of a larger engineering process.** AI should be one tool in a larger workflow, not only a standalone chatbot. Use AI where language understanding, summarization, analysis, or idea generation adds value. Create example inputs and expected outputs so you can measure performance and compare alternatives. Continue to use structured engineering methods, models, databases, requirements management tools, and human reviews where they are needed.
- 3. Treat prompting like engineering.** GenAI is great at answering simple questions, but not so great at responding to simple questions with correct answers. You must give the GenAI context with your questions – in other words you must prompt the GenAI with what you want from your answer and in what type of language. Prompting should be handled with the same care you use in engineering specifications. Provide clear objectives, note your assumptions, list constraints, define expected outputs including format, and specify acceptance criteria. Use templates and save successful prompts so they can be reused and improved over time. A useful first step is to ask the AI: "How could this prompt be improved before I use it?"
- 4. Manage context deliberately across sessions and keep a record of AI-assisted work.** When AI contributes to an engineering artifact, preserve enough information for later review: what was asked, what source material was provided, what answer was returned, what was accepted or rejected, and who made the final decision. Do not rely on chat history as project memory. Keep assumptions, decisions, source references, glossary terms, constraints, and prompt templates in durable project artifacts. When returning to a task, restate the current context or provide the maintained reference material so the AI is working from the same baseline as the engineering team. This does not need to be burdensome, but engineering work should remain traceable.
- 5. Let AI ask questions before it answers.** GenAI searches for answers within a "context window." A context window is the "working memory" of the GenAI tool or LLM. It is the maximum amount of data—including your prompt, system instructions, and conversation history—the AI can "see" and reference at any given time. GenAI often fills in the information missing from its context window with assumptions. It needs help with context outside of its narrow view. Encourage it to identify missing information, ambiguous terms, conflicting constraints, and unclear objectives before producing a final answer, and require clarification when needed. Ask the GenAI "What questions do you have before proceeding?" Resolving uncertainty early improves quality and reduces rework. This improves the context.
- 6. Ask for structured outputs.** Remember you are doing systems engineering. Request outputs in forms that are easy to review and reuse. Ask for forms that support review and digital reuse, such as tables, schemas, glossaries, model code, matrices, checklists, and templates, instead of free-form prose. But ask for free-form prose also, such as the GenAI's rationale. Just be careful how you use it. Structured outputs make it easier to validate completeness, compare alternatives, trace decisions, and integrate results into the digital thread or project documentation.
- 7. Use clear rubrics and evaluation criteria.** Don't ask vague questions such as: "Is this design good?" Instead, define specific evaluation criteria such as requirement completeness, traceability, compliance, test coverage, and risk identification. Ask the GenAI for requirements but also ask it to evaluate them based on the INCOSE Guide to Writing Requirements. Use objective true/false or pass/fail checks whenever possible.
- 8. Be transparent about sources and uncertainty.** Always understand what information the AI is using. Ask: What sources are you relying on? What assumptions are you making? What are you uncertain about? What additional information would improve your answer? Prompt the AI to state what context it is using (provided artifacts, retrieved sources, configured knowledge bases, or general model knowledge). This helps reviewers determine how much trust to place in the results.
- 9. Learn how retrieval-augmented generation (RAG) works.** Retrieval-augmented generation (RAG) is a primary pattern for systems engineering because work depends on requirements, standards, specifications, policies, defined workflows, and project-specific artifacts. RAG allows GenAI to use these documents as context rather than relying only on its training data. If you don't know the concepts of "chunking" and "embedding," learn what these mean and practice optimal retrieval performance with the LLM. Treat your document collections, knowledge bases, and custom assistants as engineering assets that require maintenance and governance. GenAI is most useful when it can access the right information, in the right order.
- 10. Build and govern your source data carefully.** Pay attention to source document quality and completeness. Collect and curate high quality source documents, manage chunking and embeddings carefully, maintain vector stores or equivalent indexes, and rebuild them when source content changes. Have the GenAI evaluate your RAG inputs before you go live with these, it can help you manage your input documents for optimal context. Always pay attention to proprietary, export-controlled, or safety-critical information you use and learn how the selected GenAI tool manages data

- storage before giving it your data.
11. **Stay current. AI capabilities change rapidly.** What was impossible six months ago may be easy today. Try multiple GenAI models and compare results. Regularly revisit available models, tools, best practices, and engineering use cases. Avoid designing around limitations that no longer exist.
 12. **Choose models based on evidence, not hype.** Different models perform differently on different tasks. Treat LLM selection and configuration as an engineering trade, and benchmark before standardizing. A smaller, specialized model may outperform a larger model for a specific engineering problem. Cloud-based or open models are easiest to use and have the most context, but smaller “local” models may be much faster and be trained on data closer to your needs. Learn how to run a local model on your computer, you will learn more and it will offer privacy for your data. Ask others which models they like to use for which tasks. Benchmark candidate models against representative tasks before standardizing one. Test before you commit.
 13. **Treat deployment and data constraints as part of the task.** Before using AI, consider where the work is being done and what data is involved. Public cloud, secure cloud, local tools, and air-gapped environments create different constraints. Privacy, export control, proprietary information, customer data, and organizational policy should shape how the AI is used.
 14. **Build for change.** Models, tools, and providers will change faster than your systems. Design workflows that allow you to swap models, change configurations, update thresholds, and move between cloud and secure environments without redesigning the entire system. Your workflows are your tested and stored prompts, along with your RAG. Always maintain and use a baseline of these but experiment with changes to see if you get better results.
 15. **Understand where AI works—and where it doesn't.** AI capabilities advance unevenly - success on one task does not guarantee success on a related task. Study model performance – How does it fail? Does it fail consistently? Can it signal uncertainty? Are failures bounded? Deployment decisions should be based on reliability, not just benchmark scores.
 16. **Spend more time verifying than generating.** AI can create content quickly. Determining whether that content is correct remains the difficult part. Before scaling, test on small samples, review findings, measure false positives and false negatives, and validate results. The engineer's role increasingly becomes reviewer, verifier, and curator. Generating is cheap. Verification and validation are valuable. GenAI does not change a system engineer's job, just their workflow
 17. **Don't trust confidence estimates until you test them.** GenAI confidence estimates quantify an AI system's belief in the correctness, accuracy, or reliability of its output—serving as an essential interface for both human-AI collaboration and automated task delegation. GenAI confidence estimates express the model's sense of statistical certainty, not measured or calibrated probabilities. Because AI generates text by predicting the next most likely word based on its training, a high confidence score may only mean it found a strong pattern—even if that pattern is entirely incorrect (being “confidently wrong”). Training and preference-tuning can also encourage sycophancy, where models tend to generate outputs that conform to or affirm the user's stated or implied beliefs. As a result of these and other factors, GenAI systems are often more confident than they should be. A reported confidence of 90% does not mean the answer is correct 90% of the time. Measure actual performance, then use calibrated confidence estimates to automate decisions. Confidence is not accuracy.
 18. **Pair agent reasoning with deterministic tools.** Use AI agents to orchestrate engineering work, not to replace engineering tools. GenAI is useful for framing problems, interpreting ambiguous information, comparing alternatives, drafting outputs, and explaining results. But calculations, simulations, database queries, requirements traceability, model execution, compliance checks, and formal validation should be performed by deterministic or governed tools. Teach the agent which tools are available, when to use them, what inputs they require, and how to verify their outputs. Build new tools for repeatable tasks, connect them through skills, rules, hooks, and plugins, and refactor workflows so stochastic reasoning is paired with deterministic evidence.
 19. **Design workflows that allow the AI to abstain.** The best GenAI systems know when not to answer. Structure your outputs so they contain confidence estimates. Instruct (prompt) the GenAI to explicitly defer low confidence results rather than guessing. Prompt the system to flag uncertainty, escalate difficult cases, and request human review. If you are building an Agentic AI workflow, include branches to direct results with low estimated confidence to human input. Human expertise should be focused where it creates the most value. A trustworthy system knows when to say, “I don't know.”
 20. **Focus on human-AI teaming and continuous improvement.** Lastly, AI is most effective when it augments human expertise rather than replacing it. Measure your prompt effort, review effort, error and falsification rates, why you have to reject returned information, and where corrective actions improve GenAI performance. Keep detailed notes for yourself or your team. Use those measurements to improve prompts, workflows, models, and training over time. Remember that AI does not automatically understand engineering assumptions. If a junior engineer would need instructions, the AI probably does too. The goal is not automation alone—it is better decisions.

Finally, thank you Barclay Brown for your tireless efforts to bridge AI and systems engineering. We miss you! ■

Automated Curation and Execution of Engineering Models

Jaya Kambhampaty jkambhampaty3@gatech.edu; Olivia Fischer olivia.pinson@asdl.gatech.edu; Dimitri Mavris dimitri.mavris@ aerospace.gatech.edu

Copyright ©2026 by Jaya Kambhampaty, Olivia Fischer, and Dimitri Mavris. Permission granted to INCOSE to publish and use.

■ ABSTRACT

The 2020s have been marked by increased attention to data-intensive information systems that maintain consistency with the evolving state of physical assets. We call the digital models of the connected physical systems “digital twins” (AIAA Digital Engineering Integration Committee 2021), while “digital thread” refers to the foundational framework of protocols, standards, and linked representations that maintains contextual meaning and traceability as data crosses tool, discipline, organizational, and lifecycle boundaries (AIAA Digital Engineering Integration Committee 2023). The practical and theoretical methods for producing these systems make up digital engineering (Griffin 2018), a discipline that formalizes and institutionalizes modeling as a primary, lifecycle-spanning engineering approach and is characterized by improved data sharing, model interoperability, and effective reuse.

INTRODUCTION

In this short paper, we document perspectives on how metadata, or “data about data”, supports the knowledge management central to implementing and maintaining these digital systems. Emphasis on metadata and the infrastructure of building data-intensive systems is an essential part of realizing the “value promise” of digital engineering (Schindel 2022). As models and data proliferate, the methods for managing them and distilling insights from them must evolve. We see this as a need which will only become more pressing for the rest of the 2020s and beyond.

One source of such methods includes the field of artificial intelligence (AI). The intersection of systems engineering and artificial intelligence has long included knowledge representation especially via ontologies and languages for architecture representation and specifications. Now, language model research has yielded innovative methods and tools for performing digital work including AI agent architectures that work

increasingly independently. Open questions remain around how to integrate the promising technologies into existing workflows, including (a) in domains requiring deep expertise (such as engineering) and (b) in collaboration with humans. Furthermore, the contributions of artificial intelligence to the structured representations that might underpin future digital engineering infrastructures extend far beyond natural language processing alone.

From 2020 to 2026, we present developments in systems engineering literature about knowledge management, including metadata, integrating this review with recent advances in artificial intelligence. We present an integrated vision of the role of standardized schemas for representing knowledge (i.e., ontologies, knowledge graphs), data (i.e., metadata schemas), and procedures for using them (i.e., application program interfaces or “APIs”) designed for practitioners, and use that discussion to illustrate the key areas of research through 2030 and beyond.

THE FUTURE DIGITAL ECOSYSTEM

Visions for what a future digital ecosystem could look like emerged early in the 2020s, in response to the US Department of Defense (DoD) digital engineering strategy (United States Department of Defense 2018). In the Department of the Air Force’s Digital Building Code memo (Hunter and Calvelli 2021), integrated digital environments (IDEs) are defined as compilations of data, models, and tools for managing engineering development. These efforts to evolve the development ecosystem centered around the idea that these innovations would help “realize the value promise of digital engineering” (Schindel 2022) by supporting automated approaches for system development, design, testing, evaluation, production, operation, training and sustainment (Office of the Under Secretary of Defense for Research and Engineering 2023; Roper 2021) and modernized reviews (Blackburn and Kruse 2022).

Digital engineering processes rely on models. Modeling and simulation are

required to represent real-world physics and phenomena in a way that supports digital analysis. These models then need to be connected to produce integrated analysis of some system of interest. In fact, techniques that leverage a digital engineering environment like product line engineering may even assume models can be used together (Hause and Hummell 2019; Granrath et al. 2021). However, realizing a digital ecosystem requires formalized engineering processes, analytical methods, governance structures, and decision frameworks that structure how digital artifacts are created, validated, controlled, and used across the lifecycle. This is made difficult by the vast quantities of heterogeneous digital artifacts generated by modern engineering enterprises and the challenges associated with defining interoperability patterns that expose the structure and semantics of digital artifacts.

These artifacts span the product lifecycle, including parametric computer-aided design (CAD) files, finite-element meshes, computational fluid dynamics (CFD) input decks, material datasheets, test reports, maintenance logs, and regulatory submissions. The information embedded in or derivable from these artifacts, including material properties, boundary conditions, dimensional tolerances, design intent, and simulation parameters, are essential for tasks such as design reuse, traceability, automated compliance checking, and the construction of digital twins. Yet this information is frequently locked inside proprietary file formats or expressed only in unstructured natural-language text within PDFs and technical reports. It may also be scattered across siloed software systems, including those for product lifecycle management, enterprise resource planning, and simulation platforms.

Running multiple models in sequence relies on structured interfaces and logic to exchange this information between them. In this paper, we focus on the challenging work of building these interfaces and that logic. This is one element of digital curation, “the active management and enhancement of digital information assets for current and future use” (National Research Council 2015). The conventional approach leverages tools including metadata schemas, ontologies and knowledge graphs, and APIs. These technologies represent three foundational approaches to implementing them. Metadata schemas label data or models with information about their format or use. APIs define a reusable set of components, including data structures, endpoints, formats, and procedures that support the exchange of data and integration of multiple tools. Ontologies and knowledge graphs are ways of representing

knowledge in a specific domain to support reasoning, either about results or data. This includes explicitly encoding rules for verifying modeling artifacts (Gregory and Salado 2024b). These three technologies all represent possible outputs of a curation task and are similar in the way that they encode reliable knowledge about a model or modeling context in reusable ways.

Several reference examples of digital engineering ecosystems prototyped between 2020 and today illustrate how these curation approaches can be applied to digitize, integrate, and automate engineering processes. Namely, they show that high levels of integration are characterized by reliance on ontological modeling of constituent models and semantic web technologies (Hagedorn et al. 2020; Ritter et al. 2022; Gregory and Salado 2024a, 2024b; Dunbar et al. 2023). Once models and data repositories are integrated, analysis can be run automatically (“model execution”). Best practice and standardization continue to evolve today (Gregory et al. 2025).

A critical challenge remains. Automatic execution of modeling work or engineering analysis using models depends on effective curation of models and data, but performing this curation is slow and knowledge-intensive. Existing approaches for achieving these integrated, automated ecosystems tend to be manual. When proposing an approach to automate the transition between functional design and engineering processes, Paniagua observes that this is in many cases “due to a lack of compatibility between the tools” (Paniagua and Caso 2025). As a result, the curation work must include tagging and organization of models with their functional purpose and implementation of interfaces so engineers can perform the integration of a more complex analytical workflow. Ontology engineering is perhaps the most illustrative of the challenges that remain. While ontologies deliver strong semantic guarantees and enable logical reasoning, they impose significant costs. Building and maintaining a comprehensive engineering ontology requires deep domain expertise, substantial manual effort, and continuous curation as standards, tools, and physics models evolve. Critically, every time a new simulation tool, material class, or physics coupling is introduced, the ontology must be extended, a bottleneck that directly conflicts with the scalability requirements of modern multi-physics, multi-tool engineering workflows.

One way to reduce this cost is to automate the curation process itself. Given a collection of engineering models, uncurated, which may be used to perform an engineering analysis, we present a

notional curation procedure to motivate our discussion:

1. Identify the inputs and outputs of those models.
2. Determine how those inputs and outputs can be connected.
3. Implement a method for connecting the model inputs to model outputs.

While this procedure is stylized for clarity, it produces an artifact structurally consistent with the assessment flow diagram formalism introduced by Cilli (2015) and referenced as part of the decision analysis data model (Parnell 2025). The subtasks of each step presented here may vary based on the context. Step 2, for example, will look different based on features of the inputs and outputs like units, domain, and fidelity. Step 3 may require different actions based on if an integration is written from scratch or with a modeling tool's existing integrations. However, this illustrative example allows us to begin defining the areas where new advances in artificial intelligence could prove useful. A system which can automatically solve problems of integration and automation would allow engineers to spend less time running and connecting tools, and more time on questions of validation and verification of their outputs.

CURATION AND AI TODAY

Using techniques from artificial intelligence to support engineering work has been an interest of the systems engineering community as early as 2020 (McDermott et al. 2020). In the 2023 update to the Systems Engineering Research Center (SERC)'s AI and autonomy roadmap, the field had made progress on maturing both the capabilities of data collection and curation and ontological modeling (McDermott et al. 2024). At the same time, AI approaches to achieving the same goals were rapidly becoming possible through advanced natural language processing (Ouyang et al. 2022).

In 2026, large language models (LLMs) have been used in engineering to produce engineering results and act as an assistant to a human by generating models and data (Zhang et al. 2026; Mandegari et al. 2026; Paniagua and Caso 2025). LLM-driven architectures have been proposed to generate several useful engineering artifacts, including CAD (X. Li et al. 2024), OV-1 diagrams (Topcu et al. 2025), and SysML (Zhong et al. 2023). This is driven largely by rapid advances in large language models, statistical models of language which can generate correct sequences of code and other text based on input text sequences that they receive (for example, OpenAI 2023; Team et al. 2025; Yang et al. 2025; Olmo

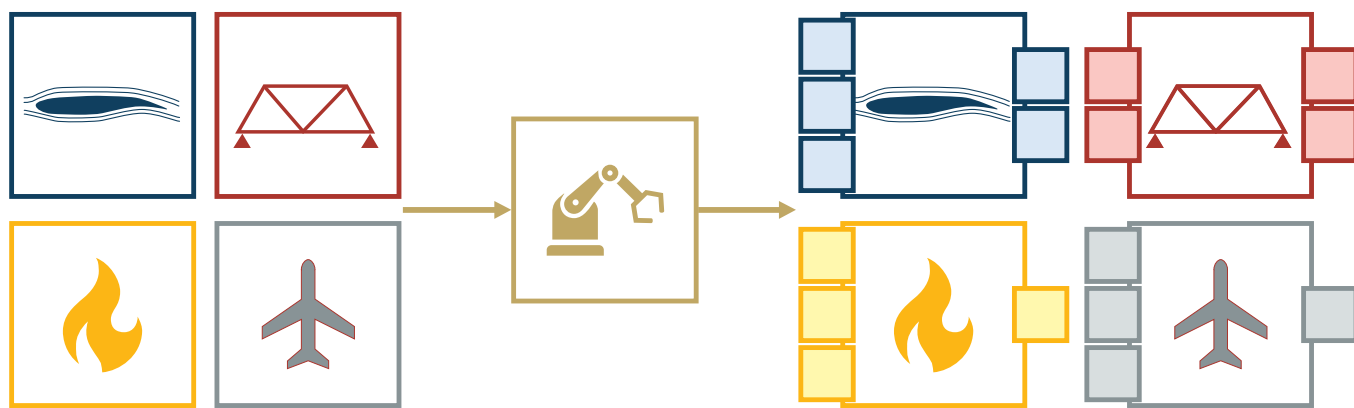


Figure 1. Conceptual diagram of the automatic curation problem. In the automatic curation problem, the interfaces for domain-specific models and tools are discovered and maintained by an artificially intelligent system.

et al. 2025). However, less focus has been given to addressing curation tasks.

Furthermore, these models can achieve strong performance on many tasks when deployed not as a standalone model, but as an LLM system, which incorporates retrieval techniques and structured repositories of information to improve performance on knowledge-intensive tasks, including in engineering (Lewis et al. 2021; AlphaCode Team 2023; Edge et al. 2025; Huang et al. 2026). The essential pattern of an LLM system is that the LLM may be given access to multiple tools for doing or checking work programmatically, for example, external to the LLM. This has the advantage of explicitly encoding the math and logic of a problem in code, which can be verified, rather than relying on the LLM to map the inputs to a correct answer in its weights.

LLMs introduce a fundamentally different technology to the context of model curation and execution. Pre-trained on enormous text corpora that include technical manuals, research papers, engineering standards, and code documentation, LLMs can generate useful statements about engineering concepts and their relationships. This opens the possibility of engineering structured interfaces to existing models with natural-language prompts, even bypassing full curation workflows (for example, explicit schema definition, ontological engineering, API design). It remains a research question as to whether such approaches can achieve sufficient accuracy, consistency, and scalability to serve as practical alternatives to manual curation or if they are simply complements.

AI system progress in the LLM literature is measured on benchmarks: common sets of reference tasks and a way to score the performance of a system that attempts to solve them. The definition of open, shared benchmarks has been a major driver of AI system progress (Mohammadi et al.

2025). The tasks must represent the work that an AI system should do, by providing a sample of inputs and a way to score that system's generated answers. By examining the benchmarks that mirror the work required of an automatic model curation or execution system, we motivate the claim that the technology required to achieve such a system exists, and that such systems can be achieved by the end of the decade. We review developments from the AI literature to substantiate this claim, focusing on what tasks modern LLM-based AI systems are being designed to complete. We relate this to our goal of automatic systems that integrate and run engineering tools.

TOWARDS SOLVING THE AUTOMATIC CURATION PROBLEM

For the illustrative curation process we defined in Figure 1, an example task for Step 1 above would be to provide an AI system with an engineering model and ask it to identify the names of the inputs and outputs of that model. Steps 2 and 3 may involve example tasks including asking the model to look at the inputs and outputs of two models (A and B) that are meant to be executed in sequence, identify the map between the inputs of A and the outputs of B, and implement a wrapper in a general purpose programming language to make the actual connection.

These benchmarks mirror many tasks in AI system benchmarks developed since 2020. Importantly, frontier model performance is increasing across the board on such tasks. Identifying the inputs and outputs of a model is similar to a highly technical question answering task like Google-Proof Question Answering (Rein et al. 2023). Benchmarks for implementing a wrapper around engineering tools are very similar to software engineering benchmarks (for example, SWE-Bench Pro (Deng et al. 2025)).

Despite these similarities, explicit curation benchmarks do not exist for engineering artifacts and are underrepresented in the AI literature (Di et al. 2026), perhaps because of the closeness of the curation task to code generation tasks. Despite this apparent similarity, it is critical to continue evaluating LLM Systems in exactly their intended use cases, as there are not strong guarantees that the performance will transfer between domains. Since a system that can effectively automate curation makes integrating tools much easier for practitioners, this is likely to be a growing area of research for the rest of the decade.

TOWARDS SOLVING THE AUTOMATIC EXECUTION PROBLEM

Earlier, we described an LLM system as internally containing an LLM augmented with external tools for running explicit computations. As these systems have incorporated these additional components, benchmarks have also grown to reflect increasingly complex tasks. Early LLM benchmarks, such as MMLU (Hendrycks et al. 2021) and HumanEval (Chen et al. 2021) focused on limited tasks such as question answering or code completion in isolation. In these cases, the performance of an LLM on a benchmark indicated how many questions were answered correctly from some set. However, new benchmarks, for example, Toolathlon, BrowseComp, GDPval (J. Li et al. 2026; Wei et al. 2025; Patwardhan et al. 2025) demonstrate a substantial increase in task complexity, where models perform work in increasingly realistic environments by getting access to tools. In these cases, the benchmark results measure if the LLM called tools in the right order and produced the correct answer (Figure 2). The analogy between this measurement approach and the automatic execution of engineering models is very strong. Many tools that have had years of use through the Internet (for

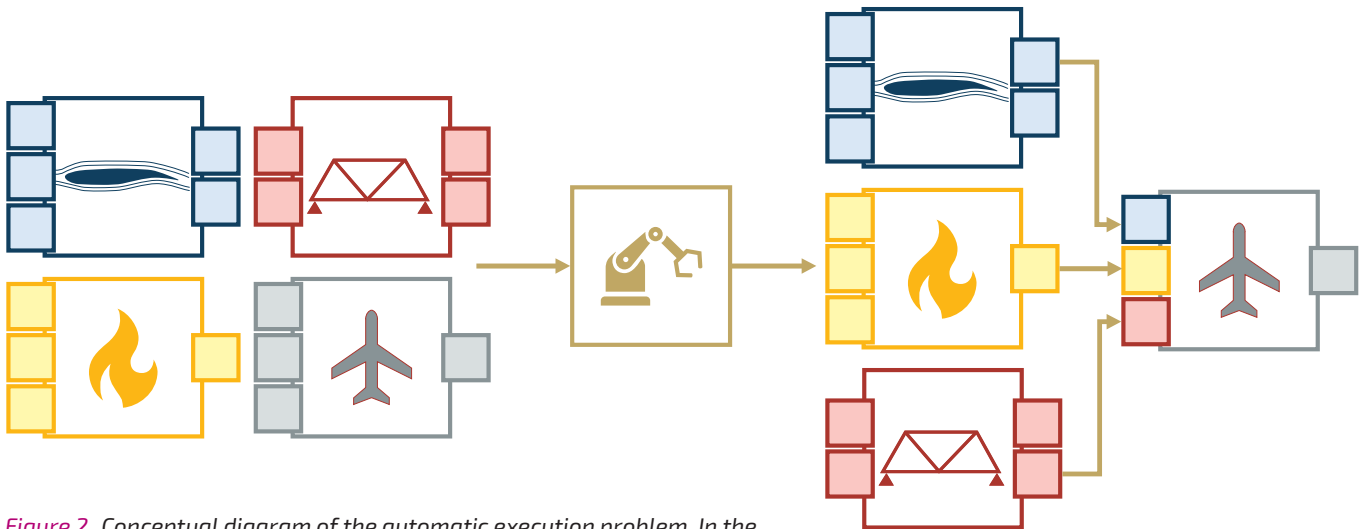


Figure 2. Conceptual diagram of the automatic execution problem. In the automatic execution problem, an artificially intelligent system implements the connections between different modeling tools or environments so they can be run in a multi-step process to generate a solution.

example, Google Calendar) have already implemented internal application programming interfaces (APIs). This made the implementation of such benchmarks very fast. Substantial work remains to catalog the interfaces of engineering tools, especially custom tools where interfaces may need to be implemented. This work could be greatly reduced by a system that successfully implements the automatic curation task.

CONCLUSION

Advanced automation has often relied on techniques from artificial intelligence. Knowledge-based approaches such as ontologies have been used to great success in digital engineering ecosystems to integrate models and automate analysis. However, they have relied on painstaking manual

curation processes. An examination of existing LLM benchmarks finds many tasks similar to those that make up curation and execution; it seems reasonable to suggest that curation and automation can be substantially more automated before the end of this decade.

This cannot be achieved without intentional research. While the framework exists for producing an automatic curation system benchmark, there is no standardization or consensus on the specific curation tasks and output formats that should be included. As we mentioned before, metadata, ontologies, and APIs are all valid targets for curation task outputs. Automatic execution is similar to tool use, an area of LLM system research that received significant attention through

the end of 2025 into 2026. Nevertheless, collecting sets of problems that specifically represent engineering workflows is critical for rigorously assessing novel AI systems.

For both tasks, collaboration between engineers building these AI systems and curation practitioners will help identify the subtasks within curation activities that are best suited for automation, and thus where to start building benchmarks from. Examples should cover many models, diverse in complexity, language, and topic area. Finally, sharing these datasets openly amongst the systems engineering community is imperative for making progress; the prolific literature on LLM system benchmarks provides an inspirational example. ■

REFERENCES

- AlphaCode Team. 2023. “AlphaCode 2 Technical Report.” White paper, Google DeepMind. https://storage.googleapis.com/deepmind-media/AlphaCode2/AlphaCode2_Tech_Report.pdf.
- American Institute for Aeronautics and Astronautics Digital Engineering Integration Committee. 2021. “Digital Twin: Definition and Value” White paper, American Institute for Aeronautics and Astronautics and Aerospace Industries Association. <https://aiaa.org/wp-content/uploads/2024/12/digital-twin-institute-position-paper-december-2020.pdf>.
- American Institute for Aeronautics and Astronautics Digital Engineering Integration Committee. 2023. “Digital Thread: Definition, Value, and Reference Model” White paper, American Institute for Aeronautics, Astronautics, Aerospace Industries Association, and NAFEMS. https://aiaa.org/wp-content/uploads/2025/04/digital-thread-implementation-paper_june_2023.pdf.
- Anil, Rohan, Sebastian Borgeaud, et al. 2025. “Gemini: A Family of Highly Capable Multimodal Models.” arXiv:2312.11805. Preprint, arXiv, May 9. DOI:10.48550/arXiv.2312.11805.
- Blackburn, Mark R., and Benjamin Kruse. 2022. “Conducting Design Reviews in a Digital Engineering Environment.” *INSIGHT* 25 (4): 42–46. DOI:10.1002/inst.12410.
- Chen, Mark, Jerry Tworek, Heewoo Jun, et al. 2021. “Evaluating Large Language Models Trained on Code.” arXiv:2107.03374. Preprint, arXiv, July 14. DOI:10.48550/arXiv.2107.03374.
- Cilli, M. 2015. Seeking Improved Defense Product Development Success Rates Through Innovations to Trade-Off Analysis Methods, Dissertation, Stevens Institute of Technology, Nov.
- Deng, Xiang, Jeff Da, Edwin Pan, et al. 2025. “SWE-Bench Pro: Can AI Agents Solve Long-Horizon Software Engineering Tasks?” Version 2. Preprint, arXiv. DOI:10.48550/ARXIV.2509.16941.
- Di, Shimin, Xujie Yuan, Hanghui Guo, et al. 2026. “ToolRosetta: Bridging Open-Source Repositories and Large Language Model Agents through Automated Tool Standardization.” arXiv:2603.09290. Version 1. Preprint, arXiv, March 10. DOI:10.48550/arXiv.2603.09290.

- Dunbar, Daniel, Thomas Hagedorn, Mark Blackburn, et al. 2023. “Driving Digital Engineering Integration and Interoperability through Semantic Integration of Models with Ontologies.” *Systems Engineering* 26 (4): 365–78. DOI:10.1002/sys.21662.
- Edge, Darren, Ha Trinh, Newman Cheng, et al. 2025. “From Local to Global: A Graph RAG Approach to Query-Focused Summarization.” arXiv:2404.16130. Preprint, arXiv, February 19. DOI:10.48550/arXiv.2404.16130.
- Granrath, Christian, Christopher Kugler, Sebastian Silberg, et al. 2021. “Feature-Driven Systems Engineering Procedure for Standardized Product-Line Development.” *Systems Engineering* 24 (6): 456–79. DOI:10.1002/sys.21596.
- Gregory, Joe, Clarence (Moe) Moreland, James S. Wheaton, and Celia Tseng. 2025. “The Need for a Shared Vocabulary of Digital Engineering.” *INSIGHT* 28 (5): 58–61. DOI:10.1002/inst.70009.
- Gregory, Joe, and Alejandro Salado. 2024a. “An Ontology-Based Digital Test and Evaluation Master Plan (dTEMP) Compliant with DoD Policy.” *Systems Engineering* 27 (6): 1012–26. DOI:10.1002/sys.21769.
- Gregory, Joe, and Alejandro Salado. 2024b. “Towards a Systems Engineering Ontology Stack.” *INCOSE International Symposium* 34 (1): 1304–18. DOI:10.1002/iis2.13210.
- Griffin, Michael D. 2018. “United States Department of Defense Digital Engineering Strategy.” White paper, Office of the Deputy Assistant Secretary of Defense for Systems Engineering. <https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy-Approved-PrintVersion.pdf>.
- Hagedorn, Thomas, Mary Bone, Benjamin Kruse, Ian Grosse, and Mark Blackburn. 2020. “Knowledge Representation with Ontologies and Semantic Web Technologies to Promote Augmented and Artificial Intelligence in Systems Engineering.” *INSIGHT* 23 (1): 15–20. DOI:10.1002/inst.12279.
- Hause, Matthew, and James Hummell. 2019. “Model-Based Product Line Engineering—Enabling Product Families with Variants.” *INSIGHT* 22 (2): 43–48. DOI:10.1002/inst.12245.
- Hendrycks, Dan, Collin Burns, Steven Basart, et al. 2021. “Measuring Massive Multitask Language Understanding.” Paper presented at International Conference on Learning Representations. Proceedings of the 9th International Conference on Learning Representations, May 3. <https://arxiv.org/abs/2009.03300v3>.
- Huang, Yinchien, Tien-Yueh Fung, and Daniel A. DeLaurentis. 2026. “Addressing Complexity in System of Systems With GraphRAG: An AI-Driven Framework for Dynamic Data Integration.” *Systems Engineering* 29 (2): 150–68. DOI:10.1002/sys.70012.
- Hunter, Andrew, and Frank Calvelli. 2021. “Digital Building Code for the Transformation of Acquisition and Sustainment.” <https://guide.dafdto.com/wp-content/uploads/2023/01/Digital-Building-Code-2022-Memo-Signature-Final.pdf>.
- Lewis, Patrick, Ethan Perez, Aleksandra Piktus, et al. 2021. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” arXiv:2005.11401. Preprint, arXiv, April 12. DOI:10.48550/arXiv.2005.11401.
- Li, Junlong, Wenshuo Zhao, Jian Zhao, et al. 2026. “The Tool Decathlon: Benchmarking Language Agents for Diverse, Realistic, and Long-Horizon Task Execution.” arXiv:2510.25726. Preprint, arXiv, February 26. DOI:10.48550/arXiv.2510.25726.
- Li, Xingang, Yuewan Sun, and Zhenghui Sha. 2024. “LLM4CAD: Multimodal Large Language Models for Three-Dimensional Computer-Aided Design Generation.” *Journal of Computing and Information Science in Engineering* 25 (021005). DOI:10.1115/1.4067085.
- Mandegari, Siavash, Mayank J. Bhalerao, Janet K. Allen, and Farrokh Mistree. 2026. “Generative AI for Interpretable Satisficing Solution Design in Manufacturing.” *Journal of Computing and Information Science in Engineering*, March 24, 1–25. DOI:10.1115/1.4071473.
- McDermott, Tom, Dan DeLaurentis, Peter Beling, Mark Blackburn, and Mary Bone. 2020. “AI4SE and SE4AI: A Research Roadmap.” *INSIGHT* 23 (1): 8–14. DOI:10.1002/inst.12278.
- McDermott, Tom, Kara Pepe, and Megan Clifford. 2024. “The Updated SERC AI and Autonomy Roadmap 2023.” *INCOSE International Symposium* 34 (1): 1135–48. DOI:10.1002/iis2.13200.
- Mohammadi, Mahmoud, Yipeng Li, Jane Lo, and Wendy Yip. 2025. “Evaluation and Benchmarking of LLM Agents: A Survey.” Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (New York, NY, USA), KDD ’25, August 3, 6129–39. DOI:10.1145/3711896.3736570.
- National Research Council. 2015. “Preparing the Workforce for Digital Curation.” Washington, DC: The National Academies Press. DOI:10.17226/18590.
- Office of the Under Secretary of Defense for Research and Engineering. 2023. DoD Instruction 5000.97 Digital Engineering. DoD Instruction 5000.97. U.S. Department of Defense. <https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500097p.PDF>.
- Olmo, Team, Allyson Ettinger, Amanda Bertsch, et al. 2025. “Olmo 3.” arXiv:2512.13961. Preprint, arXiv, December 15. DOI:10.48550/arXiv.2512.13961.
- OpenAI. 2023. “GPT-4 Technical Report.” arXiv:2303.08774. Preprint, arXiv, March 27. DOI:10.48550/arXiv.2303.08774.
- Ouyang, Long, Jeff Wu, Xu Jiang, et al. 2022. “Training Language Models to Follow Instructions with Human Feedback.” arXiv:2203.02155. Preprint, arXiv, March 4. DOI:10.48550/arXiv.2203.02155.
- Paniagua, Cristina, and Fernando Labra Caso. 2025. “Models2Code: Autonomous Model-Based Generation to Expedite the Engineering Process.” *Systems Engineering* 28 (2): 224–37. DOI:10.1002/sys.21789.
- Parnell, G. S., C. R. Kenley, D. Clark, J. Smith, F. Salvatore, C. Nwobodo, and S. Davis. 2025. The Decision Analysis Data Model. *INSIGHT*, 28: 44–53. DOI:10.1002/inst.70011
- Patwardhan, Tejal, Rachel Dias, Elizabeth Proehl, et al. 2025. “GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks.” arXiv:2510.04374. Preprint, arXiv, October 5. DOI:10.48550/arXiv.2510.04374.
- Rein, David, Betty Li Hou, Asa Cooper Stickland, et al. 2023. “GPQA: A Graduate-Level Google-Proof Q&A Benchmark.” arXiv:2311.12022. Preprint, arXiv, November 20. DOI:10.48550/arXiv.2311.12022.
- Ritter, Christopher, Jeren Browning, Peter Suyderhoud, et al. 2022. “Versatile Test Reactor Open Digital Engineering Ecosystem.” *INSIGHT* 25 (1): 56–60. DOI:10.1002/inst.12374.
- Roper, William. 2021. “Bending_the_Spoon: Guidebook for Digital Engineering and e-Series.” January 19. https://www.af.mil/Portals/1/documents/2021SAF/01_Jan/Bending_the_Spoon.pdf.
- Schindel, William D. 2022. “Realizing the Value Promise of Digital Engineering: Planning, Implementing, and Evolving the Ecosystem.” *INSIGHT* 25 (1): 42–49. DOI:10.1002/inst.12372.

> continued on page 42

Trustworthy LLM-Augmented Data Reconciliation for Deployment-Constrained Environments

Danny J. O'Leary, and Allison B. Ledford

Copyright ©2026 by Danny J. O'Leary, and Allison B. Ledford. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Digital engineering initiatives depend on trustworthy data lineage and a shared understanding of what enterprise data means. In practice, those conditions are often absent. The same concept may be represented in databases, forms, spreadsheets, email, and conversation-based exchanges, with different labels, different levels of structure, and different degrees of documentation. Organizations attempting to build a digital thread or expand model-based systems engineering (MBSE) often discover that the limiting factor is not tool availability. It is uncertainty about how data actually moves through work and how apparently related elements should be interpreted across sources (Ledford 2023; Ledford, Harris, and Purdy 2023; Ledford et al. 2025).

That problem is the foundation for data element mapping and analysis (DEMA). DEMA was developed to make hidden data work visible by tracing information at the levels of function, vessel, and individual data element. Prior DEMA applications have shown that a large share of enterprise data handling remains undocumented and that many exchanges still depend on manual transfer rather than integrated digital flow (Ledford 2023; Ledford, Harris, and Purdy 2023). Those findings are important because a digital thread is only as reliable as the organization's understanding of the data relationships it claims to connect.

INTRODUCTION

This article addresses one specific bottleneck within that broader data element mapping and analysis (DEMA) workflow: data element reconciliation. Once an organization begins cataloging elements across heterogeneous sources, it must determine which elements describe the same underlying concept and which do not. That activity is currently performed largely through pairwise expert review. It yields defensible decisions, but it does not scale efficiently when organizations uncover hundreds or thousands of element references across many vessels.

The purpose of this article is therefore narrow and practical. It examines how large language models can support reconciliation

in deployment-constrained systems engineering environments when embedded in a disciplined, evidence-based, human-in-the-loop workflow. Specifically, this paper presents DEMA-LLM, a hybrid workflow in which LLM-based semantic adjudication, deterministic filtering, and explicit policy enforcement combine to support data-element reconciliation at scale. In proof-of-concept evaluation, the same code base supported both cloud and local deployment configurations, with quality, cost, and review-burden tradeoffs varying systematically by environment. The central claim is not that generative AI can automate DEMA end to end; rather, this work argues that AI can contribute useful, traceable decision support to one difficult systems engineer-

ing task, provided that trustworthiness, traceability, and analyst authority remain central.

All quantitative results in this article are proof-of-concept measurements from synthetic and small real-world test sets (tens of pairs); we report them with binomial confidence intervals and identify pilot-scale validation as the next step.

DEMA AND THE RECONCILIATION BOTTLENECK

DEMA is the first methodology to holistically uncover the tacit (tribal) knowledge of how individual units of data (data elements) flow across system functions, vessels, and disparate places of storage within an enterprise (Ledford

Table 1. Key DEMA terminology and definitions

Term	Definition
Functional Area	The highest levels by which the functional activities in the system can be grouped.
Sub-Functional Area	Sub-groupings of the functional activities within the functional areas.
Functional Activity	The activities in the system that transform data vessel inputs into outputs.
Data Vessel	The documents, emails, personal notes, drawings, CAD files, and any other possible container (i.e., vessel) of data.
Data Element	The individual pieces of data contained within data vessels such as document titles, dimensions, software file inputs, individual requirements, and due dates.

2023; Ledford et al. 2024). Previous visual mapping techniques, such as value stream mapping, integration definition for function modeling (IDEF0), data flow diagrams, and business process model and notation (BPMN), are suitable for evaluating functional, document-based, and software-centric views of data and information and do not consider the individual units of data that would make up a true digital thread (Ledford, Harris, and Purdy 2023).

DEMA provides a structured way to map systems at three levels: 1) functional level, 2) data vessel level, and 3) data element level. Key DEMA terminology is defined in Table 1. Previous applications of DEMA have been conducted to architect the digital thread in a prototyping organization (Ledford, Harris, and Purdy 2023)

and enable model-based systems engineering for the development of a verification and validation process of a modeling and simulation environment (Ledford et al. 2025). Each of the three levels of DEMA mappings are created by a series of structured interviews with subject matter experts. The final step of DEMA, data element level creation, is where a data element level view table is created that captures the flow of data elements across all system functions, vessels, and places of storage between system actors. From the data element level view, an improved state data element level view can be created with a return on investment (ROI) based on the reduction of labor hours associated with reduced data handling (Ledford 2023).

The potential payoff from the realiza-

tion of the digital thread is immense. An improved data architecture created from an initial DEMA application was calculated to eliminate around \$2 million worth of labor hours associated with manual data handling (Ledford 2024). However, despite significant cost savings that can be enabled by DEMA, the current DEMA process is manual and tedious, resulting in a significant barrier to adoption. The key bottleneck of the DEMA application process lies in the manual mapping and reconciliation of interview results. Reconciliation of DEMA results becomes necessary when multiple stakeholders describe overlapping system realities differently. Figure 1 overviews the DEMA workflow with manual reconciliation at each of the three DEMA steps. The data element layer is especially costly because it often uncovers hundreds or thousands of elements across many vessels. For that reason, this article focuses specifically on data-element reconciliation as the most immediate and tractable opportunity for trustworthy LLM support within the broader DEMA workflow.

WHY TRADITIONAL APPROACHES FALL SHORT

Traditional reconciliation methods remain useful, but they do not fully address the type of evidence DEMA exposes. Rules-based matching performs well when names are standardized and the problem is largely syntactic. Schema-matching and traditional master data management approaches are effective when the task is a clean

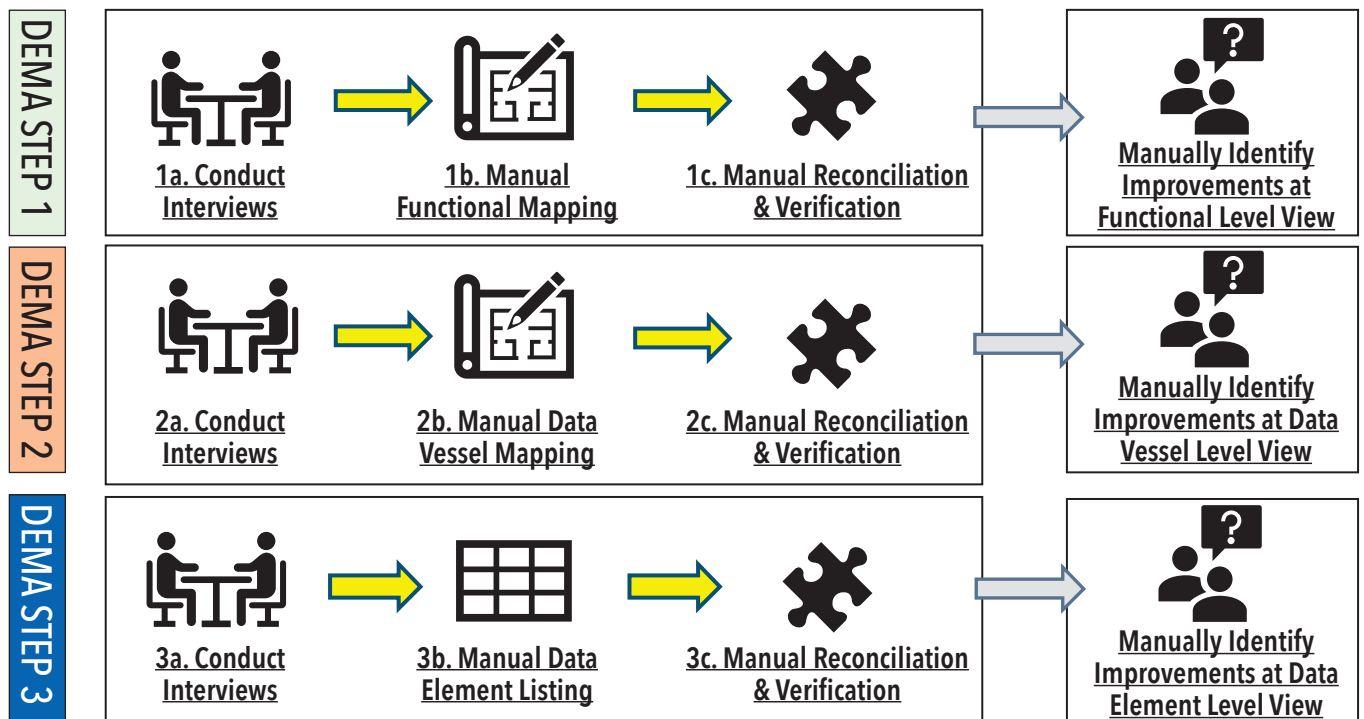


Figure 1. Manual DEMA application workflow

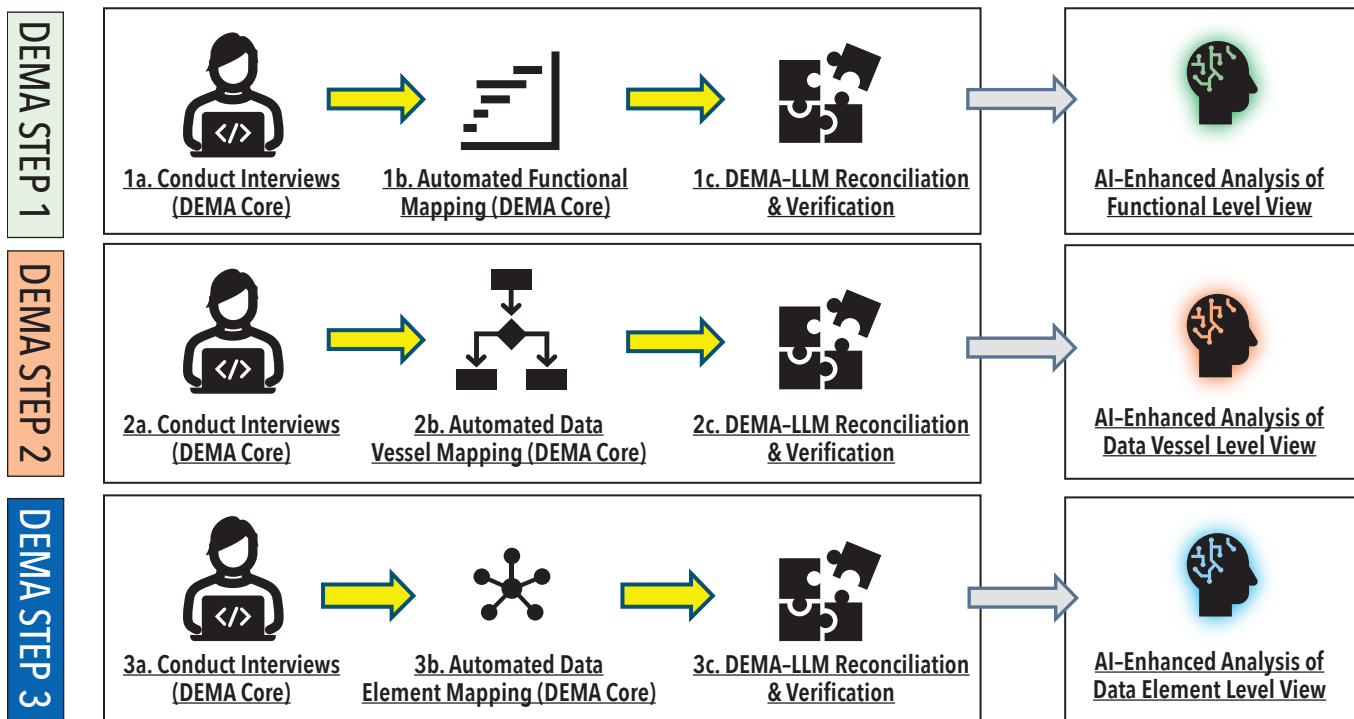


Figure 2. DEMA-LLM enhanced DEMA application workflow

database-to-database alignment. Embedding-based similarity can improve lexical matching by capturing semantic proximity. Each of these approaches contributes value.

The difficulty is that DEMA often reveals a more heterogeneous operating environment. The same concept may appear in a structured system field, a spreadsheet created for handoff, a request form, a project note, an interview transcript, or an email exchange. In that setting, reconciliation is not simply a matter of comparing strings. It requires judging whether the available evidence supports semantic equivalence despite different wording, vessel types, and local usage patterns.

Manual review can make those distinctions, but it is slow, difficult to scale, and error-prone. A purely generative approach is also insufficient. In systems engineering contexts, plausible answers are not enough. Recommendations must be reviewable, traceable to evidence, and compatible with the deployment constraints under which the organization operates. From that combination of requirements a hybrid approach emerges: deterministic where possible, LLM-assisted where necessary, and always accountable to analyst review.

WHAT THIS MEANS FOR SYSTEMS ENGINEERING PRACTICE

This special issue asks how AI and automation in systems engineering have progressed across the decade from 2020 to 2030. DEMA-LLM fits that theme because it reflects a shift from broad discussion of

AI potential to bounded support for specific engineering tasks. Early AI4SE conversations often emphasized future possibilities such as optimization, autonomy, and broad tool transformation. Those conversations were valuable, but they were sometimes distant from the practical bottlenecks that slow systems engineering work on actual programs.

Now, generative AI has made a new class of problems more tractable: tasks that are text-heavy, context-dependent, and resistant to purely rules-based automation. Data reconciliation is one such problem. It is not a peripheral activity. It sits near the foundation of digital engineering because traceability, integration, and model consistency all depend on reliable interpretation of the underlying data.

DEMA-LLM also illustrates a second development that is likely to matter even more as the decade progresses. Systems engineering is not only applying AI to engineering work. It is also adapting its own methods to govern AI-enabled tools in a disciplined way. Evidence, traceability, reviewability, and deployment realism are not secondary concerns; they are part of what makes an AI-enabled capability acceptable for practice. Our earlier AI4SE workshop framing emphasized the broader direction of AI-enhanced DEMA (O'Leary and Leford 2025). The current article presents a narrower and more operational view: useful AI support emerges when the workflow is bounded, inspectable, and aligned with the way practitioners actually work.

DEMA-LLM AS A TRUSTWORTHY RECONCILIATION AID

A DEMA software platform (DEMA Core) has been created to prompt interviewees to answer DEMA interview questions and then have the software automatically create DEMA mappings (thus semi-automating the data collection and mapping steps 1a, 1b, 2a, 2b, 3a, and 3b of the manual DEMA workflow shown in Figure 1). DEMA-LLM is designed as a recommendation engine within the DEMA workflow, not as an autonomous reconciliation service. Its purpose is to help analysts evaluate candidate data-element pairs more efficiently by combining deterministic filtering, LLM-based semantic interpretation, and explicit policy checks (thus semi-automating steps 1c, 2c, and 3c of the manual DEMA workflow shown in Figure 1). The intended division of labor is straightforward: DEMA-LLM recommends, and human analysts working in DEMA Core decide. This improved DEMA Core and DEMA-LLM workflow is shown in Figure 2. Although the broader DEMA workflow includes reconciliation at multiple levels, the proof of concept described in this article is limited to the data-element reconciliation stage.

The DEMA-LLM workflow begins with triage. Because the full pairwise comparison space grows rapidly as element inventories expand, DEMA-LLM first uses fast similarity methods to surface promising matches and assemble candidate pairs for adjudication. Those methods include

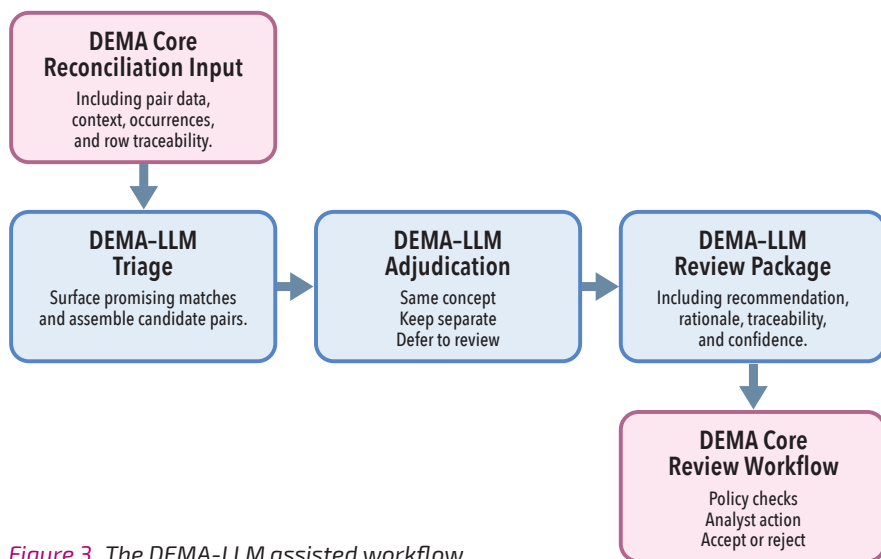


Figure 3. The DEMA-LLM assisted workflow

lexical similarity, co-occurrence signals, and semantic similarity measures. The resulting candidate pair records include enterprise context such as function or process location, vessel and vessel type, actor involvement, and traceability to specific source rows or occurrences. This evidence package is essential because reconciliation depends on how an element is used, not only on how it is labeled.

The adjudication process then evaluates each candidate pair as a three-outcome classification problem. The result recommends one of three actions: treat the elements as the same concept, keep them separate, or defer the case to analyst review when the evidence is insufficient or conflicting. That third outcome is important. In a forced binary formulation, ambiguity is often hidden inside a low-confidence negative judgment. Our approach handles uncertainty explicitly and operationally.

Each recommendation is delivered as a structured review package rather than as a free-form answer. DEMA-LLM returns a recommendation, a short rationale, traceability to the relevant source rows or occurrences, and a confidence estimate. DEMA Core presents that information to the analyst as part of the reconciliation workflow, with an interface designed to facilitate acceptance, rejection, or further review. This design preserves analyst authority while reducing the amount of manual effort required to examine each pair from scratch.

That same separation of concerns puts policy enforcement in DEMA Core. Before presenting available actions to the analyst, DEMA Core verifies that the recommendation by DEMA-LLM satisfies the evidence and threshold requirements for the selected operating configuration. It then determines whether the case can be offered for analyst

batch approval or must remain in the review queue. The analyst sees not only the recommendation and rationale, but also the actions that remain permissible under policy. This is a key trustworthiness mechanism: the model cannot bypass policy simply by producing plausible language.

This entire architecture is shaped by deployment realities as much as by model behavior. It supports a single workflow across cloud-connected, secure-cloud, and fully air-gapped environments. A modern configuration-as-code approach is used for deployment settings, allowing choices such as model provider and operating thresholds to be changed, baselined, and version controlled without modifying the reconciliation logic. That flexibility matters in systems engineering settings where organizations may move between commercial, government-approved, and isolated environments. It also reinforces the advisory character of the system: the appropriate level of automation, review burden, and thresholding depends on the deployment context in which the analyst is working.

WHAT THE PROOF OF CONCEPT TAUGHT US

The most important result of the proof of concept was not a single performance value. It was a clearer understanding of what practitioners should expect when they apply LLMs to reconciliation work in realistic systems engineering settings.

First, deployment constraints must be treated as first-order design parameters. In many organizations, the key question is not whether an LLM can help in principle, but whether the workflow can operate under the network, security, and infrastructure conditions that actually exist. Those constraints affect not only provider selection, but also acceptable thresholds, review burden, and the amount of automation that is prudent.

Second, the hybrid pipeline is what makes LLM support operationally usable. LLMs are valuable for interpreting heterogeneous evidence, but their outputs are probabilistic and require bounded use, policy checks, and analyst oversight. Deterministic triage provides repeatability and cost control. LLM adjudication addresses the cases where semantic interpretation is needed. DEMA Core then applies post-adjudication gates before recommendations are acted upon. This separation of roles is central to adoption because it gives analysts a workflow they can inspect and understand.

Third, cross-vessel equivalence requires explicit policy support. One of the clearest behaviors observed in early work was a tendency to over-separate element pairs drawn from different source types. Email-to-database and conversation-to-system comparisons were especially affected. In DEMA practice, however, vessel differences often reflect context rather than contradiction. If that policy is not written into the workflow, the system will tend to preserve separations that analysts would judge to be unnecessary.

In an early synthetic-data evaluation, the rate of correctly recovering same-concept cross-vessel pairs improved from 0 of 12 to 10 of 12 gold-labeled pairs (95% confidence interval 0.55 to 0.95) only after that policy was made explicit in the prompts and review logic. Despite the small sample size and wide confidence interval, the direction is unambiguous because zero merges were recovered before the rule was encoded. Cross-vessel equivalence is one example, albeit an important one, of a broader rule. Even seemingly obvious foundational assumptions must be made explicit if the workflow is to behave in a way analysts recognize as reasonable.

Fourth, confidence estimates matter operationally only if they are calibrated and monitored. Large language models are known to overstate their confidence when asked to estimate it. Since those estimates are used to drive merge thresholds and abstention bands, calibration must be part of the system design rather than a secondary evaluation exercise.

Fifth, the quality gap between cloud and local models is real, but it does not eliminate the value of local deployment. In proof-of-concept tests, a cloud-hosted configuration achieved roughly 90-95 percent precision on accepted merge recommendations at less than \$0.04 per decision, while one local open-weight configuration was closer to 80 percent precision at infrastructure cost only. Sample sizes are small (tens of MERGE decisions per configuration) and the corresponding 95% confidence

Table 2. Comparison of commercial cloud, secure cloud, and local air-gapped operating assumptions, including likely model quality, review burden, and infrastructure implications

Consideration	Cloud-Connected	Secure Cloud	Air-Gapped/Local
Typical Model Access	Commercial APIs	Approved cloud APIs	Local open-weight models
Expected Quality	Highest current quality ceiling	Moderate to high, subject to approved providers	Lower current quality ceiling
Cost Profile	Per-decision API cost	Usage cost plus approved environment overhead	Infrastructure cost only
Review Burden	Lowest likely review burden	Moderate review burden	Highest likely review burden
Main Constraint	Data-sharing and governance limits	Compliance and approved-service limits	Lower model quality and local operations burden

intervals overlap, so these figures are directional indicators rather than calibrated estimates. For practitioners, the lesson is not that one setting is universally preferable. It is that configuration choice must be evaluated empirically against the actual deployment context, because quality, cost, latency, and review burden move together.

Finally, human review remains central. Reconciliation is not only a pattern-recognition task; it is also a governance task. Analysts bring knowledge of scope, authority, lifecycle, and local practice that may not be visible in the source evidence. The role of the model is to improve the first-pass analysis, provide traceable support for a recommendation, and focus human

attention where it is most needed. In that context, deferring ambiguous cases is not a weakness. It is part of what makes the workflow trustworthy.

A PRACTICAL PILOT PATH FOR ORGANIZATIONS

Organizations interested in this type of LLM capability should begin with a bounded pilot designed to answer operational questions rather than to demonstrate automation for its own sake.

1. Select a process area where reconciliation burden is already visible. Suitable pilots have recurring terminology mismatches, heterogeneous source ma-

terials, and a review scope that can be managed by a small analyst group.

2. Operate initially in shadow mode or review-assisted mode. The objective at this stage is to evaluate recommendation quality and analyst usability, not to transfer authority away from the review process.
3. Build a reviewed evaluation set before expanding scope. A practical starting point is 200-500 labeled observations, which is enough to compare candidate configurations and estimate precision, coverage, and calibration with more than anecdotal confidence.
4. Define a small set of success metrics and thresholds in advance. At minimum,

Table 3. Recommended pilot metrics: Precision@MERGE, coverage, expected calibration error, analyst review time per pair, usefulness of provided traceability, configuration-specific cost or latency, and deployment feasibility by environment

Metric	Why It Matters	Early Pilot Use
Precision@MERGE	Protects against false merges and provides the clearest quality measure for accepted merge recommendations.	Treat 80 percent as a minimum floor and 90 percent as the desired operating point.
Coverage (1-abstain rate)	Shows how often the system produces recommendations that can move forward in the workflow.	Balance against precision; very low coverage may leave too much work with analysts.
Expected Calibration Error	Indicates whether reported confidence aligns with observed correctness.	Use a threshold below 0.10 before relying on confidence to support routing or thresholds.
Analyst Review Time per Pair	Measures whether the workflow reduces review burden rather than simply shifting it.	Compare review-assisted performance against a manual-review baseline.
Usefulness of Provided Traceability	Tests whether the rationale and evidence package actually helps analysts make decisions.	Capture through reviewer feedback, acceptance patterns, and override notes.
Configuration-Specific Cost or Latency	Shows whether a given configuration is practical enough for routine use.	Compare cost per decision and response time across cloud, secure-cloud, and local options.
Deployment Feasibility by Environment	Confirms whether the workflow remains usable under real infrastructure and security constraints.	Record required adjustments, operational constraints, and review burden in each environment.

track Precision@MERGE, coverage (1–abstain rate), analyst review effort, usefulness of the provided traceability, latency or cost by configuration, and calibration if confidence is used to direct workflow decisions. Our current design treats 80 percent precision as a minimum floor, 90 percent as the desired operating point, and expected calibration error below 0.10 as the threshold for trusting confidence estimates in workflow decisions. These are design targets, not measured outcomes; in a pilot, report each metric with its sample size and a 95% confidence interval, and count a target as met only when the lower bound clears it.

5. Compare deployment contexts honestly using the same reviewed set. If commercial cloud access is allowed, it may provide the strongest initial operating point. If secure-cloud or air-gapped deployment is required, run the same evaluation through those configurations, tune thresholds for that environment, and plan for higher review burden if local quality is lower.
6. Use analyst decisions to build a reviewed corpus over time. That corpus supports better calibration, threshold selection, later expansion of the pilot scope, and gradual improvement in precision, coverage, and review burden over time.

The near-term objective should be validation of a trustworthy decision-support workflow, not immediate autonomous reconciliation. Even an imperfect pilot can be valuable if it clarifies the organization's data conditions, review burden, and deployment requirements. A successful pilot, by contrast, creates the foundation for measured expansion into broader reconciliation use.

CONCLUSION: TOWARD 2030

Data element reconciliation is a useful test case for AI in systems engineering because it sits between strategic digital-engineering ambition and the practical realities of enterprise work. It is necessary, labor-intensive, and often shaped by fragmented evidence spread across systems, documents, and informal exchanges.

DEMA-LLM suggests that LLMs can contribute meaningfully to that problem today, but only when they are embedded in a disciplined workflow: deterministic preprocessing, explicit policy, traceable evidence, calibrated thresholds, and retained analyst authority. The central lesson is therefore not about autonomous judgment. It is about accountable augmentation.

If the first half of the decade demonstrated that generative AI could interpret messy engineering evidence, the second half should determine how to operationalize that capability responsibly. By 2030,

the most useful AI4SE capabilities may be those that make difficult engineering judgments more reviewable, more scalable, and more defensible without removing the human accountability on which systems engineering depends. ■

Note from the Authors

The Interdisciplinary Center for Advanced Manufacturing Systems (ICAMS) at Auburn University has developed software that facilitates the DEMA Core interview questions, stores the results in a database, and transforms the interview results to visualizations of the functional level, data vessel level, and data element level views. ICAMS is currently working to integrate the DEMA-LLM into the DEMA software to recommend reconciled interview results at each of the three steps of DEMA and to automatically generate insights and process improvements to improve the current process as captured by DEMA and generate models in SysML v1, with future research planned for integration with v2. For more information on the DEMA software, please reach out to the authors.

Funding Disclosure: This research is funded by the Interdisciplinary Center for Advanced Manufacturing Systems (ICAMS) through the Industrial Base Analysis and Sustainment Office of the Office of the Secretary of Defense.

REFERENCES

- Ledford, Allison B. 2023. "A Data Element Mapping and Analysis (DEMA) Approach for Implementing a Complete Digital Thread." PhD diss., Auburn University. <http://etd.auburn.edu/xmlui/handle/10415/8918>.
- Ledford, Allison B. 2024. "Data Element Mapping and Analysis (DEMA): A Systematic Method to Implement a Complete Digital Thread." Presentation at the MBE & QIF Summit.
- Ledford, Allison B., Gregory Harris, Susan Askew, and Gregory Purdy. 2025. "Application of Data Element Mapping and Analysis for System Definition to Enable Model-Based Systems Engineering." *Systems Engineering* 28 (1): 15-28. <https://doi.org/10.1002/sys.21776>.
- Ledford, Allison B., Gregory Harris, H. Patterson, and P. Farrington. 2024. *Data Element Mapping and Analysis (DEMA)*. US Patent 12,131,134, issued October 29, 2024.
- Ledford, Allison B., Gregory Harris, and Gregory Purdy. 2023. "Implementing a Complete Digital Thread: The Need for Data Element Mapping and Analysis." *IEEE Open Journal of Systems Engineering* 1: 139-52. <https://doi.org/10.1109/OJSE.2023.3325189>.
- O'Leary, Danny J., and Allison B. Ledford. 2025. "AI-Enhanced DEMA: Transforming Implicit System Knowledge into Intelligent, Compliant, and Documented Processes." Presentation at the AI4SE & SE4AI Research and Application Workshop, The George Washington University.

Kambhampaty et al. continued from page 36

- Topcu, Taylan G., Mohammed Husain, Max Ofsa, and Paul Wach. 2025. "Trust at Your Own Peril: A Mixed Methods Exploration of the Ability of Large Language Models to Generate Expert-Like Systems Engineering Artifacts and a Characterization of Failure Modes." *Systems Engineering* 28 (5): 583–604. DOI:10.1002/sys.21810.
- Wei, Jason, Zhiqing Sun, Spencer Papay, et al. 2025. "BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents." arXiv:2504.12516. Preprint, arXiv, April 16. DOI:10.48550/arXiv.2504.12516.
- Yang, An, Anfeng Li, Baosong Yang, et al. 2025. "Qwen3 Technical Report." arXiv:2505.09388. Preprint, arXiv, May 14. DOI:10.48550/arXiv.2505.09388.
- Zhang, Wenheng, Callum Cockburn, Michael Henshaw, et al. 2026. "MBSE Co-Pilot: A Research Roadmap." *Systems Engineering* 29 (1): 20–33. DOI:10.1002/sys.70011.
- Zhong, Shaohong, Andrea Scarinci, and Alice Cicirello. 2023. "Natural Language Processing for Systems Engineering: Automatic Generation of Systems Modelling Language Diagrams." *Knowledge-Based Systems* 259 (January): 110071. DOI:10.1016/j.knsys.2022.110071.

On (Human) Learning in Design Space Exploration and the Impact of AI Assistants

Daniel Selva, dselva@tamu.edu, Texas A&M University

Copyright ©2026 by Daniel Selva. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Design space exploration (DSE) is a foundational activity in early-phase systems engineering aimed at identifying promising designs through modeling, optimization, and analysis. With the rapid emergence of AI-powered assistants, engineers can now delegate increasingly complex DSE tasks, from generating candidate designs to analyzing entire tradespaces. While this delegation may improve efficiency and solution quality, it raises an important question: what is lost when engineers offload cognitive effort to AI in DSE? This article argues that DSE is not only an optimization process but also an information discovery and learning activity through which engineers develop intuition, uncover trade-offs, and build defensible rationales for design decisions. Drawing on some of our recent empirical studies, we examine how AI assistance affects both DSE performance and learning, highlighting a nuanced relationship between the two. The findings suggest that while AI can enhance DSE outcomes, excessive automation may hinder learning, particularly for less experienced engineers. We conclude with implications for designing AI assistants that balance performance gains with the need to preserve and enhance human understanding in engineering design.

INTRODUCTION

Design space exploration (DSE), also known as tradespace exploration (Ross and Hastings 2005), is a cornerstone of early-phase systems engineering. In DSE, engineers combine performance models, search and optimization algorithms, and visualization and data mining tools to iteratively explore candidate designs with the goal of identifying promising designs for further study. As design spaces grow in size and complexity, engineers increasingly rely on artificial intelligence (AI) to navigate them efficiently without succumbing to information overload. This raises an important question: are we delegating too much to the AI?

AI agents powered by foundation models and generative AI can now support engineers in DSE in a variety of

ways. They can create designs that meet certain requirements, answer questions about individual designs or whole design spaces, or run several DSE studies in the background under different assumptions and compare the results. Figure 1 shows a snapshot of an example AI agent called Daphne, which supports engineers in the design of distributed Earth observing satellite missions (Viros-i-Martin and Selva 2020; Apaza and Selva 2024). In that setting, the user defines a hierarchy of mission objectives and measurement requirements and a catalog of remote sensing instruments, and Daphne helps design a distributed satellite mission carrying some of those instruments in different orbits to maximize requirement satisfaction while minimizing lifecycle cost.

The DSE task in such systems is truly interactive and collaborative. The human user and the AI agent can see and build upon each other's designs, generate new designs manually or by running optimization, design of experiments, or generative AI, use sensitivity analysis and data mining to find patterns in certain regions of the design space, and retrieve information from databases of components or past designs. A large language model provides a natural language interface that makes this collaboration much smoother.

As engineers delegate more of the functions of DSE to AI assistants, the question arises whether this cognitive offloading may have undesirable side effects. We know from other domains that delegating cognitive tasks to automation can lead to a loss of



Figure 1. Snapshot of the Daphne AI assistant for designing constellations of Earth observing satellites

knowledge or skill—who can still navigate well without digital maps?

To answer this question, we first need a deeper understanding of DSE itself before examining how AI assistance may affect it. In the remainder of this article, we begin by framing DSE as a knowledge discovery process, where understanding why and when a design performs well is as important as identifying high-performing designs. We then interpret this process through the lens of learning and discuss how concepts from education can be used to measure learning in DSE. Next, we examine how engineers explore design spaces, focusing on how different search strategies influence both performance (in terms of design quality, quantity, and diversity) and learning. Building on this foundation, we review recent human-subject studies to shed light on how AI assistants shape DSE processes and outcomes. We conclude with practical implications for engineers developing or using AI-assisted DSE tools.

DSE IS NOT JUST ABOUT FINDING THE OPTIMAL DESIGN

The design automation community has long emphasized the goal of DSE as finding the optimal or most preferred designs. But DSE is also, and perhaps primarily, an information discovery task. As engineers explore design spaces, they build intuition about the structure of the design problem itself. They develop understanding of

relationships between design variables, objectives, and constraints, dominant trade-offs, recurring architectural patterns, and families of high-performing solutions.

This information can help engineers identify better designs faster, improving DSE performance in terms of the quantity, quality, and diversity of designs considered. But its value goes well beyond that. It enables engineers to identify shortcomings in models and refine the problem formulation. Optimization algorithms are good at exploiting weaknesses in our models and formulations, including missing physics, missing objectives, or missing constraints that allow unrealistic performance at no apparent cost. Learning during DSE can also help engineers recognize when the design space itself needs to be expanded to include additional decisions or alternatives. After all, what value is there in finding the best design in the wrong design space?

Equally important, DSE learning helps engineers justify design decisions to stakeholders and decision makers. It enables the development of defensible design rationales grounded in an understanding of trade-offs and system behavior. Over longer timescales, repeated DSE activities can also help engineers build domain knowledge and develop reusable heuristics and design guidelines that improve future design efforts (Fu, Yang, and Wood 2016). We call this process **DSE learning** because it goes beyond simple information discovery. Data mining can

identify patterns, but making sense of those patterns requires building explanations and mental models and then applying them to improve performance. That is learning.

Given its importance, DSE learning should arguably be treated as a first-class objective rather than a byproduct of finding good designs. Yet it has received limited attention in the systems engineering and design automation literature, beyond early work by Sim and Duffy (2004) and work from the SEAK Lab in the 2018–2023 timeframe, partially funded by the Systems Engineering Research Center. This article summarizes the main findings from those studies, which show mixed effects of AI assistants on DSE outcomes, including both performance and learning.

MEASURING DSE LEARNING

One reason DSE learning may have received relatively little attention is that it is difficult to define and quantify. It is hard to improve something that is not measured.

Bang and Selva (2020) proposed and compared different approaches to measuring DSE learning. The traditional way to measure learning in education is through tests. Test items can range from multiple-choice questions to complex problems and essays. The revised Bloom's taxonomy (Krathwohl 2002) is commonly used in education to categorize learning goals, progressing from remembering and understanding at lower levels to applying,

evaluating, and creating at higher levels. A common best practice is to assess multiple levels of this taxonomy rather than only factual recall.

Applying this idea to DSE, Bang and Selva proposed different types of test questions, including questions about designs, such as “Which of these two designs is more likely to be on the Pareto front?”, and questions about design features, such as “Which of these two features is more likely to be found among designs on the Pareto front?” Results from Bang and Selva’s 2020 study with 11 Ph.D.-level participants showed that most types of questions correlated reasonably well with one another and with participants’ self-assessment of learning, suggesting an approach for a principled and consistent way of measuring learning in DSE.

HOW DO ENGINEERS LEARN IN DSE?

Human learning is a complex topic with many complementary and competing theories. One useful way to think about learning in a structured setting like DSE is through the lens of the scientific method: iterative building and testing of hypotheses that refine engineers’ beliefs.

Under this view, engineers learn in DSE by iteratively forming and testing hypotheses about the design space. For example, observing similarities among non-dominated designs may lead a designer to hypothesize that certain features drive performance or cost. The engineer can then find and/or create designs with and without those features and test the hypothesis, leading to updated beliefs and new directions for exploration.

WHAT DRIVES SEARCH STRATEGIES IN DSE?

In practice, DSE learning is intertwined with other processes where search is driven by the engineer’s goals. Broadly speaking, DSE is mediated by a combination of performance goals and learning goals. Engineers switch between different goals and search strategies based on signals about progress, uncertainty, and their own cognitive state.

DSE search strategies also face a classical exploration-versus-exploitation trade-off. Basic exploratory strategies, such as random search or systematic search similar to fractional factorial enumeration, are often used early on. As patterns begin to emerge, more structured strategies are adopted and guided by more specific learning or performance goals. Toward the end of the search, the process typically shifts toward exploitation, focusing on the most promising regions found so far and using local search and accumulated knowledge to try to identify the best possible designs within those regions.

THE RELATION BETWEEN SEARCH STRATEGIES AND DSE OUTCOMES

Several studies show that how engineers explore matters as much as how much they explore. Search strategies clearly affect both performance and learning.

Song and Selva (2023a) found in an observational study with graduate students that evaluating more designs does not necessarily lead to more learning. In fact, students who followed a brute-force strategy based on evaluating many designs with large random moves in the design space—what the authors called *mindless creation*—learned less about the problem than those who evaluated fewer, more deliberate designs, extracted patterns from them, and then used those patterns to guide subsequent designs.

Interestingly, Song and Selva (2023b) found that forcing engineers to include periods of structured reflection during DSE increased interactions with the AI assistant but did not necessarily improve DSE learning. On the other hand, some simple interventions targeting engineers’ search strategies can be very effective. Chaudhari, Suresh Kumar, and Selva (2021) found that setting explicit DSE goals aimed at either improving learning or improving performance led to significant improvements in the targeted outcome, at the expense of the other. Similarly, Bang et al. (2022) showed that engineers who were forced to explore the feature space—a space of design patterns visualized in precision-recall coordinates—learned more than those who explored only the traditional design space. That increase in learning, however, came at a cost in performance.

THE COMPLEX RELATION BETWEEN DSE PERFORMANCE AND LEARNING

As engineers learn during DSE, they can often use that knowledge immediately to find better designs. This suggests a positive relationship between DSE performance and learning. Chaudhari and Selva (2023) did find such a correlation. Bang and Selva (2020) found that learning scores measured using the approach defined above correlate with the ability to create a dominant design feature (i.e., a combination of design variables common among good designs), but not necessarily with the ability to synthesize optimal designs from scratch. That is plausible: generating high-performing designs in complex systems requires understanding many interactions among design features, often beyond what a human can integrate unaided. If engineers could create optimal designs from scratch without models, simulations, search, and optimization, DSE would not be needed in the first place.

Several studies found a negative correla-

tion between performance and learning (Bang et al. 2022; Chaudhari, Suresh Kumar, and Selva 2021; Viros-i-Martin 2022). One plausible explanation is that, in all of these studies, the DSE tasks were fixed in duration and relatively short, often between 10 and 30 minutes. Under the hypothesis-building-and-testing view of DSE learning, a trade-off between performance and learning is not surprising in time-limited settings. Testing hypotheses often requires evaluating designs that are not likely to be optimal, simply to improve understanding. Those evaluations consume time and attention that could otherwise be spent exploiting promising regions of the design space.

This phenomenon resembles the classical exploration-versus-exploitation trade-off in global optimization and machine learning. But there is an important difference: DSE tasks do not exist in isolation. They are embedded in enterprises and in a continuing stream of design problems. Performance goals may explore in order to find promising regions faster. Learning goals may explore because engineers do not fully trust their models, because they need a convincing explanation for stakeholders, or because they want reusable knowledge for future tasks. Those motivations go beyond the narrow context of a single DSE run.

THE IMPACT OF AI ASSISTANTS

As AI became more capable, even before the appearance of large language models, it became clear that engineers would soon be teaming with AI agents for DSE. That immediately raised a natural question: does AI improve DSE outcomes?

An early study with nine JPL engineers found that using the AI assistant described in the introduction and shown in Figure 1 to explore a satellite constellation design space improved search performance. In fact, human-AI teams performed better than either humans or AI alone, likely due to a synergy between the powerful metaheuristics of the AI’s optimization algorithms and the expert judgment of the engineers guiding the search. However, DSE learning, as measured by a test, was lower when using the AI assistant (Viros-i-Martin and Selva 2020). In a different study using an AI assistant for a truss structure DSE task, Chaudhari and Selva (2023) found that increasing the level of automation improved DSE performance but reduced learning.

These results are consistent with cognitive delegation: as engineers rely more on the AI, they may engage less deeply with the problem. At the same time, AI agents also offer real opportunities to enhance learning. They can help engineers follow more structured search strategies, sup-

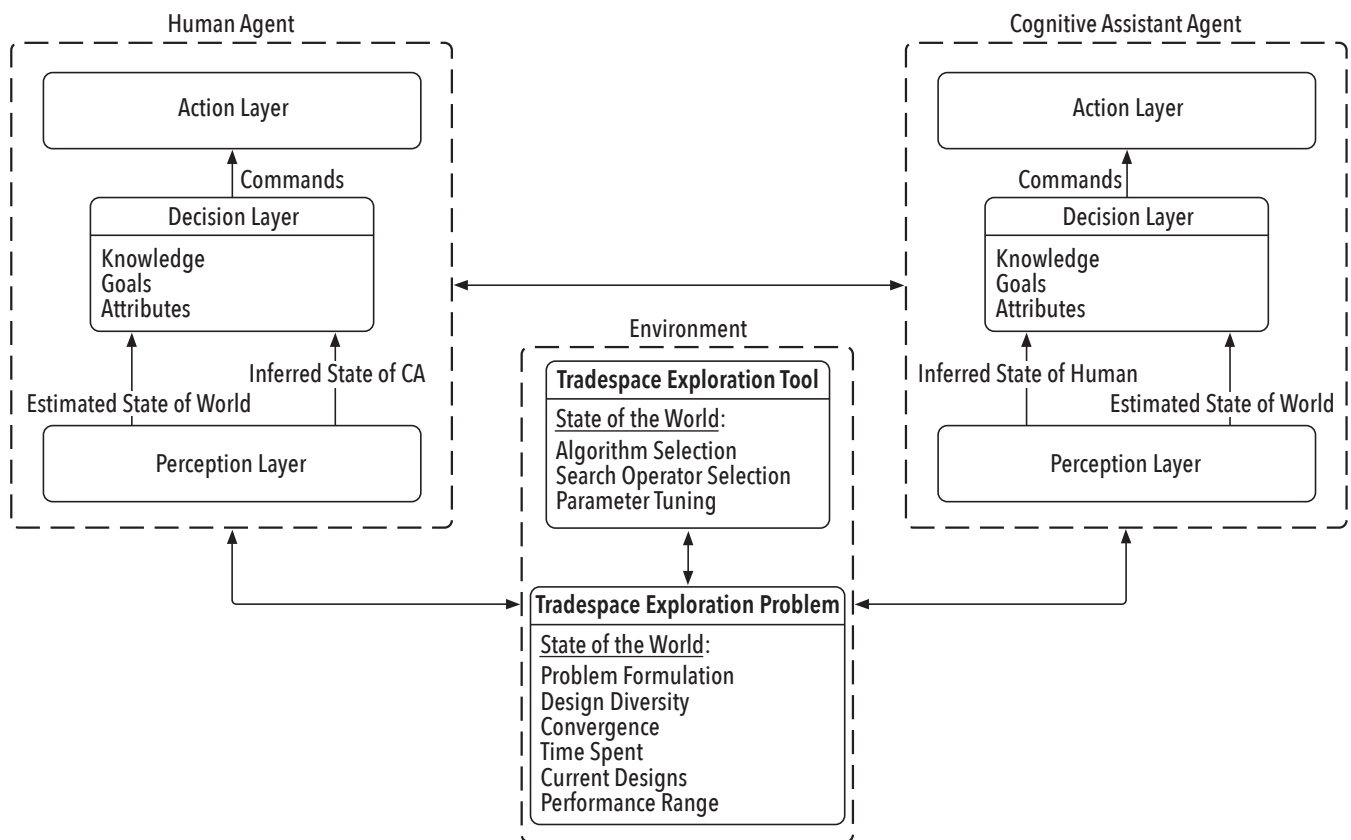


Figure 2. A framework for Human-AI collaborative DSE, adapted from (Viros-i-Martin and Selva, 2021). The engineer and the AI assistant are both agents that make decisions based on their perceptions, goals, state and knowledge.

port hypothesis generation and testing by identifying patterns among good designs, generate targeted candidate designs, quantify trade-offs, and retrieve relevant information across heterogeneous sources.

For example, Viros-i-Martin and Selva (2022a) found that in an AI-assisted satellite constellation DSE task, the more participants used the AI assistant, the more they learned about the problem. Unfortunately, efforts to identify the best roles and functions for the AI agent to play—that is, the best functional allocation in the human-AI collaboration—have shown that outcomes depend strongly on implementation details, human factors, and individual differences.

Engineers with different levels of expertise use different DSE strategies and have different needs from the AI. What works well for one group may not work well for another. In a small study with 22 subjects spanning different levels of expertise, Viros-i-Martin (2022) found that expertise strongly shapes how engineers interact with the AI assistant and how much they benefit from it. Novices appear to benefit from increased interaction with the assistant, including more explanations, more evaluated designs, and more hypothesis testing. Experts, in contrast, may need fewer explanations and fewer hypotheses to achieve

similar gains. A one-size-fits-all assistant is therefore unlikely to be optimal.

IMPROVING AI ASSISTANTS

To improve AI assistants for DSE, it is useful to build on what is known from the human-agent interaction literature. Figure 2, adapted from Viros-i-Martin and Selva (2021), illustrates a framework for human-AI collaborative DSE in which both the engineer and the AI assistant are modeled as agents jointly working on a DSE task using tools such as optimization and data mining algorithms. Both agents have goals and knowledge about their own state, the other agent, and the state of the world, in this case the DSE task itself.

Viewed through this lens, improved collaborative performance should result from better alignment between the engineer's and the AI agent's goals and mental models. There is some empirical support for this idea. Viros-i-Martin and Selva (2022b) found that adapting the AI assistant to the engineer's explicit DSE goals, especially to support hypothesis testing, improved learning scores, whereas a non-adaptive AI assistant did not. Interestingly, the adaptive AI assistant also reduced performance, reinforcing the idea that performance and learning can be in

tension during DSE. In Viros-i-Martin (2022), adapting to the engineer's level of expertise did not improve learning, but it did improve trust and satisfaction.

This brings up another important variable: trust. Trust plays a central role in human-AI collaboration because it directly shapes how engineers use AI assistants in practice. If engineers place too much trust in the assistant, they may accept its suggestions without sufficient scrutiny, reducing engagement with the problem and potentially limiting learning. If they place too little trust, they may ignore useful recommendations, underutilizing the capabilities of the AI. Both over-reliance and under-reliance can therefore degrade joint performance (Lee and See 2004).

In the context of DSE, trust is closely linked to the level of automation, the transparency of the assistant, and the engineer's own expertise. For example, highly automated systems may encourage over-reliance, while more interactive systems that expose reasoning or support hypothesis testing may promote more appropriate levels of trust and engagement. Designing AI assistants that calibrate trust appropriately—encouraging effective use without discouraging critical thinking—remains an important and largely open challenge.

IMPLICATIONS FOR PRACTITIONERS

The studies discussed here only scratch the surface of DSE learning and the impact of AI assistants. Most involved relatively small samples, and there were important differences between graduate students and expert practitioners. In addition, the DSE tasks were short, typically less than one hour, and most studies used AI tools developed before the recent wave of LLM-based systems. Larger and longer-term studies with expert practitioners using frontier AI models will be needed to improve our understanding of these issues.

That said, several practical lessons already stand out:

1. AI agents can improve DSE performance and can sometimes

improve learning, but they must be carefully designed and implemented. Excessive delegation to the AI without active countermeasures may reduce DSE learning, especially for less experienced engineers.

2. **Higher levels of automation are not always better.** While engineers may ask for more automation from AI agents, more automation can reduce engagement with the problem and therefore reduce learning.
3. **Adaptive AI agents are promising, especially when they adapt to the engineer's goals and attributes such as level of expertise.** However, such adaptation is difficult to implement well because of substantial individual differences.

CONCLUSION

As we move toward an era of superhuman AI performance in many tasks, we face a real conundrum. On one hand, improving DSE learning may become more feasible than ever with AI agents that have very large context windows, strong reasoning and planning abilities, and the ability to control DSE tools directly. On the other hand, one might ask what the point of improving human learning is if fully autonomous DSE eventually becomes possible.

In practice, DSE is likely to remain fundamentally human-centered for the foreseeable future. Ultimately, DSE is not just about optimization. It is also, and perhaps primarily, about helping customers, managers, and decision makers believe that selected designs are understood, justified, and robust. ■

REFERENCES

- Apaza, G., and D. Selva. 2024. "Leveraging Large Language Models for Tradespace Exploration." *Journal of Spacecraft and Rockets* 61 (5): 1165–1183. <https://doi.org/10.2514/1.A35834>.
- Bang, H., and D. Selva. 2020. "Measuring Human Learning in Design Space Exploration to Assess Effectiveness of Knowledge Discovery Tools." In *Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2020)*. <https://doi.org/10.1115/DETC2020-22631>.
- Bang, H., Y. L. Z. Shi, G. Hoffman, S.-Y. Yoon, and D. Selva. 2022. "Exploring the Feature Space to Aid Learning in Design Space Exploration." In *Design Computing and Cognition '20*, 529–546. Springer. https://doi.org/10.1007/978-3-030-05363-5_11.
- Chaudhari, A. M., R. Suresh Kumar, and D. Selva. 2021. "Supporting Designer Learning and Performance in Design Space Exploration: A Goal-Setting Approach." In *Proceedings of the ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2021)*. <https://doi.org/10.1115/DETC2021-71257>.
- Chaudhari, A. M., and D. Selva. 2023. "Evaluating Designer Learning and Performance in Interactive Deep Generative Design." *Journal of Mechanical Design* 145 (5): 051403. <https://doi.org/10.1115/1.4056374>.
- Fu, Katherine K., Maria C. Yang, and Kristin L. Wood. 2016. "Design Principles: Literature Review, Analysis, and Future Directions." *Journal of Mechanical Design* 138 (10): 101103.
- Lee, John D., and Katrina A. See. 2004. "Trust in Automation: Designing for Appropriate Reliance." *Human Factors* 46 (1): 50–80.
- Ross, A. M., and D. E. Hastings. 2005. "The Tradespace Exploration Paradigm." In *Proceedings of the International Council on Systems Engineering Conference 2005*, 15:1706–1718.
- Sim, Siang Kok, and Alex H. B. Duffy. 2004. "Evolving a Model of Learning in Design." *Research in Engineering Design* 15 (1): 40–61.
- Song, H., A. Viros-i-Martin, and D. Selva. 2023a. "Understanding the Relation Between Designer Search Strategies and Designer Learning During Design Space Exploration." In *Proceedings of the ASME 2023 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2023)*.
- Song, H., A. Viros-i-Martin, and D. Selva. 2023b. "Effect of Reflection and Incubation on Artificial-Intelligence-Assisted Design Space Exploration." In *Proceedings of the ASME 2023 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2023)*.
- Viros-i-Martin, A., and D. Selva. 2020. "Daphne: A Virtual Assistant for Designing Earth Observation Distributed Spacecraft Missions." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13: 30–48. <https://doi.org/10.1109/JSTARS.2019.2948921>.
- Viros-i-Martin, A., and D. Selva. 2021. "A Framework to Study Human-AI Collaborative Design Space Exploration." In *Proceedings of the ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2021)*. <https://doi.org/10.1115/DETC2021-67619>.
- Viros-i-Martin, A. 2022. *A Cognitive Assistant for Human-AI Collaborative Early Mission Formulation*. PhD dissertation.
- Viros-i-Martin, A., and D. Selva. 2022a. "Learning Comes from Experience: Effects on Human Learning and Performance of a Virtual Assistant for Design Space Exploration." In *Design Computing and Cognition '20*, 547–564. Springer. https://doi.org/10.1007/978-3-030-90625-2_39.
- Viros-i-Martin, A., and D. Selva. 2022b. "Improving Designer Learning in Design Space Exploration by Adapting to the Designer's Learning Goals." In *Proceedings of the ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2022)*.

STPD – Operational Design as a Model-Based Extension of STPA

Avi Harel, Ergolight

Copyright ©2026 by Avi Harel. Permission granted to INCOSE to publish and use.

■ ABSTRACT

The paper argues that the growing complexity of distributed socio-technical systems exposes a fundamental limitation of the STPA methodology. Although STPA transformed safety engineering (Leveson 2011; Leveson 2004) by treating safety as a control and constraint problem rather than purely a reliability problem, it still assumes that hazardous scenarios and the constraints needed to prevent them can largely be identified during design time. The paper claims that this assumption no longer holds in modern adaptive systems (Rasmussen 1997; Woods 2015), where operational contexts evolve continuously, AI-based components change behavior dynamically, distributed actors develop inconsistent interpretations of the operational situation, and unexpected interactions emerge during runtime.

According to the paper, safety in such environments cannot rely only on predefined constraints. Instead, systems must continuously validate during runtime whether operational assumptions, coordination structures, and safety constraints still remain valid under changing conditions. This shifts the engineering challenge from specifying static safe behavior toward maintaining operational coherence under uncertainty. To address this challenge, the paper proposes extending STPA into a broader framework called system-theoretic process design, (STPD). STPD treats runtime surprise (Woods 2015; Dekker 2011) as an inherent property of complex adaptive systems (Rasmussen 1997; Woods 2015) rather than as an exceptional case outside the model. Under this approach, systems must be capable of detecting deviations from expected operational assumptions, adapting coordination structures, managing degraded modes, recovering from anomalies, and learning from unforeseen situations while still constraining behavior within acceptable operational limits.

The paper emphasizes that this extension substantially increases analytical complexity. The analysis must address not only fixed control loops and predefined unsafe control actions, but also adaptive behaviors, runtime transitions, resilience mechanisms (Hollnagel, Woods, and Leveson 2006; Woods 2015), evolving operational contexts, and partially unknown future states. Resilience itself introduces additional coupling,

variability, coordination overhead, and verification difficulty. Consequently, a central tradeoff emerges: systems must remain adaptive enough to survive operational surprise while simultaneously remaining constrained enough to preserve safety, controllability, productivity, and usability.

A central criticism presented in the paper concerns the scalability limitations of classical STPA. The number of possible socio-technical operational states grows exponentially with the number of system variables, making exhaustive analysis infeasible. The paper also argues that STPA's constraint-based logic requires engineers to identify all hazardous control actions explicitly, whereas the original STAMP philosophy focused more broadly on constraining the system toward correct operation according to an intended operational plan. Based on this critique, the paper proposes combining STPA with the original STAMP concept of operational guidance. The proposed solution proactively defines operational scenarios during design time and enforces corresponding system states during runtime transitions. The system continuously verifies whether ongoing operations remain aligned with the intended coordination structure, detects deviations, and activates emergency or recovery mechanisms when inconsistencies appear. To implement this idea, the paper introduces a generic operational model based on a network of interactions. Each interaction is defined through protocols between a controller and multiple services. Operational constraints are represented as scenarios consisting of an external task and the coordinated internal states of participating components. Participants may initiate additional interactions, thereby forming a dynamic interaction network. The model includes both proactive coordination mechanisms and reactive anomaly detection and response mechanisms.

Finally, the paper argues that this modular interaction-based structure improves the practical application of STPA by associating control states with specific interaction participants. According to the paper, this association increases traceability and helps achieve broader analytical coverage across all stages of the STPA process.

■ **Keywords:** STAMP, STPA, STPD, Scenario-Centered Operation, Operational Design, Operational network, Canonical Scenario Representation, Controller Multi-Service Interaction, Exceptions, Decision Support, Feedback, Feedforward, Proactive Troubleshooting, Model-Based Systems Engineering, System State Consistency, Human Factors, HSI, Socio-Technical Systems, Digital Twin, LLM

INTRODUCTION AND CORE METAMORPHOSIS

The paper introduces a transition from the STPA approach to the system-theoretic process design (STPD) approach, where the hierarchy of constraints relies on a hierarchy of components within the system's operational model. In this approach, runtime surprises (Woods 2015; Dekker, 2011) are not perceived as external anomalies but rather as an inevitable part of complex systems. Within this modeling framework, systems are required to detect deviations from operational assumptions, adapt coordination and control mechanisms, manage degraded operational modes, and learn from unexpected events—all while maintaining safe constraint boundaries.

The article addresses the core dilemma of operational flexibility, which is necessary to cope with surprises, versus protection against operational threats and loss of control in anomalous situations. The extension to STPD significantly increases analysis complexity. Instead of dealing solely with predefined hazardous control actions, it must also contend with changing contexts, runtime state transitions, adaptive behavior, and resilience mechanisms (Hollnagel, Woods, and Leveson 2006; Woods 2015). Furthermore, the very addition of resilience manifests as complexity that induces coordination overhead, behavioral variance, and tighter coupling between components.

Additionally, the paper analyzes the theoretical boundaries of STPA in preventing unexpected failures (Safety I). These limits stem from the exponential growth in the number of possible system states and the practical difficulty of enumerating all unsafe control actions (UCAs). This stands in contrast to the original STAMP approach, which suggested prioritizing proactive constraint of the system toward proper operation according to an operational plan (Safety II (Hollnagel 2014)). The paper proposes to bridge STAMP and STPA through the proactive design of scenario-compatible system states and enforcing these states at runtime.

STPD extends STPA by integrating explicit mechanisms for runtime scenario coordination, rather than relying solely on the static design of the control structure. Within this framework, a canonical scenario representation (CSR) is defined, enabling a uniform, formal, and consistent description of operational modes and the transitions between them. Moreover, STPD

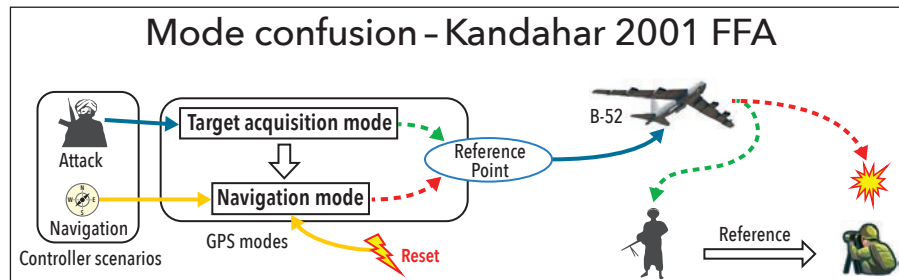


Figure 1. Mode confusion in the Kandahar accident

focuses on enforcing scenario consistency across distributed controllers (Leveson 2011; Hutchins 1995) and components, ensuring that the entire system operates according to a shared operational understanding, even under conditions of uncertainty and dynamic changes.

The methodology also includes state transition control mechanisms that are aware of the scenario's coordination context, as well as capabilities for runtime verification of operational coherence among distributed controllers (Leveson 2011; Hutchins 1995). This is intended to mitigate deviations, miscoordination, and degradation into hazardous states. Additionally, the system is continuously required to verify that operations align with coordination definitions, detect deviations, and activate emergency, recovery, and troubleshooting mechanisms.

To implement this approach, a generic system operation model is proposed via a directed network representation, where nodes represent interactions between controllers and services, and the connecting lines represent scenarios. An interaction is defined through dedicated protocols, and scenarios are generated by combining external tasks received from the higher echelon, system states, and context variables. The operational model allows for both automatic discovery and reactive response to deviations, as well as the proactive initiation of diagnostic and recovery processes. Each system component can trigger additional interactions, thereby forming a dynamic network of controlled interactions. This generic model allows organizing the STPA control structure according to the models, thereby ensuring that the STPA safety constraints analysis addresses all conditions in normal operation.

OPERATIONAL BACKGROUND AND SYSTEM CHALLENGES

The paper focuses on the central challenge of operational design in complex

socio-technical systems—namely, how to guarantee both performance/throughput and safety/coordination among system components at runtime. The starting point is the shift from the 'blame the operator' paradigm to a systemic approach that views failure as a result of design flaws and structural mismatches between system components.

A prime historical example is the 2001 Kandahar incident (Mode confusion – Kandahar 2001 FFA), where a battery replacement caused a GPS reset. Consequently, its operational mode defaulted back to navigation mode, and the waypoint shifted from the Taliban target's location to the coordinates of the GPS unit itself, as illustrated in Figure 1.

The failure stemmed from the design allowing a state where the GPS operational mode was mismatched with the active operational scenario. In such cases, the risk did not arise from a classic hardware or software malfunction, but rather from an inconsistent interpretation of the scenario across different control layers. The challenge lies in the fact that multiple independent units must collectively produce consistent behavior, even though their information regarding the state and intentions of neighboring units is partial and occasionally outdated. More examples are listed in the Appendix.

In alignment with the STAMP approach, safety is not achieved merely by reacting to failure triggers, but by constraining system operations to remain within proper, controlled boundaries. In this context, the STPA method is introduced, allowing the identification of unsafe control actions and the derivation of safety constraints early in the design phase. STPA provides support for real-time coordination, synchronization, and constraint enforcement in distributed, dynamic systems. The problem becomes particularly acute when different

subsystems hold differing internal models of the system state—meaning an incompatibility exists between the ‘process models (Leveson 2011)’ of various controllers and the actual reality. This condition can lead to operational errors, coordination breakdown, and systemic failure even when each individual component operates ‘correctly’ from its local perspective. Although STAMP/STPA address process models, feedback, and loss of control, the engineering realization of runtime coordination among distributed controllers (Leveson 2011; Hutchins 1995) remains an open challenge.

The paper emphasizes that the complexity of STPA scales rapidly with system complexity. As the number of control actions, operational modes, and interactions increases, an immense number of potential unsafe control actions (UCAs) is generated. This situation complicates generalization, coverage, and prioritization, often making the analysis overly expensive and cumbersome. To address this, a model-driven STPA approach is introduced, shifting the analysis focus from individual control actions to operational states and deviations from operational assumptions. Instead of manually analyzing every possible action, the system examines which states deviate from the normal operational model, thereby improving generalization capabilities and reducing the risk of missing hazardous conditions.

Furthermore, the paper proposes a ‘lean’ version of STPA for systems where risk is not catastrophic but stems primarily from interaction and coordination issues—such as software systems, smart home devices, or office services. This streamlined approach focuses primarily on state transitions, synchronization logic, and user-service interactions to rapidly identify coordination gaps, timing errors, and conflicting state assumptions.

The paper also draws an important distinction between prevention strategies and protection strategies. Prevention aims to avoid the creation of hazardous states altogether through safety constraints, state control, input validation, and resilient design. Conversely, protection assumes that coordination failures and unforeseen conditions will inevitably occur, and thus includes mechanisms for detection, alerting, transitioning to a safe state, controlled degradation, and recovery. Resilient systems must combine both approaches: minimizing failure probability while effectively handling coordination loss when it occurs in practice.

STPD AS AN EXTENSION OF STPA

This section presents STPD as a systemic

extension of STAMP/STPA for distributed socio-technical systems, where safety and efficiency depend not just on component integrity, but primarily on maintaining consistent coordination among the state interpretations of controllers, services, and human operators. The core argument is that many systemic risks do not stem from a single component failure, but rather from a deviation from the operational coherence envelope—manifesting as a state where different components act under mismatched understandings of the operational scenario. In complex systems, partial information, delayed feedback, poor synchronization, or normal performance variance can cause the system to continue operating ‘correctly’ at a local level while its global behavior becomes highly dangerous. Therefore, the central challenge is not just preventing known failures but identifying emerging loss of control and the collapse of the assumptions supporting the original control scheme at runtime. The contribution of STPD is not in replacing STPA’s control mechanisms, but in adding an operational coordination layer that manages scenario consistency, transition synchronization, interpretation alignment, and runtime constraint enforcement.

The text emphasizes that resilient systems must be capable of detecting even unpredicted situations, such as errors in specification, design, and component implementation. This requires an explicit representation of normal operations, including operational plans, constraints, process models (Leveson 2011), coordination rules, and assumptions regarding resources and expected behavior. The system cannot merely rely on detecting component faults; it must monitor the integrity of the operational coherence itself, identifying deviations from expected task structures, coordination issues among distributed process models, breaches of coordination protocols, or gaps between expected and actual behavior. The proposed approach views anomalous situations as semantic rather than merely physical phenomena, since the earliest indication of a problem may be a contradictory interpretation or a loss of shared situational awareness. Therefore, the system must integrate structural monitoring, semantic consistency checks, uncertainty estimation, and the capability to transition to safe backup states when the active scenario can no longer be determined with confidence.

A central concept introduced is the unsafe operational state (UOS), where controllers or services no longer agree on the mission state or operational mode. In such a state, even commands normally deemed safe can become hazardous

because inter-unit coordination is lost. From this arises the assertion that safety in distributed systems depends on consistency across the controllers’ process models (Leveson 2011). The chapter cites several classic examples, such as the Therac-25 accidents, Three Mile Island (TMI), and friendly-fire incidents, to illustrate how critical failures are generated by a gap between the internal representation of the system state and physical reality. In these cases, it was not necessary to isolate a specific software, hardware, or operator fault; instead, the failure cause is attributed to different subsystems operating under conflicting assumptions regarding the operational state. Accordingly, the paper interprets many accidents as violations of systemic consistency constraints rather than isolated local failures.

Based on this, the chapter proposes a transition from an exhaustive failure-cause analysis model to a scenario-constrained situational control (SCSC) model. Instead of trying to analyze all possible system states—an intractable approach in complex systems due to ‘state explosion’—systems engineers are required to define a limited number of operational scenario classes specified in operational documentation. Each scenario represents a group of states with shared operational meaning and constraints, along with permitted transitions and backup rules. At runtime, the active scenario is represented via the CSR, which serves as a shared reference base for interactions between the controller and services. This significantly reduces coordination complexity because the system is not required to analyze risk across all theoretical states, but only to operate within a well-defined scenario space. When the system encounters an uncertain situation that cannot be confidently mapped to one of the scenarios, it must transition to a safe mode scenario where only conservative, safe actions are permitted.

The chapter also defines ‘scenario confusion’, a condition where different components within the system operate according to different scenarios. Even though each component acts ‘correctly’ locally, the discrepancy between them creates a dangerous systemic failure.

Consider a system with two elements: a controller and a service, enabling them to perform two functions, depending on the service mode. The service provided should comply with the controller scenario. This defines normal situation. The operation is successful if the service complies with the operational scenario and is failure otherwise. Figure 2 illustrates this condition, in which the controller and service operate under inconsistent scenarios.

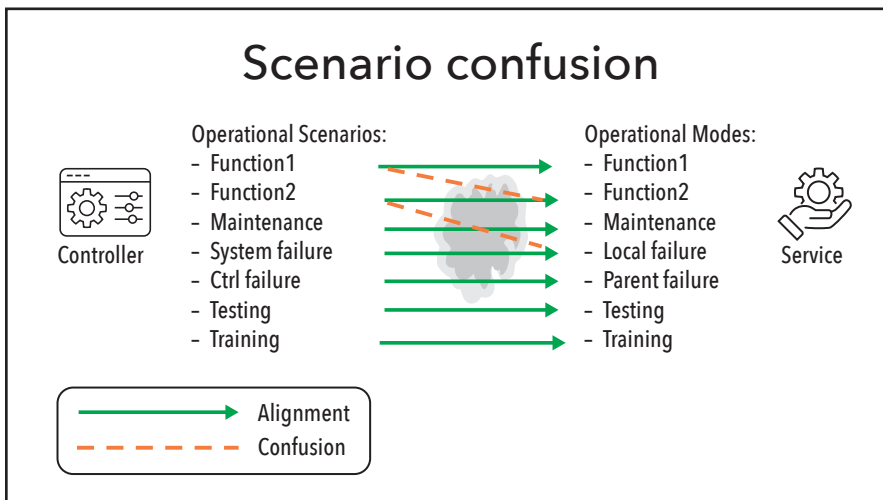


Figure 2. Scenario confusion

The figure illustrates a way to enforce scenario-mode compliance (green arrows) and examples of scenario confusion, when the modes divert from those induced by the scenarios (broken red lines).

The model shows enforcement mechanisms for scenario-mode alignment alongside examples of scenario confusion where operational modes deviate from the scenarios that should dictate them. In many historical instances, the scenario was never explicitly defined, leaving the control dilemma unresolved. In *Human Factors Engineering* (Reason 1990; Dekker 2014) (HFE), this is termed a ‘mode error (Sarter and Woods 1995; Norman 1981)’. The underlying, flawed assumption was that the operator must ensure the operational mode matches the scenario. In reality, this assumption proved incorrect, and the system was incapable of self-protection. From a systems engineering perspective, this constitutes a coordination failure between the controller and the service. The argument is that most famous ‘mode errors’ in HFE are actually systemic coordination failures resulting from the absence of an explicit representation of the active

scenario. Hence, the scenario must become an explicit system variable whose consistency can be verified at runtime. Figure 3 illustrates how scenarios are defined and how they impact the design.

The scenario serves as a shared reference point: it represents the mission as defined by the system owners and is used both for enforcing coordination and for identifying unexpected behavior when scenario consistency is compromised. Different components may hold a partial or delayed view of the state, but their behavior is constrained by the scenario available to them at execution time. Thus, coordination is achieved through consistency of interpretation rather than full system-state synchronization. Within this framework, the scenario functions as a control variable. Maintaining consistency of interpretation becomes an explicit control objective, and deviations from scenario boundaries become detectable states. In this manner, scenarios bridge system design and actual execution, supporting coordination, consistency, and cognitive load management in dynamic environments. The proposed architecture enforces scenario consistency as a system invariant, such that control actions

are permitted only when valid within the active scenario and coordinated across all components. When a mismatch is discovered, normal operations are suspended until synchronization is restored.

Furthermore, the challenges of transitions between scenarios are addressed. Transitions are considered a critical vulnerability point because subsystems may temporarily operate under differing states or assumptions during the switch. The Therac-25 accidents are presented as an example where rapid state transitions without synchronization created a time window in which the software state, hardware state, and operator intent were severely misaligned. Therefore, safety depends not just on stability within a state, but on the ability to rapidly regain coordinated control during a transition. The paper underscores the need for risk mitigation during synchronization, utilizing monitoring mechanisms for risk indicators such as contradictory states, timeouts, or degrading sensor trust, alongside anomaly response mechanisms that encompass alerts, troubleshooting, resilience (Hollnagel, Woods, and Leveson 2006; Woods 2015), and recovery. Diagnostics and troubleshooting are viewed here as proactive systemic activities focused on identifying coordination degradation before the system manifests hazardous behavior. Ultimately, while STPA focuses on identifying unsafe control actions within control hierarchies, STPD adds a new layer of distributed operational coordination and runtime coherence management, moving the needle from static control constraints to engineering operational resilience under uncertainty and surprise.

OPERATIONAL MODELING FRAMEWORK

This section details the framework for modeling operations in distributed socio-technical systems based on scenario coordination rather than exhaustive state analysis. The framework establishes a conceptual hierarchy separating the physical system state, local controller process models (Leveson 2011), operational states, scenarios, and coordinated runtime control. At the core of the architecture lies the canonical scenario representation (CSR), serving as a shared reference point for all controllers and services. The CSR does not describe the complete system state but only the information necessary for coordination, synchronization, and enforcing operational constraints.

The operational plan is defined as a directed network of tasks, constraints, and dependencies that restricts the permitted behavior of the system. The system state represents the objective, full state, whereas the controllers’ process models (Leveson

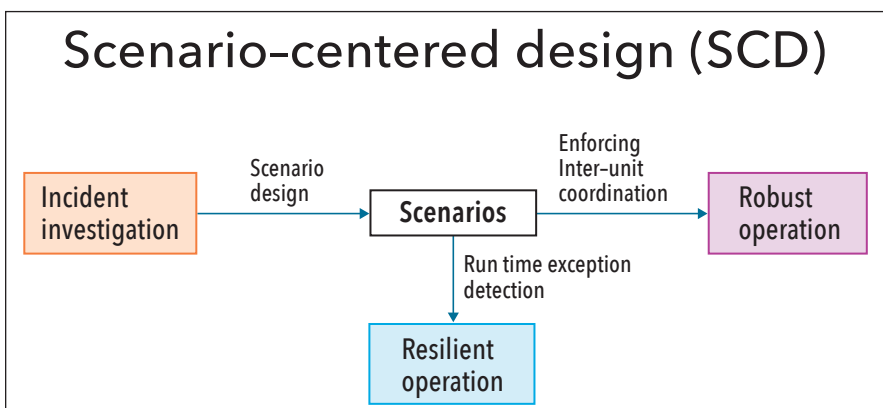


Figure 3. Scenario-centered design

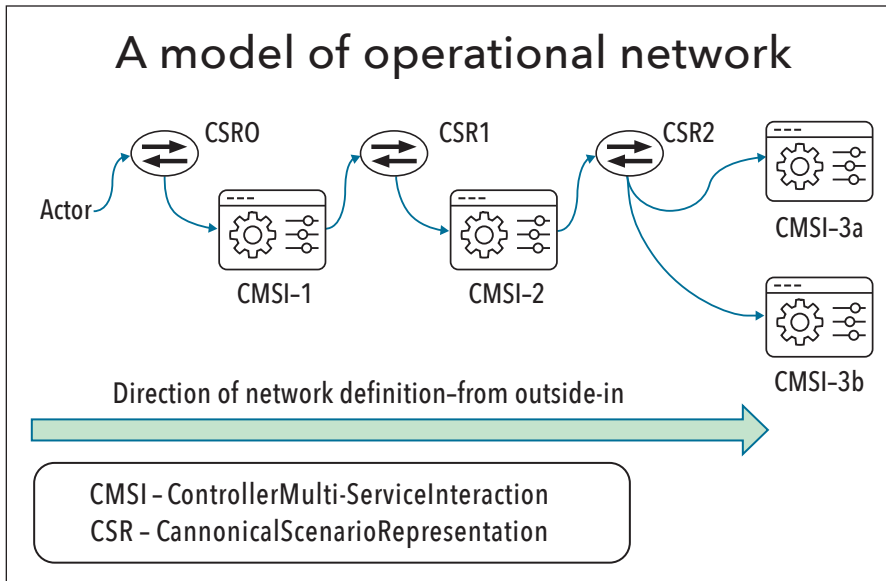


Figure 4. A model of operational network

2011) are based on partial info and may thus be inconsistent. An operational state is a mission-relevant interpretation of the system state, and a scenario is a bounded operational context defining which actions and coordinations are allowed. The CSR acts as a systemic coordination anchor enabling controllers and services to maintain a shared interpretation of the operational state.

The operational model describes system activity as a network of interactions (controller multi-service interaction—CMSI) in each of which a single controller operates in partnership with several services, based on a shared CSR. Coordination is performed by propagating the CSR among components, synchronizing scenario transitions, and enforcing coordination

constraints at runtime.

CSR control specifies how scenarios are established, updated, and propagated in the CMSI network, governing the way contextual information is maintained and shared. Its purpose is to ensure that transitions between scenarios remain coherent, that dependent interactions are provided with updated context, and that coordination is preserved across interaction boundaries. According to Boy and Narkevicius (2014) operational design should proceed from outside in. Figure 4 describes a model of operational networks, illustrating how an external actor triggers a sequence of interactions through passing canonical scenario representations.

This figure illustrates a simple network with four controller multi-service

interactions, in which an external actor activates CMSI-1 by a canonical scenario representation CSR0, which activates CMSI-2 by CSR1, which activates CMSI-3a and 3b by CSR2. The CSR operates as a control layer over the CMSI network. Scenario control enables the CSR to function as a system-level reference rather than a local variable.

System operation may be modeled as a network of interactions, in form of a directed graph composed of CMSIs, where nodes correspond to interaction contexts implemented as CSR, and edges represent activation and dependency relationships between them. This structure provides a way to describe how coordination flows and propagates throughout the system. Figure 5 illustrates that scenarios should enable coordinating the supplier with the client task (on the left) and should enable coordinating the scenarios with the controller (on the right).

This figure shows a way for transition from scenarios to CSR, and the way it controls the interactions. The owner of the interaction defines the scenario based on the task and the context and delivers it to the controller. Based on the active scenario of the interaction (active, available, functional ...) the controller defines the CSR, a system variable used as an anchor for the coordination and sets the states of the services according to assignment rules.

Scenario control enables the CSR to function as a system-level reference point rather than just a local variable. The article highlights that safety failures in distributed systems frequently originate from inconsistent operational interpretations (scenario confusion), leading locally correct actions to produce dangerous global behavior.

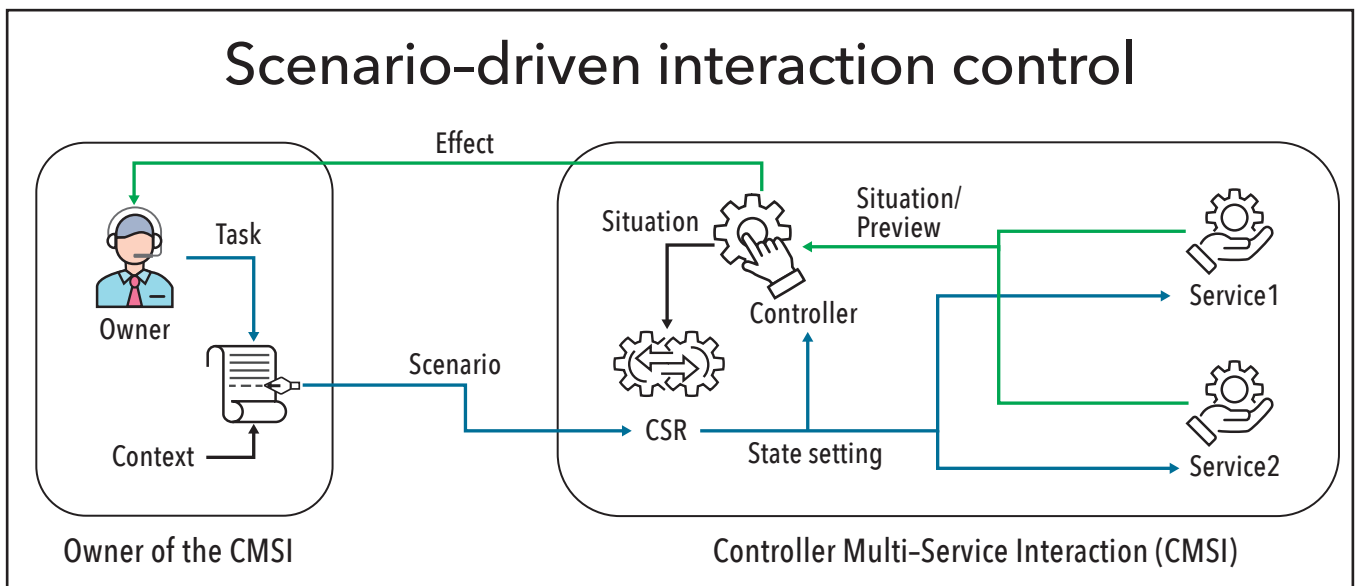


Figure 5. Scenario-driven operation

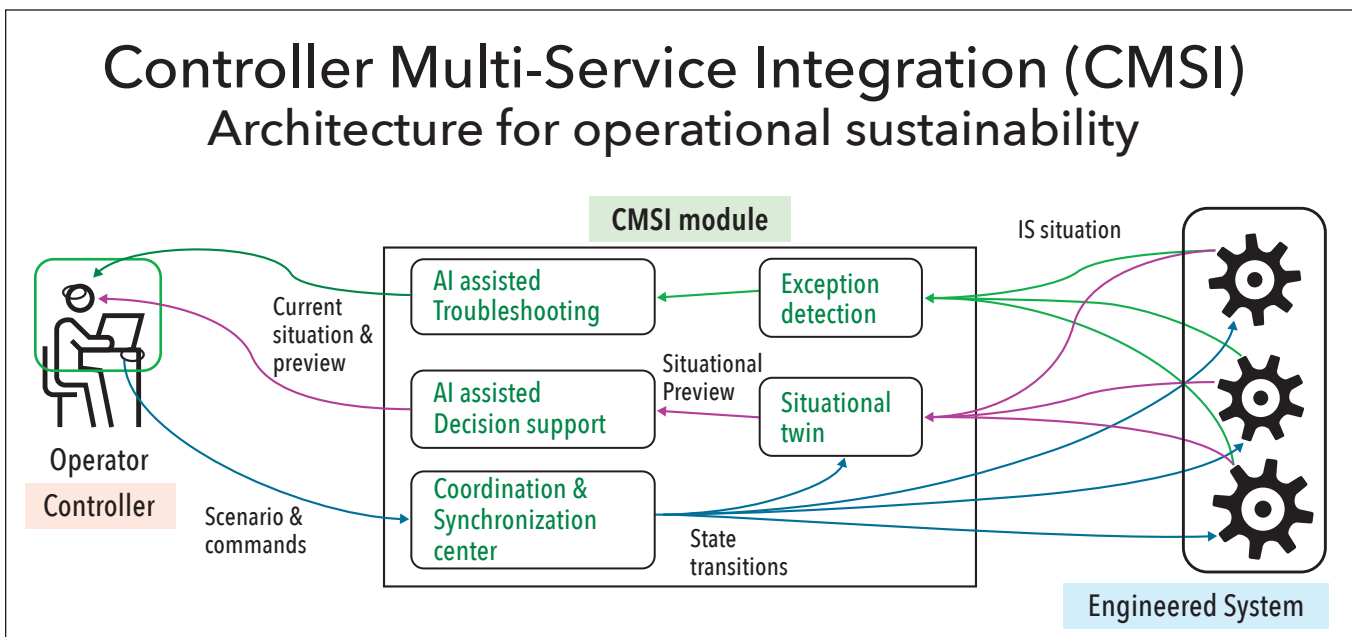


Figure 6. CMSI operation

Consequently, the safety objective is defined as the continuous preservation of scenario consistency and inter-component coordination.

The CSR is also described as a structural invariant. Instead of relying on reactive arbitration between components, transitioning to a specific scenario pre-allocates permitted and validated states to all participating services. This eliminates dangerous intermediate states and establishes a 'structural normalcy' where many hazards are logically removed from the possible state space. The operational framework is driven by scenario-based interactions where the owner defines the scenario based on the task and context, passing it to the controller, which sets the CSR and assigns service modes accordingly.

The model focuses particularly on CSR transitions and synchronization. During a scenario transition, the system is expected to temporarily enter a safe mode, propagate transition commands to all components, await synchronization confirmations, and correct timing anomalies. Normal operation during transitions is prohibited because asynchronous state transitions can generate unpredictable behavior.

The paper introduces an extension of traditional STPA control loops through the CMSI structure. Instead of a single controller acting against a single process, the CMSI coordinates multiple services under a shared scenario. Additionally, the model introduces a feedforward layer where the controller evaluates future operational alternatives using previews, workload assessments, and service simulations. This allows for the early identification of actions

that could create inconsistencies or violate scenario constraints.

Figure 6 illustrates how a CMSI module can provide feed-forward information for optimizing the controller decision.

The figure illustrates a hypothetical CMSI module that may be used to coordinate and synchronize the controller and the services. Employing agents (implemented by situational twins) this module can provide the controller with information required for proactive troubleshooting and reacting. The framework does not prescribe specific prediction methods. Its effectiveness depends on model accuracy and computational constraints.

A key element involves handling unpredicted exceptions. The core claim is that it is impossible to map all potential UCAs in complex systems in advance. Therefore, instead of focusing on preventing each hazardous action in isolation, the system must ensure it remains within a normal, coordinated operational envelope. Exceptions are identified by comparing actual behavior against the behavior expected under the active scenario. Deviations may appear as synchronization faults, illegal task transitions, resource anomalies, timing inconsistencies, or loss of coherence between subsystems.

Many failures occur because the system detects anomalous conditions too late, provides insufficient recovery support, or continues operating after coordination has begun to degrade. A generic model of exception management outlines how an exception evolves through sequential operational states (Normal operation → Exceptional situation → Unexpected operation

→ Failure), identifying the critical points where coordinated handling is required to prevent escalation.

Even with structural enforcement mechanisms, not all inconsistencies can be prevented. Systems operating under uncertainty, incomplete information, or degraded conditions may still deviate from expected behavior.

Exception management concerns the way deviations from expected behavior are handled. Within this context, two complementary modes of operation can be distinguished. The first is situation control, which is continuous in nature. Here, operation progresses through ongoing regulation relative to the defined scenario, with adjustments made as needed to maintain consistency. The second is exception management in the narrower sense, which is event-based. In this mode, attention is focused on deviations that exceed predefined thresholds and therefore require explicit corrective action.

Many failures occur because the system detects abnormal conditions too late, provides insufficient recovery support, or continues operating after coordination has already degraded. Figure 6 illustrates a conceptual model of exception management.

Figure 7 highlights how an exception evolves through successive operational conditions and identifies the points at which coordinated handling is required to prevent escalation. Following an initiating event, the system may transition from normal operation to an exceptional state. If the exception is not properly handled, the system may further degrade into unexpected operation and ultimately failure.

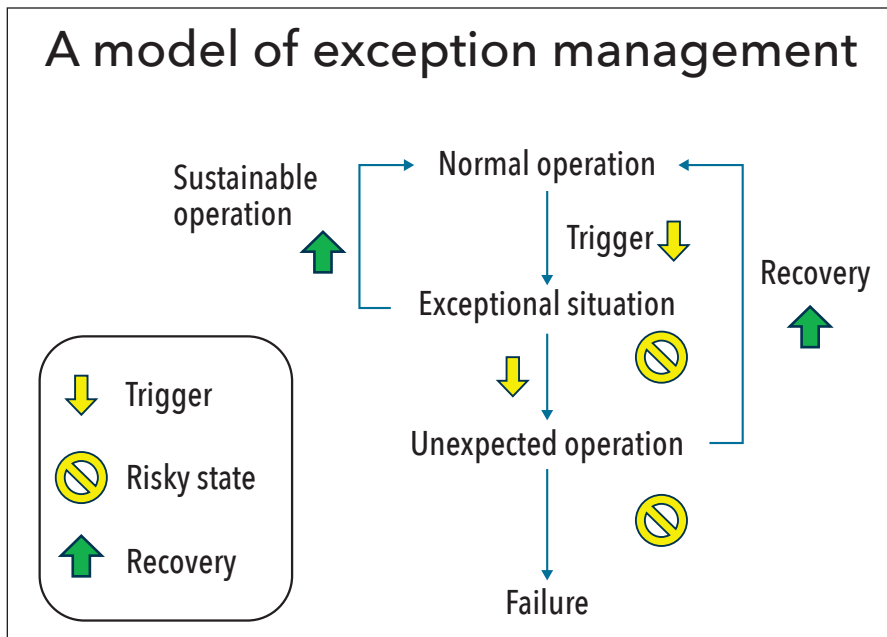


Figure 7. Exception handling

The model of normal operation allows for the detection of even unforeseen exceptions by defining what constraints must be maintained; any violation is interpreted as a deviation, shifting the safety focus to run-time validation of operational consistency.

OPERATIONAL ENGINEERING AND DESIGN METHODOLOGY

The engineering framework focuses on enforcing operational constraints through scenarios, distributed coordination, and transition control. Instead of looking at isolated components and isolated failures, the framework centers on preserving total operational consistency and preventing unsafe conditions before they emerge. The proposed framework may be implemented through a progressive top-down design process that begins with externally observable operational behavior and gradually refines the internal control architecture. The sequence reflects the principle that operational meaning should drive system structure, rather than emerging as a byproduct of component integration. Figure 8 shows a plan with stages of the operational design.

We start with a definition of the system functions and features used to get these functions, then define the scenarios that should constrain the system states, then design the situational alignment in the CMSI, going down to control actions by STPA, and finally design the run-time reaction to exceptions, comprising the detection, alarms, troubleshooting, resilience (Hollnagel, Woods, and Leveson 2006; Woods 2015) and recovery.

Enforcement is based on two comple-

mentary mechanisms: preventing inconsistent scenarios, and detecting/interpreting deviations when they appear. The approach restricts the set of possible behaviors relative to the active operational scenario. Within each CMSI, only actions matching the scenario are permitted, while inconsistent commands are blocked upfront, reducing the need for procedural protections and human corrective intervention.

In the proposed methodology, the design process advances 'outside-in'. First, the operational boundary between the system and its environment is defined, and only then are internal mechanisms examined. The analysis focus is on external interactions, services provided, connections

with operators, collaborative systems, and the operational environment. From these external interactions, the operational network is derived, representing the distributed coordination structure between controllers, services, and operational assumptions. The AF296 accident demonstrates how inconsistent assumptions between pilot intent and thrust service states created a hazardous coordination structure, even though each component independently operated reasonably.

Following the definition of external interactions, the operational space is structured using scenarios and operational state representations. The design includes defining the ownership mission, characterizing the operational context, and deriving relevant controller states. Scenarios minimize ambiguity by creating coherent operational narratives, while the CSR layer translates these narratives into operational control states. The Air France 447 accident illustrates how differing interpretations of the same state between the autopilot and human pilots created coordination gaps that led to inappropriate responses during the stall.

The next phase develops the CMSI coordination architecture, responsible for maintaining interpretation consistency across system components. The architecture includes scenario-to-state mapping, operational state representations, component simulation via digital twins (Grieves and Vickers 2017), and mechanisms to preserve consistency between distributed process models (Leveson 2011). A CMSI mismatch arises when operational interpretations become mutually inconsistent despite active coordination dependencies. The Therac-25 accidents show how valid local operational assumptions produced a

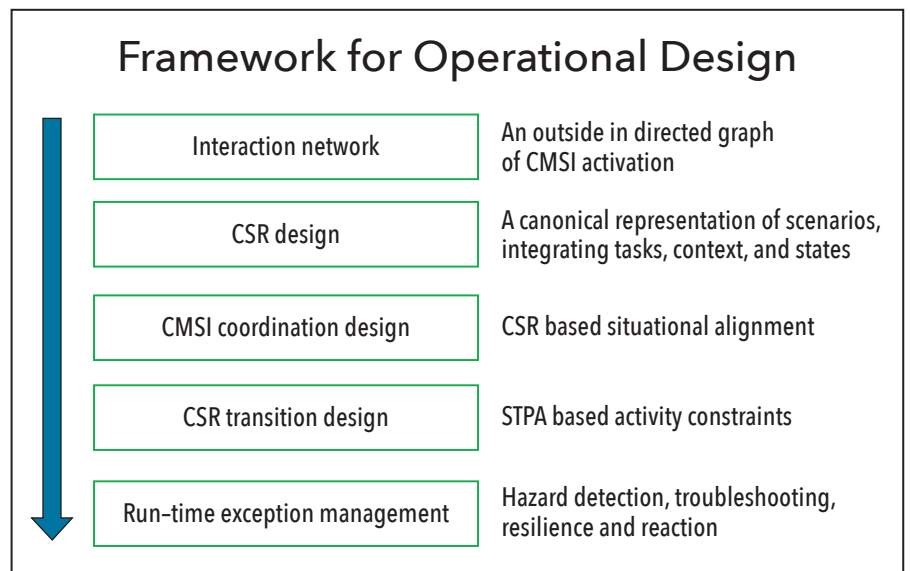


Figure 8. Framework for operational design

hazardous global state due to a mismatch between the radiation controller and the physical tray position. The proposed approach counters this complexity by bounding permitted transitions and commands per scenario instead of enumerating all possible system states.

Subsequently, the transition control architecture is developed to manage interaction dynamics and time-dependent behavior. This phase includes preview mechanisms for early prediction, transition constraint analysis, verification of timing relationships and action sequences, and operational decision support. The emphasis shifts from static state evaluation to understanding potential future system evolution. The Air China Nagoya accident demonstrates a situation where the automation transitioned to TO/GA mode while the pilots continued acting under normal landing assumptions. The risk arose not from a single fault but from a lack of synchronization in the transition between human and automated control states, highlighting the need to pre-identify hazardous interaction sequences, feedback delays, and inconsistent timing conditions.

The theory also addresses troubleshooting and operational resilience (Hollnagel, Woods, and Leveson 2006; Woods 2015). Troubleshooting is designed to return the system to an acceptable functioning level under time pressure and partial information, rather than providing an exhaustive explanation of the failure. While traditional systems relied on heuristics and pattern recognition, modern systems include backup mechanisms, including the reconstruction of state sequences, decisions, and interactions that led to the anomaly. The final stage involves exception and resilience planning, assuming that all conditions cannot be foreseen. This stage includes alerting mechanisms, resilience strategies, STPA integration, and converting unsafe control conditions into operational defense structures. The core idea is that the system must not only prevent failures but continue operating in a controlled manner after coordination loss and recover from it. In summary, the chapter presents a hierarchical design path where operational intent constrains scenarios, scenarios constrain control states, and control states constrain system behavior and resilience mechanisms.

DISCUSSION AND FUTURE HORIZONS

The text presents an operational safety engineering framework that expands STPA from static design analysis into runtime coordination. The primary argument is that in complex systems, safety does not arise solely from individual component integrity, but from the capacity of controllers, services,

and operators to maintain a coordinated operational interpretation of the system state. Thus, many systemic failures are not the result of a single faulty component, but of a divergence between local process models (Leveson 2011), operational assumptions, and differing state perceptions. The proposed framework adds a runtime scenario coordination mechanism to STPA using the CSR, which acts as a shared operational reference to synchronize distributed components. Rather than treating safety merely as the prevention of unsafe control actions, the system continuously monitors whether all participants still share the same understanding of the operational situation, mission context, and operating conditions. This maintains STAMP's core assumptions while expanding the operational aspect into dynamic coordination.

The article distinguishes between the physical state of the system and its operational interpretation. Controllers operate processes based on local models derived from partial observations and delayed feedback, whereas scenarios define bounded coordination classes containing rules for transition, synchronization, and permitted behavior. This approach addresses the combinatorial growth of the state space in modern systems. Instead of mapping every possible combination of states, the system operates within a limited number of operational scenarios with defined constraints, executing an engineering shift from exhaustive state enumeration to managing architectural constraints at a higher abstraction level.

The framework also alters the conception of runtime coordination. Instead of relying primarily on procedures and human coordination, coordination constraints are embedded directly into the architecture. The CSR functions as an operational invariant against which states and control actions are evaluated. Unsafe behavior is prevented not just via alerts, but through the structural exclusion of inconsistent operational configurations. This aligns with broad trends in resilient systems design, where safety is achieved via pre-bounded and pre-controlled operational structures. In the human factors domain, the framework views operator errors primarily as coordination phenomena rather than personal cognitive failures. In many accidents, operators acted reasonably relative to their local information, but the overall system lost operational consistency. Therefore, engineering responsibility shifts from an operator-centric view to designing systems that actively maintain consistency relative to the scenario and reduce cognitive load. Scenarios serve as operational abstractions that limit the number of interpretations

an operator must hold simultaneously. However, a central role remains for human judgment under conditions of uncertainty or degraded functioning.

The paper stresses that complex systems will inevitably encounter unpredicted situations. Therefore, the framework does not attempt to guarantee complete correctness under every possible condition but rather bounds behavior during uncertainty. When a scenario-guided consistency breach occurs, the system transitions to more restricted, safe operational states to prevent uncontrolled escalation. Alongside the benefits, substantial limitations are noted. Successful implementation depends heavily on the quality of scenario definitions and transition logic precision. Furthermore, maintaining CSR consistency across distributed systems can generate synchronization overhead and delays. The framework also does not resolve fundamental failures such as software bugs, defective sensors, or cyber-attacks. Moreover, it remains at an architectural level and requires formalization of synchronization processes, consistency metrics, validation mechanisms, and runtime enforcement protocols.

Looking forward, the connection to MBSE (Madni and Sievers 2018; Friedenthal et al. 2014) and digital twins (Grieves and Vickers 2017) is presented. The framework integrates naturally with SysML models (Friedenthal et al. 2014), control structures, runtime monitoring, and resilient operational design. The CSR could eventually enable a bridge between design models and real-time operational supervision via scenario-oriented digital twins. Digital twins are presented as physical simulations for controller decision-making, but also as active support mechanisms for preserving operational coherence, identifying inconsistencies, and validating scenario constraints.

A significant portion of the discussion is dedicated to the contribution of large language models (Bommasani et al. 2021; Weidinger et al. 2022) (LLMs (Bommasani et al. 2021)). AI models (Russell and Norvig 2021) are presented as supporting tools rather than operational authorities. In design, they can assist in organizing knowledge, identifying latent assumptions, generating alternative scenarios, and analyzing consistency. In STPA, they can assist in identifying unsafe control actions and managing the combinatorial complexity of hazardous interactions. At runtime, LLMs could serve as an adaptive interpretation layer that explains anomalies, merges information from disparate sources, and supports situational awareness. However, because their output is probabilistic and prone to semantic

hallucinations, they cannot replace formal models and validated engineering constraints. In troubleshooting, LLMs are described as semantic correlation engines capable of unifying logs, reports, telemetry, and procedures into a single interpretative space, supporting hypothesis generation and operational reconstruction even with partial data. Yet, because they relate to semantic plausibility rather than causal truth, they can produce convincing but incorrect narratives. Thus, it is proposed to embed them within constraint-based

diagnostic frameworks, where formal models and verification mechanisms remain the authoritative components.

The conclusion emphasizes a conceptual shift from safety engineering focused on individual components toward managing the coherence of distributed operational interpretations. The primary engineering goal becomes maintaining scenario-guided consistency and runtime coordination under conditions of uncertainty, partial observation, and operational changes. The proposed theory still requires formalization

and empirical validation but is presented as a scalable architectural direction for complex socio-technical systems. ■

ACKNOWLEDGMENTS

The author thanks Avigdor Zonnenshain, Valerie Sitterle, and Thomas Allen McDermott Jr. for their insightful comments. The author used AI-based tools (OpenAI ChatGPT and Google Gemini) to assist in drafting and editing the manuscript. The author is solely responsible for the content.

REFERENCES

- Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. "Concrete Problems in AI Safety." arXiv preprint arXiv:1606.06565.
- Bommasani, Rishi, Drew A. Hudson, Ehsan Adeli, et al. 2021. "On the Opportunities and Risks of Foundation Models." arXiv preprint arXiv:2108.07258.
- Boy, Guy A. 2013. *Orchestrating Human-Centered Design*. Berlin: Springer.
- Boy, Guy A., and Jurgis Narkevicius. 2014. "Context-Based Design of Socio-Technical Systems." *Proceedings of the European Conference on Cognitive Ergonomics*.
- Dekker, Sidney. 2011. *Drift into Failure: From Hunting Broken Components to Understanding Complex Systems*. Farnham: Ashgate.
- Dekker, Sidney. 2014. *The Field Guide to Understanding Human Error*. 3rd ed. Farnham: Ashgate.
- Endsley, Mica R. 1995. "Toward a Theory of Situation Awareness in Dynamic Systems." *Human Factors* 37 (1): 32-64.
- Friedenthal, Sanford, Alan Moore, and Rick Steiner. 2014. *A Practical Guide to SysML*. 3rd ed. Waltham, US-MA: Morgan Kaufmann.
- Grieves, Michael, and John Vickers. 2017. "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems." In *Transdisciplinary Perspectives on Complex Systems*.
- Hollnagel, Erik. 2014. *Safety-I and Safety-II: The Past and Future of Safety Management*. Farnham: Ashgate.
- Hollnagel, Erik, David D. Woods, and Nancy Leveson. 2006. *Resilience Engineering: Concepts and Precepts*. Farnham: Ashgate.
- Hutchins, Edwin. 1995. *Cognition in the Wild*. Cambridge, US-MA: MIT Press.
- Lee, Edward A. 2008. "Cyber Physical Systems: Design Challenges." *Proceedings of the 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*.
- Leveson, Nancy G. 2004. "A New Accident Model for Engineering Safer Systems." *Safety Science* 42 (4): 237-270.
- Leveson, Nancy G. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, US-MA: MIT Press.
- Madni, Azad M., and Michael Sievers. 2018. "Model-Based Systems Engineering: Motivation, Current Status, and Research Opportunities." *Systems Engineering* 21 (3): 172-190.
- Norman, Donald A. 1981. "Categorization of Action Slips." *Psychological Review* 88 (1): 1-15.
- Rasmussen, Jens. 1997. "Risk Management in a Dynamic Society: A Modelling Problem." *Safety Science* 27 (2-3): 183-213.
- Reason, James. 1990. *Human Error*. Cambridge: Cambridge University Press.
- Russell, Stuart, and Peter Norvig. 2021. *Artificial Intelligence: A Modern Approach*. 4th ed. Hoboken, US-NJ: Pearson.
- Sarter, Nadine B., and David D. Woods. 1995. "How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control." *Human Factors* 37 (1): 5-19.
- Weidinger, Laura, John Mellor, Maribeth Rauh, et al. 2022. "Ethical and Social Risks of Harm from Language Models." arXiv preprint arXiv:2112.04359.
- Woods, David D. 2015. "Four Concepts for Resilience and the Implications for the Future of Resilience Engineering." *Reliability Engineering & System Safety* 141: 5-9.

APPENDIX – CASE STUDIES

Analysis of well documented accidents indicates that many of them involve mismatch between the situation of the controller and at least one of the services.

Source of these cases: <https://avi.har-el.com/eng/Articles/index.htm>
These studies demonstrate the need to enforce coordinated situations between the controller and the services.

Transportation

	Controller	Situation	Service	Situation
AF 296 (1988)	Pilot	Pullup	Thrust	disable
AF 447 (2009)	Autopilot	On	Pivot tube	iced
→	Pilot	Pullup	Airplane	stall
Nagoya (1994)	Pilot	Final Approach	TO/GA	activated
PL 603 altimeter (1996)	Pilot	Cruise flight	Altimeter	maintenance
Torrey Canyon (1967)	Helm	Manual control	wheel	disconnected

Industry

	Controller	Situation	Service	Situation
Bhopal (1983)	Operator	Normal	Container	Maintenance
TMI (1979)	Main pump	Disabled	Backup pump	In maintenance

Medical

	Controller	Situation	Service	Situation
Therac 25 (1985-87)	Treatment	X-ray	Tray position	Out

Missile Launch

	Controller	Situation	Service	Situation
Brahmos (2022)	Inspection	Inner state	Connectors	Junction box
Cheongung (2019)	Maintenance	Test cable	Connection	Functional

Friendly Fire Accidents

	Controller	Situation	Service	Situation
Tseelim A (1988)	Supported u.	Stage 2	Supporting u.	Stage 1
Tseelim B (1990)	Supported u.	Dry run	Supporting u.	Live ammunition
Kandahar (2001)	Supported u.	Tgt acquisition	GPS	Navigation

Home appliances

	Controller	Situation	Service	Situation
Air conditioning	Control unit	Activated	Activation	Delay
Home TV	Control unit	Scrolling	Mode	Tuning

Trustworthy AI for Quantitative Systems Engineering via Deterministic Delegation to Verified Tools

Raymond Madachy, rjmadach@nps.edu

Copyright ©2026 by Raymond Madachy. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Generative AI offers substantial capability in systems engineering processes for multimodal interaction supporting elicitation, interpretation, and synthesis. However, large language models (LLMs) remain unreliable for exact quantitative computation as they may produce incorrect or hallucinated results when executing mathematical and domain-specific quantitative calculations. Systems engineering is pervasively grounded in quantitative models that demand determinism, correctness, and traceable results, requirements that probabilistic language models cannot reliably satisfy on their own. A deterministic delegation pattern is presented that addresses this limitation by constraining the LLM to input interpretation, assumption documentation, and results presentation while assigning all authoritative computation to verified external tools through formally defined API endpoints. Identical inputs produce identical results, every output carries full provenance, and users can independently inspect or rerun any analysis via a returned source URL. The pattern is demonstrated through application domains for systems engineering cost modeling using the Constructive Systems Engineering Cost Model (COSYSMO) and system reliability analysis using fault tree and reliability block diagram methods. Examples at increasing levels of complexity show how the assistants translate diverse user inputs into complete, reviewable input configurations before invoking the deterministic backend, combining the accessibility of conversational AI with the correctness and auditability that engineering decisions require.

INTRODUCTION AND BACKGROUND

Generative AI can substantially improve systems engineering processes through natural-language interaction, multimodal interpretation, and automated synthesis of technical information. Large language models (LLMs) are especially useful for elicitation, explanation, summarization, and organization of engineering data. However, they remain unreliable for exact quantitative computation, strict model semantics, and domain-specific numerical reasoning. Because LLMs generate outputs probabilistically, they may produce incor-

rect arithmetic, inconsistent reasoning, or fabricated results that appear plausible but are not reproducible.

This limitation is particularly important in systems engineering, where decisions are grounded in quantitative models for cost, schedule, performance, reliability, and risk. Engineering analyses require determinism, traceability, and independently verifiable results. Even small numerical errors can undermine confidence, create audit concerns, or propagate into poor technical and programmatic decisions. At worst, they may lead to system failures if not detected

and fixed. Conversational capability alone is therefore insufficient for quantitative systems engineering applications.

This article presents a trustworthy AI pattern for quantitative systems engineering based on deterministic delegation. In this approach, the LLM is constrained to interpretation, clarification, assumption management, orchestration, and presentation of results, while all authoritative computation is delegated to verified external tools through formally defined API interfaces. The language model does not perform the engineering

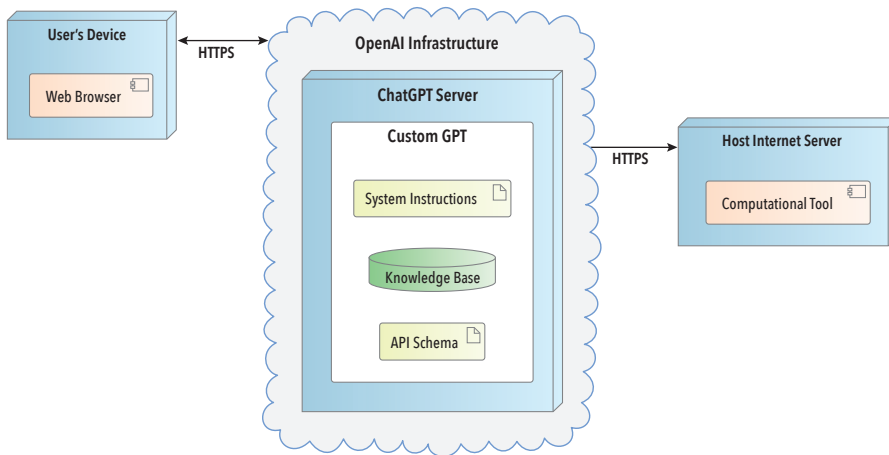


Figure 1. Custom GPT architecture with API to deterministic tool

mathematics internally. Instead, it translates user intent into structured model inputs, invokes deterministic computational services, and returns the resulting outputs with full provenance.

Many established engineering tools and models are difficult to use directly because they require specialized terminology, structured parameterization, and detailed procedural knowledge. An AI assistant can reduce this barrier by helping users express problems in ordinary language, identify missing information, organize assumptions, normalize terminology, and present reviewable inputs and outputs. Deterministic backend tools then ensure that the actual calculations remain reproducible, validated, and independently verifiable. Separating these responsibilities is therefore central to the approach: the LLM handles interpretation and communication while verified tools perform all computation, guaranteeing that engineering decisions rest on methods whose behavior is defined, repeatable, and independently verifiable.

Under deterministic delegation, identical inputs produce identical outputs. Each analysis can also return complete provenance information including interpreted inputs, assumptions, model versions, API requests, and source URLs that allow users to independently inspect or rerun the analysis. This preserves the strengths of conversational AI while bounding hallucination risk away from the computation itself.

The paper demonstrates the pattern in two representative quantitative systems engineering domains: system reliability analysis and systems engineering cost modeling. The reliability examples range from simple closed-form parallel redundancy calculations to structured reliability block diagram (RBD) analyses generated from narrative mission descriptions. In each case, the assistant supports problem framing, parameter elicitation, and interpretation of results,

while deterministic backend reliability tools perform the authoritative computations and generate the visualizations.

The second application domain is systems engineering cost estimation using the Constructive Systems Engineering Cost Model (COSYSMO) [1]. Here the assistant translates project descriptions into normalized COSYSMO parameters, applies defaults and assumptions where appropriate, presents reviewable input configurations, and invokes a verified external estimator. The returned effort, schedule, and cost results are exact outputs from the deterministic COSYSMO implementation, accompanied by traceable provenance and direct links to the populated estimation tool.

SYSTEM ARCHITECTURE AND DESIGN

The demonstrated assistants are implemented as custom GPTs using the OpenAI platform [6] [7] [8], although the deterministic delegation pattern is not specific to any particular LLM provider or deployment architecture. Equivalent implementations could be constructed using other foundation models, orchestration frameworks, agent architectures, or locally hosted environments, provided that conversational interaction remains separated from deterministic computations.

The demonstrated architecture consists of a user interaction layer, an AI mediation layer, and external deterministic computational services, as shown in the deployment model in Figure 1. The GPT mediates communication between the user and the quantitative backend, translating conversational inputs into structured model parameters and presenting the returned results in a user-oriented form.

The separation of responsibilities between the AI component and the computational service is central to the deterministic delegation pattern. The GPT supports elicitation, clarification, normalization of

terminology, assumption management, and explanation of results, but it does not perform authoritative quantitative computation internally. All engineering calculations are executed by independently validated external tools whose behavior is deterministic and reproducible.

The external quantitative services are exposed through formally defined APIs. The services provide authoritative quantitative computation through reproducible interfaces. Each invocation may also return provenance information including interpreted inputs, model version, generated request, and a verifiable source URL that allows users to independently inspect or rerun the analysis directly within the external tool environment.

The AI mediation layer is configured through control artifacts including system instructions, a domain knowledge base, and formally defined API schemas. The instructions govern assistant behavior and operational constraints, the knowledge base provides supplementary contextual information, and the schemas define permitted tool interfaces, input parameters, and returned outputs. Together, these artifacts constrain assistant behavior and ensure that quantitative requests are routed only through approved deterministic services.

All communication between components occurs over HTTPS. The computational services are hosted independently from the AI assistant, preserving separation between conversational interaction and deterministic model execution. This architecture improves reproducibility, auditability, regression testing, and governance while allowing conversational AI to simplify interaction with otherwise specialized engineering tools.

Although the examples in this paper focus on system reliability analysis and systems engineering cost estimation, the architecture generalizes to other quantitative systems engineering domains. The overall process remains unchanged while the domain knowledge, instructions, API schemas, and deterministic backend services are adapted to the target analytical method.

Operational Sequence

Figure 2 illustrates the interaction sequence between the user, the assistant, and the external computational service. The process begins when the user provides engineering information such as system descriptions, project characteristics, requirements, reliability parameters, or other quantitative inputs. The assistant interprets the request, normalizes terminology into structured model parameters, identifies missing information, applies defaults where

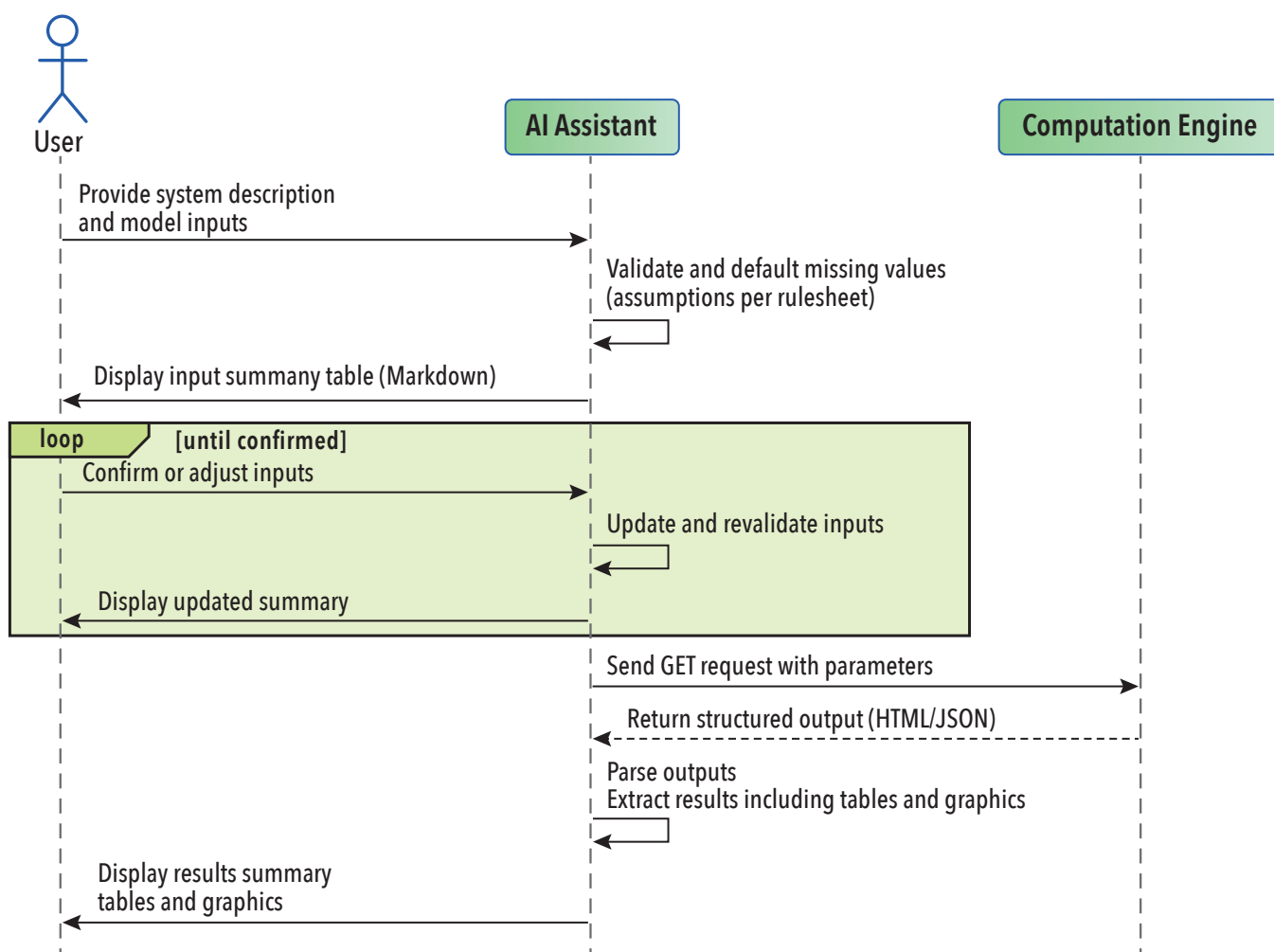


Figure 2. Operational sequence model

appropriate, and presents a reviewable input summary for confirmation. This iterative elicitation process allows assumptions and parameter interpretations to be inspected before computation occurs.

After confirmation, the assistant constructs an encoded API request according to the associated schema and submits it to the computational service, typically as an [HTTPS](#) GET request with parameterized inputs. The external service performs the authoritative computation and returns outputs in structured formats such as JSON or HTML together with provenance information and a verifiable source URL.

The assistant parses the returned outputs and presents them in a user-oriented form suitable for interpretation and decision support. Returned results may include numerical outputs, tables, graphics, model artifacts, assumptions, and hyperlinks to the populated external tool environment. Users may independently inspect, modify, or rerun the analysis outside the conversational environment, providing an additional mechanism for verification and auditability.

The interaction sequence intentionally constrains the LLM to interpretation, orchestration, and presentation responsibilities while preserving quantitative computation within validated external tools. This separation reduces hallucination risk in numerical analysis while retaining the accessibility and flexibility of conversational interaction.

Instructions

Each assistant is governed by explicit written instructions that define operational behavior, interaction style, permissible assumptions, required tool usage, and output formatting. The instructions constrain how the assistant interprets requests, collects inputs, invokes external services, and presents returned results. They also explicitly prohibit internal quantitative computation when authoritative tools are available.

The instructions for these example assistants are written in Markdown to support hierarchical organization, readability by both humans and machines, and structured operational logic. They complement the API schema by governing

behavioral logic rather than interface structure. Whereas the schema defines the allowable inputs and outputs of an external service, the instructions govern how the assistant reduces user requests into required parameters, determines when clarification is needed, applies defaults or assumptions, invokes named operations, and formats outputs for review.

The instructions also establish operational boundaries intended to improve trustworthiness and consistency. Examples include transparent disclosure of assumptions, mandatory tool invocation for quantitative analysis, constraints on unsupported reasoning paths, and prescribed provenance presentation in returned results. Representative instruction excerpts are provided in later examples.

API Schemas

The API schemas define the formal interface between the assistant and the external quantitative services. The demonstrated assistants use OpenAPI 3.1 specifications written in YAML to describe available endpoints, named operations,

required parameters, returned outputs, and parameter constraints for each computational tool.

Each field is explicitly typed and validated so the assistant can construct well-formed requests and reliably interpret returned outputs. Named operation identifiers provide a stable mapping between assistant instructions and the corresponding service invocation. The schemas therefore function both as interface contracts and as constraints on tool usage.

This structured interface definition improves reproducibility and interoperability while reducing ambiguity in how the assistant accesses external computational services. Because the schemas are both machine-readable and human-readable, they also support inspection, maintenance, extension to additional tools, and integration into broader digital engineering environments. Examples of instruction sets and corresponding API schemas are provided in the following sections.

SYSTEM RELIABILITY ANALYSIS

System reliability analysis is well suited to deterministic delegation because reliability models require exact probabilistic computation, explicit structural assumptions, and reproducible results. Reliability methods are commonly used to evaluate system reliability, identify vulnerable structures, and compare alternative architectures under failure assumptions. Common representations such as closed-form reliability equations, fault trees, and reliability block diagrams (RBDs) support quantitative reasoning about system failure and redundancy, but they also require disciplined parameterization and mathematically consistent evaluation.

Generative AI can assist by interpreting system descriptions, organizing model structure, identifying missing parameters, and explaining results, while validated reliability solvers perform the computation. The examples below illustrate increasing levels of reliability modeling complexity. The first uses a simple parallel redundancy fault tree calculator to reduce a natural-language request into the required quantitative inputs for deterministic evaluation. The second extends the approach to structured reliability block diagram (RBD) modeling generated from a narrative mission description.

Examples

The first reliability example demonstrates direct deterministic evaluation of a simple parallel system. The user requests the system reliability of a two-component redundant system over a fixed mission duration, providing only the essential quantities needed for calculation. The assistant reduces the request to the required reliability parameters, invokes the external reliability calculator tool [2], and returns the exact component reliability, overall system reliability, system failure probability, fault-tree visualization, and verifiable source URL.

For a component with constant failure rate, reliability over mission time is given by:

$$R_i(t) = e^{-t/\theta_i}$$

Where

R_i is the reliability of the i^{th} component

t is the mission time (hours)

θ_i is the MTBF of the i^{th} component as the mean time between failures (hours)

For m identical components operating in parallel, the overall system reliability is:

$$R_{\text{sys}}(t) = 1 - (1 - R_i(t))^m$$

Where

R_{sys} is the system reliability

R_i is the reliability of the i^{th} component

m is the number of identical components

The instruction set and API schema together operationalize deterministic delegation by constraining both assistant behavior and allowable computational interfaces. The instructions excerpt in Listing 1 shows how user requests are reduced to the required quantitative parameters while prohibiting internal computation. Missing values may be estimated when reasonable with assumptions stated explicitly. The instructions also require direct invocation of the named schema operation `ReliabilityCalculate` and prescribe inclusion of the calculation source URL and returned SVG fault tree visualization in the output.

Listing 1. GPT instructions excerpt

```
## Processing
Reduce all reliability requests to these three parameters:
- mtbf
- mission_time
- redundancy

If any missing parameter can be reasonably estimated, state that it is assumed.
If required parameters cannot be confidently estimated, ask only for the missing ones.

Use the named schema operation `ReliabilityCalculate` to invoke the external tool with the following inputs. Do not compute internally.

```json
{
 "mtbf": ...,
 "mission_time": ...,
 "redundancy": ...
}
```

## Outputs
After the tool returns, summarize:
- component reliability
- system reliability
- system failure probability

Also include:
- the exact calculation source URL
- the returned fault tree rendered in a fenced `svg` block
```

The API schema in Listing 2 defines the formal interface between the assistant and the external reliability service. Written in OpenAPI 3.1 YAML, it specifies the server endpoint, referenced operation identifier, and typed query parameters for `mtbf`, `mission_time`, and `redundancy`. The schema constrains request construction to valid parameter combinations and ensures reproducible invocation of the external reliability model. The same delegation pattern can be generalized to other quantitative tools by adapting the operation identifiers, input parameters, and output formatting rules.

The user prompt for this simple three component parallel system is shown below, and the resulting assistant output is illustrated in Figure 3.

Estimate the 5 day reliability of a triple-redundant marine power plant built from Caterpillar diesel generator sets.

Listing 2. API schema example

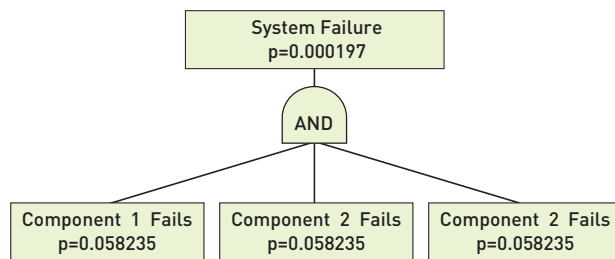
```

openapi: 3.1.0
servers:
  - url: https://se-lib.org/online/
paths:
  /reliability.py:
    get:
      operationId: ReliabilityCalculate
      parameters:
        - in: query
          name: mtbf
          description: Mean Time Between Failures in hours.
          required: true
          schema:
            type: number
            minimum: 0
        - in: query
          name: mission_time
          description: Mission time in hours
          required: true
          schema:
            type: number
            minimum: 0
        - in: query
          name: redundancy
          description: the number of identical components
            operating in parallel (integer >= 1)
          required: true
          schema:
            type: number
            minimum: 1

```

Assuming Caterpillar-class industrial diesel generator sets with a default MTBF of 2,000 hours, a 120-hour (5-day) mission, and three active-redundant generators operating in parallel the system reliability is 99.98%. The 2,000 hour MTBF is a conservative engineering estimate derived from publicly available emergency diesel generator reliability studies and adjusted to reflect marine operating conditions.

- Component reliability: 0.941765 (94.18%)
- System reliability: 0.999803 (99.98%)
- System failure probability: 0.000197 (0.02%)



Calculation source ↗

Figure 3. AI output for parallel component system reliability

The returned result includes a source URL with the input parameters populated in the external reliability tool. Users can independently verify the calculation, modify parameters, and rerun the analysis directly within the tool environment. Figure 4 shows the corresponding endpoint tool state, which reproduces the same reliability results and fault-tree visualization returned through the assistant.

Parallel Component Reliability Analyzer

This tool computes system reliability using an exponential (constant failure rate) model. Component reliability is $R = e^{-t/MTBF}$. Optional redundancy assumes m identical components in parallel: $R_{system} = 1 - (1 - R_{component})^m$

MTBF (hours):

Mission time t (hours):

Redundancy (parallel $m \geq 1$):

Calculate

Results

Component reliability $R_{component} = 0.941765$ (94.18%)

System reliability $R_{system} = 0.999803$ (99.98%)

System failure probability $F_{system} = 0.000197$ (0.02%)

Figure 4. Reliability analyzer tool with AI inputs

The next reliability example extends the approach from direct scalar reliability computation to structured system modeling from a narrative mission description. Instead of supplying only a few numerical inputs, the user describes a combat system mission architecture with redundant sensing, command-and-engagement control function, and launch functions. The assistant translates the narrative description into a structured reliability block diagram (RBD), organizes the component MTBF values into a valid RBD model representation, and submits the resulting structure to the backend reliability tool [3] for deterministic evaluation.

The RBD evaluation framework is represented as a recursive function over the system structure. Each node corresponds to either a component block, a series structure in which all child elements must succeed, or a parallel structure in which at least one redundant path must survive. System reliability is computed recursively from the component level upward through the overall architecture in the following equation.

$$R_{sys}(t) = f(\tau) \text{ where } \begin{cases} e^{-t/\theta_i} \text{ or } r_i & \tau = \text{Block}_i \\ \prod_{i=1}^N f(\tau) & \tau = \text{Series} \\ 1 - \prod_{i=1}^N (1 - f(\tau)) & \tau = \text{Parallel} \end{cases}$$

Where

$R_{sys}(t)$ is the overall system reliability at mission time t

$f(\tau)$ is the recursive reliability evaluation function applied to node τ

τ represents either a component block, series structure, or parallel structure within the RBD

τ_i is the i -th child node of structure τ

r_i is the mean time between failures (MTBF) of component i

N is a directly specified reliability value for component i is the number of child nodes within the series or parallel structure

At the leaf level, reliability may be specified directly or derived from an exponential constant-failure-rate model using component MTBF and mission duration. Series structures require all child

I modeled the combat system as a three-stage series RBD for a 12-hour mission: a redundant detection group, a redundant command-and-engagement-control group with two parallel processing paths, and a three-way redundant launcher group. Using the specified MTBF values and assuming independent failures within each redundancy group, the backend tool computed an overall mission reliability of $R_{sys}(12h)=0.999234$. The corresponding mission failure probability is 0.000766.

This result indicates a highly resilient architecture for the stated mission duration. The high mission reliability is driven by redundancy at each major functional stage, which substantially reduces vulnerability to any single component failure. The diagram below shows the exact series-parallel structure evaluated by the backend reliability tool and the component-level reliabilities used in the calculation.

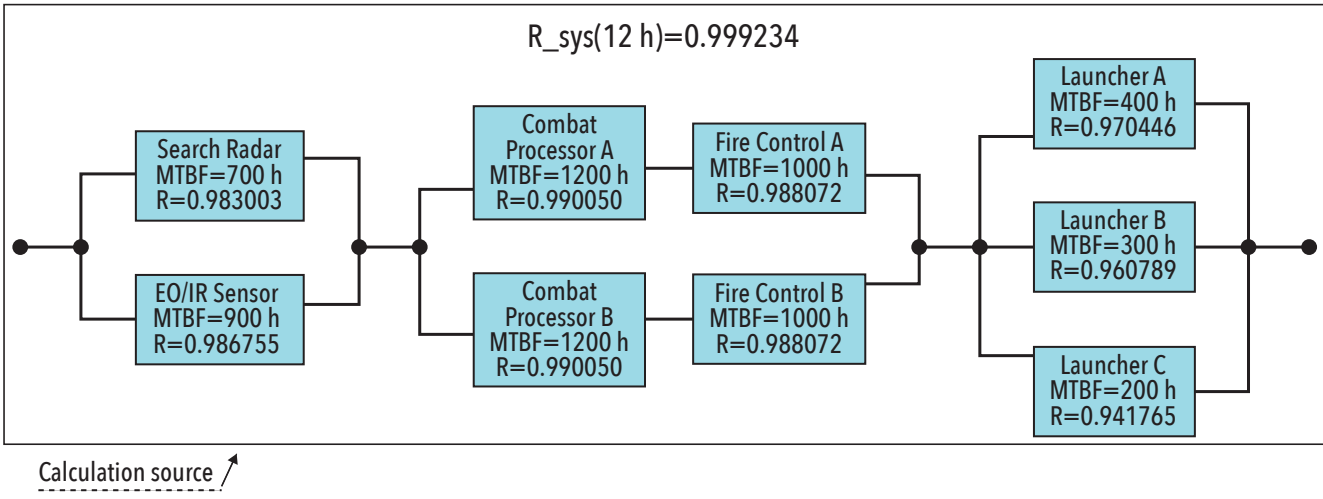


Figure 5. AI output for system reliability and reliability block diagram

subnetworks to succeed simultaneously, so their reliabilities multiply. Parallel structures fail only if all redundant paths fail, making overall reliability the complement of the product of individual failure probabilities. For the combat-system example that follows, the resulting architecture consists of parallel redundancy across major mission functions with series-connected elements within individual branches, evaluated deterministically from the specified MTBF values by the backend reliability tool.

The user request is shown below. The resulting system reliability and the generated reliability block diagram are shown in Figure 5.

Create and evaluate a reliability block diagram for a combat system mission that lasts 12 hours. Detection should succeed if either the search radar or the EO/IR sensor is available. The command-and-engagement-control function should be modeled with two redundant processing paths: one path is Combat Processor A in series with Fire Control A, and the backup path is Combat Processor B in series with Fire Control B. The weapon-launch function should succeed if either Launcher A or Launcher B or Launcher C is available. Use MTBF values of 700 hours for Search Radar, 900 for EO/IR Sensor, 1200 for each Combat Processor, 1000 for each Fire Control component, and the Launchers with respective MTBFs of 400, 300 and 200 hours. Send the model to the backend reliability tool, compute the mission reliability, and return both the visualization and the final quantitative result.

These reliability analyses demonstrate the deterministic delegation pattern across increasing levels of modeling complexity, from direct closed-form reliability computation to AI-assisted construction of structured reliability models from narrative system descriptions with deterministic backend evaluation.

SYSTEMS ENGINEERING COST MODELING APPLICATION

Systems engineering cost modeling combines formal quantita-

tive estimation with the ambiguity common to real projects. Users may know some cost drivers well and others only approximately, or may provide organizational and operational context rather than the exact parameters expected by a cost model. Generative AI can help bridge this gap by eliciting inputs in ordinary language, organizing them into complete parameter sets, documenting assumptions, and presenting reviewable configurations before computation occurs. The quantitative calculation itself, however, should remain outside the language model.

COSYSMO provides a validated method for estimating systems engineering effort, schedule, and cost from project size and cost driver information [1]. The model estimates overall labor effort and allocates it across activities and lifecycle phases. Its primary inputs are the effective size of the work to be performed together with project factors that influence effort. The high-level effort relationship expresses effort as a function of system size, a scale exponent, and effort multipliers.

$$Effort = A \cdot Size^B \cdot \prod_{i=1}^N EM_i$$

Where

Effort is in Person-Months (PM)

A is a constant derived from historical project data

Size is a sum of weighted Requirements, Interfaces, Algorithms and Scenarios

B is an exponent for the diseconomy of scale

EM_i is an effort multiplier for the *i_{th}* cost driver. The geometric product of *N* multipliers is an overall Effort Adjustment Factor (EAF) applied to the nominal effort.

The size term is further decomposed into weighted counts of requirements, interfaces, algorithms, and operational scenarios evaluated by difficulty level, providing the formal basis for the deterministic estimate produced by the external COSYSMO tool.

$$Size = \sum_{j=1}^k (\omega_e \Phi_{ef} + \omega_n \Phi_{n,j} + \omega_d \Phi_{d,j})$$

Where

k is the number of requirements, interfaces, algorithms and scenarios

$w_e, w_n,$ and w_d are the weights for easy, nominal, and difficult complexity levels

$\Phi_{x,j}$ is the quantity associated with size driver x at complexity level j

The COSYSMO assistant [6] uses instructions and API schemas adapted to the COSYSMO parameter set and the softwarecost.org

estimation service. After the interpreted inputs are confirmed, the assistant constructs a validated request to the external estimator, retrieves the returned results, and presents the outputs with full provenance and traceability.

The following example demonstrates deterministic delegation for a representative Navy C4I project requiring broader narrative interpretation, parameter normalization, assumption management, and structured preparation of tool-ready COSYSMO inputs before deterministic evaluation.

Example

A detailed naval domain example is shown in Figures 6 and 7. Rather than providing only direct COSYSMO inputs, the user

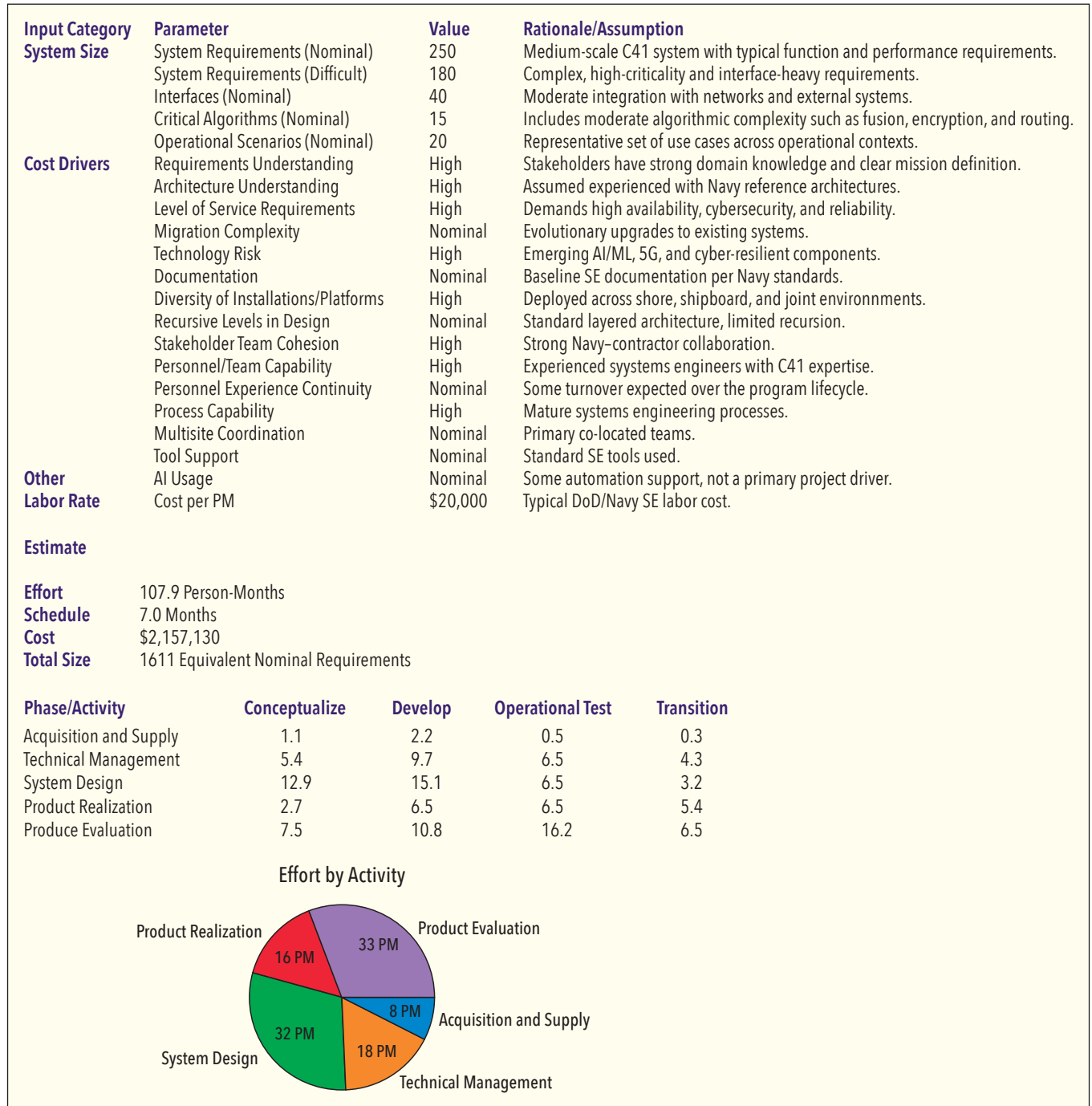


Figure 6. AI cost estimate output

Constructive Systems Engineering Cost Model (COSYSMO) with AI Usage

This tool computes systems engineering effort and schedule with COSYSMO. This version incorporates the new cost factor *AI Usage* with provisional effort multipliers

▶ **System Size**

| | Easy | Nominal | Difficult |
|-----------------------|------|---------|-----------|
| System Requirements | 0 | 250 | 180 |
| System Interfaces | 0 | 40 | 0 |
| Critical Algorithms | 0 | 15 | 0 |
| Operational Scenarios | 0 | 20 | 0 |

▶ **Cost Factors**

| | | | | | |
|-------------------------------|--------------------------------------|--|--------------------------------------|----------------------------------|--------------------------------------|
| Requirements Understanding | <input type="text" value="High"/> | Documentation Match to Lifecycle Needs | <input type="text" value="Nominal"/> | AI Usage | <input type="text" value="Nominal"/> |
| Architecture Understanding | <input type="text" value="High"/> | Number and Diversity of Installations/ Platforms | <input type="text" value="High"/> | Personnel Experience/ Continuity | <input type="text" value="Nominal"/> |
| Level of Service Requirements | <input type="text" value="High"/> | Recursive Levels in the Design | <input type="text" value="Nominal"/> | Process Capability | <input type="text" value="High"/> |
| Migration Complexity | <input type="text" value="Nominal"/> | Stakeholder Team Cohesion | <input type="text" value="High"/> | Multisite Coordination | <input type="text" value="Nominal"/> |
| Technology Risk | <input type="text" value="High"/> | Personnel/Team Capability | <input type="text" value="High"/> | Tool Support | <input type="text" value="Nominal"/> |

▶ **Labor Rate**

Cost per Person-Month (Dollars)

Figure 7. COSYSMO estimator tool with AI inputs

describes a representative Navy C4I system with multiple technical features, interfaces, and operational considerations. The assistant translates the narrative into structured COSYSMO size and cost-driver parameters, documents assumptions, and presents the resulting configuration for review before deterministic evaluation. The prompt is:

What is the estimated systems engineering effort for a representative Navy C4I project with 250 nominal requirements and 180 difficult requirements? Apply a realistic Navy C4I profile for the cost driver ratings.

The interpreted project characteristics are summarized in Figure 6 with the estimate. The table shows how the GPT translated the narrative project description into a complete, realistic, reviewable, tool-ready input set. Presenting the normalized inputs in this form gives the user a clear opportunity to verify all assumptions before computation, which is especially important when estimates depend heavily on interpretation, assumptions, and project framing. After confirmation, the structured request is submitted to the external COSYSMO estimator and returns the deterministic effort, schedule, cost, and activity-distribution results with full provenance. The calculation source link leads to the tool populated with the inputs for verification and revision [4] per Figure 7 (not shown are the identical outputs). Note that ‘with AI Usage’ in the tool name refers to a new cost factor for AI-assisted processes [9], not to the tool itself being called by AI.

Results and Discussion

Across the cost and reliability examples, the deterministic delegation pattern produced correct, reproducible tool outputs while preserving a natural conversational interaction process. In the reliability examples, the assistant reduced user requests into quantitative parameters or structured model representations, invoked external reliability tools, and returned exact results with visualizations and source URLs. In the COSYSMO examples, the assistant translated partial project descriptions into complete, reviewable parameter sets and returned effort, schedule, cost, and activity-distribution results with full provenance. In all cases, the assistant contributed primarily through interpretation, parameter organization, assumption management, and results presentation, while authoritative computation remained within validated external tools.

The examples also demonstrate increasing levels of modeling complexity. In simpler cases, the assistant functions as a guided interface that converts natural-language requests into valid model inputs and returns exact results in interpretable form. In the more complex C4I cost estimation and combat-system reliability examples, the assistant further reduces modeling burden by translating narrative descriptions into structured, tool-ready configurations that would otherwise require substantial manual effort and specialized tool familiarity.

A consistent benefit across the applications is improved transparency and verifiability. The assistant exposes interpreted inputs and assumptions, returns structured

outputs, and provides direct links allowing users to independently inspect, rerun, or refine analyses within the external tool environments. This improves auditability and user confidence while bounding hallucination risk to interpretation and elicitation rather than numerical computation itself.

Another practical advantage is traceability. Structured interaction between the assistant and backend tools simplifies logging inputs, preserving assumptions, comparing analyses, and performing regression testing. For organizations that must review estimates, audit analyses, or maintain a digital thread, these capabilities are central to acceptance and governance. The assistants intentionally preserve detailed interpreted inputs and assumptions rather than aggressively summarizing them, since reviewability and parameter transparency are central to trustworthy quantitative engineering analysis.

The examples demonstrate that trustworthy conversational AI in quantitative systems engineering depends not only on validated computational tools, but also on accurate parameter elicitation, transparent assumptions, constrained tool usage, and faithful provenance presentation. When these elements are combined, deterministic delegation provides a practical and governable approach for integrating conversational AI into quantitative engineering processes.

CONCLUSIONS AND FUTURE WORK

Large language models provide substantial capability for natural-language interaction, input elicitation, and interpretation of

engineering results, but they remain unreliable for exact quantitative computation. Quantitative systems engineering applications require determinism, reproducibility, traceability, and independently verifiable results that probabilistic language models alone cannot consistently guarantee.

The deterministic delegation pattern presented in this paper addresses this limitation by constraining the LLM to interpretation, orchestration, and presentation responsibilities while assigning authoritative computation to verified external tools through formally defined API interfaces. The resulting architecture preserves the accessibility and flexibility of conversational AI while ensuring that engineering calculations remain reproducible, auditable, and independently verifiable.

The reliability and COSYSMO examples demonstrate the pattern across increasing levels of modeling complexity, ranging from direct scalar calculations to structured model construction from narrative system descriptions. Across these examples, the assistant contributes primarily by reduc-

ing friction between realistic engineering problem descriptions and valid, tool-ready analyses, while validated backend services ensure computational correctness.

Beyond correctness, deterministic delegation provides broader systems engineering benefits including improved auditability, governance, regression testing, interoperability within the digital engineering thread, and reduced cognitive burden for users who are not specialists in quantitative modeling methods. Structured provenance, constrained tool invocation, and independently reproducible analyses also improve organizational trust and acceptance of conversational AI within engineering environments.

The demonstrated implementations use custom GPTs with external API-connected computational tools, but the delegation pattern generalizes to other LLM platforms, orchestration frameworks, and quantitative engineering domains. Future work will extend the approach to additional systems engineering applications, expanded multimodal input interpretation, AI-assisted project risk assessment augmenting the

expert COSYSMO method [5], and automated verification methods for validating equivalence of delegated computations across structured input conditions. Ongoing work is also consolidating the separate system reliability analysis capabilities into a more general-purpose reliability assessment environment with an AI front end to deterministic tools.

Looking toward 2030, generative AI will become increasingly embedded in systems engineering processes and digital engineering environments. As model capabilities continue to advance, deterministic delegation provides a scalable governance foundation for combining natural-language interaction with reproducible quantitative analysis. Constraining probabilistic AI systems to interpretation and orchestration roles while preserving validated computational backends can enable broader adoption of trustworthy AI across the systems engineering lifecycle without sacrificing the determinism, traceability, transparency, or auditability required for engineering decision making. ■

REFERENCES

- [1] Valerdi, R. 2005. *The Constructive Systems Engineering Cost Model (COSYSMO)*. PhD thesis, University of Southern California, Los Angeles, US-CA.
- [2] Madachy, R. 2026. *Parallel Component Reliability Analyzer*. Available: https://se-lib.org/online/parallel_component_reliability.py.
- [3] Madachy, R. 2025. *Reliability Block Diagram Analyzer*. Available: <https://se-lib.org/online/rbd.py>.
- [4] Madachy, R. 2025. *COSYSMO with AI Usage Tool*. Available: https://softwarecost.org/tools/cosysmo_ai/.
- [5] Madachy, R., and R. Valerdi. 2020. "Expert COSYSMO Systems Engineering Cost Model and Risk Advisor." In: *Handbook of Military and Defense Operations Research*, edited by N. Scala and J. Howard. Boca Raton, US-FL: CRC Press.
- [6] Madachy, R. 2025. *COSYSMO GPT*. OpenAI Custom GPT. Available: <https://chatgpt.com/g/g-68a4f33c568881919f1edf346af27e0b-cosysmo-gpt>.
- [7] Madachy, R. 2026. *Reliability Block Diagram Expert GPT*. OpenAI Custom GPT. Available: <https://chatgpt.com/g/g-69cac3ce7fb08191a6717276820fa2c2-reliability-block-diagram-expert>.
- [8] Madachy, R. 2025. *Parallel Component Reliability Analyzer GPT*. OpenAI Custom GPT. Available: <https://chatgpt.com/g/g-689e7a63ef1c81919c636501c6bfac4b-system-reliability-service>.
- [9] Madachy, R., R. Bell, and R. Longshore. 2025. "A Generative AI-Driven Systems Engineering Maturity and Cost Modeling Framework." In: *Augmented Intelligence in Systems Engineering and Engineered Systems: Proceedings of the 2025 Conference on Systems Engineering Research*, edited by A. Madni, D. Verma, M. Sievers, M. Wheaton, E. Ordoukhanian, and S. Purohit. Springer Nature.

Engineering Trust in AI-Enabled Systems: Lessons from the Trusted AI Challenge

Sami Saliba, Emma Meno, emmam99@vt.edu; Scott Lucero, dslucero@vt.edu; Tyler Cody, tcody@virginia.edu; Nathan Lau, nkclau@vt.edu; Carlo Lipizzi, clipizzi@stevens.edu; Hunter Moore, Jitesh Panchal, panchal@purdue.edu; Sachin Shetty, Zoe Szajnfarber, zsajnf@gwu.edu; and Peter Beling, pb3a@virginia.edu

Copyright ©2026 by Sami Saliba, Emma Meno, Scott Lucero, Tyler Cody, Nathan Lau, Carlo Lipizzi, Hunter Moore, Jitesh Panchal, Sachin Shetty, Zoe Szajnfarber, and Peter Beling. Permission granted to INCOSE to publish and use.

■ ABSTRACT

This article contributes a systems engineering perspective on trust in AI-enabled systems by synthesizing results from the Trusted AI Challenge. The Systems Engineering Research Center (SERC), with the U.S. Army DEVCOM Armaments Center at Picatinny Arsenal New Jersey, organized and conducted this student challenge across three academic semesters with six student/faculty teams. The challenge highlighted a broader evolution in systems engineering—from managing deterministic systems to architecting adaptive, human-centered systems in which trust must be continuously engineered and sustained. The challenge demonstrated engineering trust in AI is not solely a computer science problem—it is fundamentally a systems engineering challenge. By requiring teams to balance safety, explainability, and performance in partially observable, adversarial environments, the challenge catalyzed the development of actionable systems engineering methods.

1. INTRODUCTION: A DECADE OF TRANSFORMATION

Looking back across the decade from 2020 to 2030, the role of artificial intelligence (AI) in complex systems has fundamentally shifted. Early in the decade, AI primarily functioned as a supporting analytical capability—providing performance advantages while leaving final operational authority to human operators. Today, AI-enabled systems are deeply embedded across critical domains such as transportation, infrastructure, healthcare, and national security. These systems no longer simply inform decisions; they increasingly select and execute actions autonomously in real time.

As AI has transitioned toward autonomous decision-making and action-taking, the consequences of system behavior have

grown significantly. AI-enabled systems now directly influence safety, mission outcomes, and human decision-making. For systems engineering practitioners, this shift introduces a fundamental challenge: ensuring that these systems behave not only with high performance, but also with reliability, transparency, and appropriateness under conditions of uncertainty.

At the center of this challenge is **trust**. Trust characterizes the relationship between users and systems, reflecting users' confidence that the system will behave reliably—even when operating conditions differ from those anticipated during development (Choung, David, and Ross 2023). In practice, trust is neither simple nor static. It is shaped by system transparency,

predictability, and the degree to which system behavior remains aligned with user expectations over time.

Failures in trust can introduce significant operational risk. These risks generally manifest in two ways (Passi and Vorvoreanu 2022). In one case, users may distrust capable systems, leading to under-utilization and reduced mission effectiveness. In the other, users may over-trust systems, resulting in complacency and reduced oversight. Early studies in human-automation interaction illustrated this paradox. For example, participants in a simulated emergency evacuation scenario followed an unreliable robotic guide, ignoring clearly marked exits in favor of incorrect instructions (Robinette et al. 2016). Such behaviors highlight how

misaligned trust can persist even in the presence of contradictory evidence.

For AI-enabled systems, these risks are amplified. Over-reliance can lead to bypassed safeguards, erosion of situational awareness, and unnoticed divergence between human intent and system behavior—potentially persisting until a failure event occurs. As a result, trust cannot be treated as an emergent byproduct of performance; it must be intentionally engineered.

Traditional systems engineering provides a strong foundation for addressing these challenges through structured requirements, lifecycle processes, and disciplined integration. However, AI introduces characteristics that strain these approaches, including non-deterministic behavior, dependence on data quality, and the need for continuous validation as systems encounter new environments.

To explore how systems engineering practices must evolve in response, U.S. Army DEVCOM Armaments Center at Picatinny Arsenal (New Jersey, USA) sponsored a targeted research effort through the Systems Engineering Research Center (SERC). This effort took the form of the trusted AI challenge—a multi-university, student-oriented competition designed to examine how trust can be engineered into AI-enabled systems. By examining the methodologies and outcomes of this challenge, we can observe how systems engineering methods have advanced to address the rapid growth of AI-enabled systems over the past decade.

This article contributes a systems engineering perspective on trust in AI-enabled systems by synthesizing results from the trusted AI Challenge (SERC 2025). It highlights how systems engineering practices have evolved across lifecycle processes, human-AI architecture, validation infrastructure, and workforce development, and identifies emerging patterns that are likely to shape the discipline through 2030 and beyond. Taken together, these advancements reflect a broader evolution in systems engineering—from managing deterministic systems to architecting adaptive, human-centered systems in which trust must be continuously engineered and sustained.

The challenge also highlighted the importance of supporting infrastructure. Developing and validating AI-enabled systems requires both physical and digital capabilities that extend beyond traditional engineering environments. These include:

- Dedicated hardware and simulation testbeds for evaluating system behavior
- Robust data pipelines to support training, monitoring, and validation

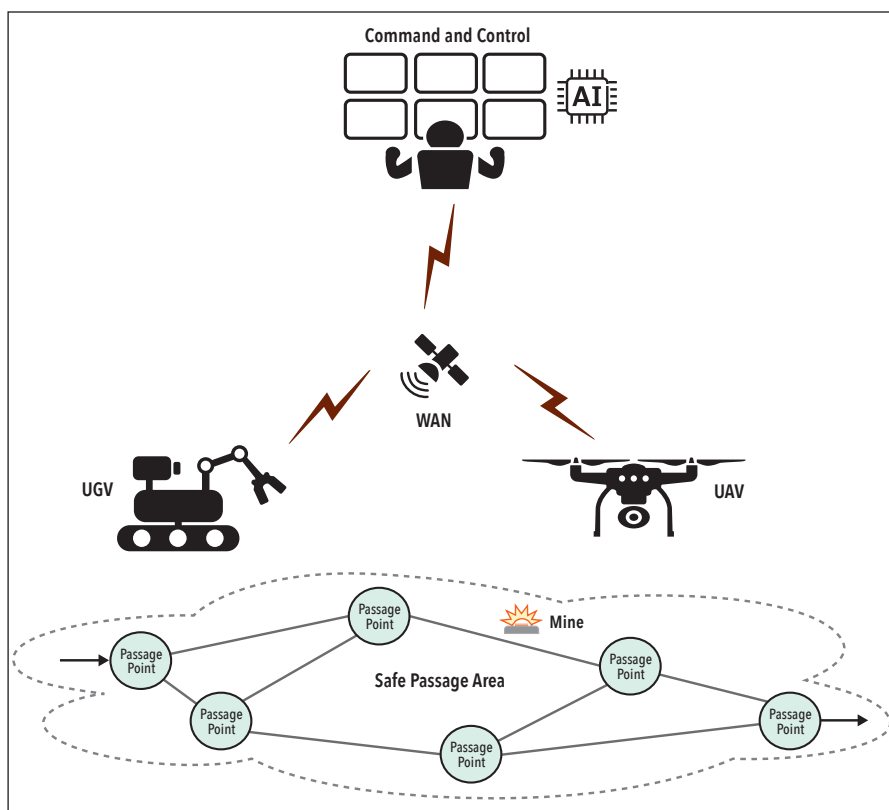


Figure 1. Overview of Operation Safe Passage. The scenario illustrates a distributed, multi-agent system in which trust must be established across sensing, decision-making, and execution under uncertainty.

- Model-based systems engineering (MBSE) platforms capable of integrating system telemetry and AI outputs
- Governance frameworks that define accountability, traceability, and oversight for AI-driven decisions.

Together, these capabilities enable not only system development, but also the continuous validation and monitoring required to sustain trust over time.

2. THE CRUCIBLE OF INNOVATION: OPERATION SAFE PASSAGE

Rather than examining trust in isolation, the trusted AI challenge engaged six student teams from SERC consortium universities in a shared, simulated mission scenario known as *Operation Safe Passage (OSP)*. In this scenario, teams were tasked with designing AI-enabled control architectures capable of guiding troops safely through mine-laden terrain using a combination of autonomous and semi-autonomous agents. The emphasis was not only on performance, but on developing system designs that could be trusted under uncertain and evolving conditions.

The scenario required coordination across three primary system elements as depicted in Figure 1:

- **Unmanned Aerial Vehicles**

(UAVs): Provided aerial reconnaissance and generated environmental estimates of potential mine locations through active sensing.

- **Mine Detection Systems:** A hybrid configuration combining an AI-based estimator with a human subject matter expert (SME). The reliability of both varied dynamically based on terrain, visibility, and weather conditions.
- **Unmanned Ground Vehicles (UGVs):** Executed navigation and hazard mitigation, physically clearing identified threats to establish a safe path for human troops.

A defining constraint of the challenge was that the core mine detection subsystems were provided “as-is,” with no opportunity for modification or retraining by the teams. Because these subsystems had been trained under conditions that did not fully align with the simulated mission environment, teams faced inherent and unavoidable uncertainty in their performance. This constraint reflects real-world integration challenges, where systems engineers must often incorporate imperfect, externally developed or commercial-off-the-shelf (COTS) AI components. As a result, teams could not rely on improving model accuracy to build trust. Instead, trust had to

AI Performance Table (% Accuracy)

| Column Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------|------|------|------|------|------|------|------|------|------|------|
| 1 | 0.95 | 0.95 | 0.95 | 0.73 | 0.95 | 0.95 | 0.73 | 0.94 | 0.95 | 0.96 |
| 2 | 0.95 | 0.95 | 0.95 | 0.70 | 0.95 | 0.73 | 0.73 | 0.95 | 0.96 | 0.96 |
| 3 | 0.95 | 0.95 | 0.94 | 0.72 | 0.70 | 0.72 | 0.56 | 0.95 | 0.96 | 0.96 |
| 4 | 0.95 | 0.56 | 0.95 | 0.70 | 0.68 | 0.56 | 0.57 | 0.95 | 0.95 | 0.96 |
| 5 | 0.56 | 0.57 | 0.68 | 0.62 | 0.94 | 0.57 | 0.95 | 0.96 | 0.94 | 0.96 |
| 6 | 0.57 | 0.57 | 0.68 | 0.64 | 0.94 | 0.57 | 0.95 | 0.96 | 0.95 | 0.96 |
| 7 | 0.96 | 0.57 | 0.70 | 0.65 | 0.94 | 0.56 | 0.57 | 0.95 | 0.95 | 0.96 |
| 8 | 0.96 | 0.96 | 0.68 | 0.65 | 0.94 | 0.56 | 0.57 | 0.94 | 0.96 | 0.96 |
| 9 | 0.96 | 0.96 | 0.69 | 0.67 | 0.70 | 0.56 | 0.56 | 0.96 | 0.96 | 0.95 |
| 10 | 0.95 | 0.96 | 0.72 | 0.95 | 0.72 | 0.56 | 0.56 | 0.96 | 0.96 | 0.95 |

Figure 2. Prediction accuracy placeholders used in Phase I to represent varying environmental conditions and system reliability. These abstract inputs enabled teams to reason about trust, uncertainty, and system performance prior to full implementation.

be engineered at the system level through architectural decisions—specifically how information was fused and presented, how decision logic accounted for uncertainty, and how roles were allocated between human operators and AI components.

To promote iterative systems engineering development, the challenge progressed through three phases of increasing realism and operational fidelity:

- **Phase I – Conceptual Design:** Teams were introduced to the scenario using abstract placeholder data. The focus was on defining trust requirements, architectural concepts, and validation strategies without immediate implementation constraints. The table of Phase I accuracy inputs provided to teams are shown in Figure 2. The rows and columns corresponded to location-specific accuracy assessments. This table allowed students to reason about uncertainty and trust before full simulation implementation.
- **Phase II – Simulation-Based Implementation:** Teams transitioned to a generalized simulation environment, enabling interactive control of UAV and UGV agents, access to environmental data, and evaluation of system behavior. This phase emphasized translating trust concepts into executable methods.
- **Phase III – Operational Realism:** The scenario introduced additional sensing capabilities and increased mission stakes, including explicit lethality. Teams were required to demonstrate coordination, robustness, and real-time risk management under more demanding conditions.

Together, these elements created a controlled but sufficiently complex environment in which trust had to be actively engineered rather than assumed.

3. EVALUATING TRUST: A SYSTEMS ENGINEERING JUDGING METHODOLOGY

To assess how effectively teams engineered trust into their systems, the challenge employed a judging methodology grounded in systems engineering practice. Following the final white paper submissions and presentations, a panel of seven judges evaluated each team's approach. The panel was intentionally composed with strong representation from industry and sponsor organizations to emphasize practical applicability alongside academic perspectives.

Evaluation was conducted using a 7-point Likert scale across twelve positively framed criteria aligned with key dimensions of AI-enabled systems engineering. Collectively, these criteria provide a concise snapshot of what the systems engineering community prioritized in developing trustworthy AI systems during this period:

1. Effectiveness of systems engineering activities and artifacts in building trust
2. Adequacy of infrastructure for validating trust
3. Identification of key workforce skills for an integrated product team (IPT)
4. Methods for integrating additional sensing to improve trust
5. Adaptation of system design to account for damage, lethality, or degraded conditions
6. Development of useful design patterns for AI-enabled systems
7. Approaches to risk-based monitoring and management
8. Use of quantitative methods to assure system performance
9. Identification of best practices
10. Introduction of novel approaches
11. Feasibility of proposed future development stages
12. Clarity and transferability of systems engineering methods.

To improve robustness and mitigate outlier effects, the scoring process mirrored methods used the trimmed mean method, wherein the highest and lowest scores for each team are removed prior to aggregation. This approach reinforced consistency across evaluations while preserving comparative distinctions between teams.

More importantly, the evaluation framework did more than rank performance—it revealed which systems engineering practices were most effective in operationalizing trust. The strongest approaches were those that made trust explicit across the system lifecycle, treated human-AI teaming as an architectural decision, connected autonomous behavior to measurable risk and confidence indicators, and validated system performance across varied operational conditions. These priorities organize the discussion that follows. Section 4.1 examines trust-integrated lifecycle engineering through the extension of the systems engineering V-model. Section 4.2 discusses human-AI teaming architectures and the allocation of authority between human and machine decision-makers. Section 4.3 focuses on reinforcement learning approaches strengthened by interpretable risk monitoring and trust metrics. Section 4.4 addresses the expansion of validation infrastructure through simulation, Monte Carlo analysis, auditable logs, and continuous monitoring. Together, these findings show that trust was no longer treated as an implicit outcome of performance, but as a measurable and engineerable system attribute. The workforce implications of applying these practices are then addressed in Section 5.

4. SYSTEMS ENGINEERING ADVANCEMENTS: INNOVATIONS FROM THE 2020S

The diverse approaches developed independently by the six participating

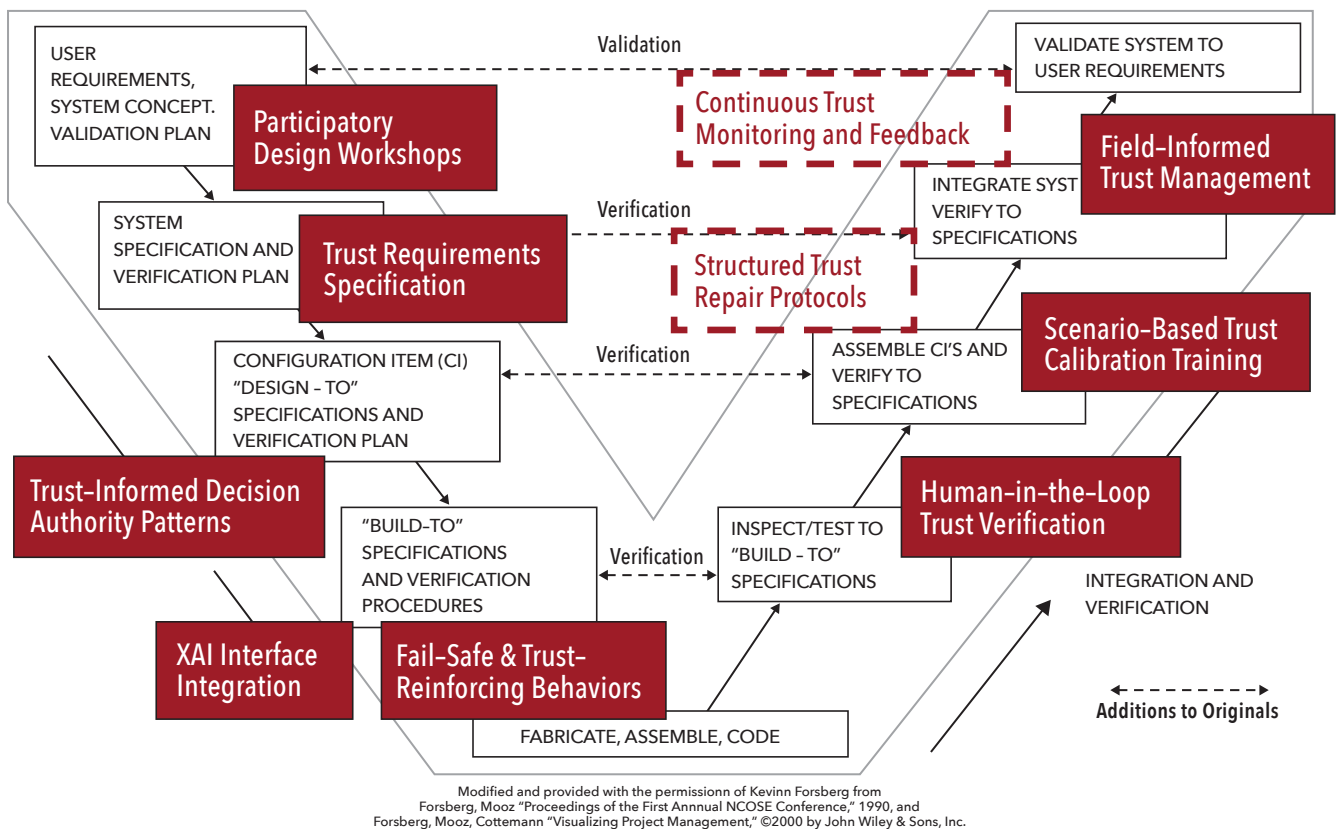


Figure 3. Extension of the systems engineering V-model to incorporate trust across the lifecycle of AI-enabled systems. Traditional SE activities are augmented with trust-specific elements—including participatory design, trust requirements specification, continuous trust monitoring, and human-in-the-loop verification—ensuring that trust is engineered, validated, and maintained alongside system performance.

universities illustrate how systems engineering processes, tools, and human-systems integration evolved over the decade to address AI-driven uncertainty. While teams operated independently and in parallel, several common themes emerged that reflect broader shifts in systems engineering practice for AI-enabled systems.

4.1 Reimagining the Systems Engineering V-Model for Trust Lifecycle Integration

A key realization of the 2020s is that trust cannot be treated as an afterthought—it must be engineered as an explicit system-level attribute from the earliest stages of development. Several teams incorporated trust directly into system requirements, including mechanisms for detecting and responding to trust degradation during operation.

Virginia Tech addressed this challenge by extending the traditional systems engineering V-model as shown in Figure 3 to embed trust considerations throughout the lifecycle, emphasizing the role of the effectiveness of systems engineering activities and artifacts in building trust. Their approach integrated human-systems

elements directly into the workflow, including trust-specific requirement specification, participatory design workshops, scenario-based trust calibration, and human-in-the-loop verification.

By formalizing activities such as **continuous trust monitoring, feedback loops, and structured trust repair protocols**, this approach ensured traceability between early design decisions and validation outcomes. In addition, the architecture incorporated dynamic role allocation between humans and AI, balancing computational efficiency with human judgment.

This extension of the V-model reflects a broader shift: lifecycle processes are no longer sufficient if they focus only on performance and functionality—trust must be explicitly engineered, verified, and maintained across all phases.

4.2 Architecting Human-AI Teaming as a Core Discipline

As AI capabilities matured, systems engineering focus shifted from simple human oversight toward sophisticated human-AI teaming. The allocation of authority between human and machine is no longer a

user interface consideration—it is a foundational architectural decision that directly shapes system behavior.

George Washington University (GWU) developed a structured human-AI architecture framework to formalize this design space. GWU's framework is summarized in Figure 4, which organizes alternative control structures according to responsibility, initiative, and access.

Their framework categorizes control relationships based on which actor—human, AI, or both—is required to complete the control loop. It further differentiates architectures based on initiative (who acts first) and control access (direct versus indirect).

Through simulation-based evaluation, GWU demonstrated that seemingly minor changes in authority allocation—such as shifting from human-over-the-loop to human-along-the-loop configurations—can produce significantly different operational outcomes and trust characteristics.

Purdue University independently explored dynamic authority allocation through comparative architecture analysis. Recognizing that both human and AI performance varied across environmental conditions, Purdue developed a

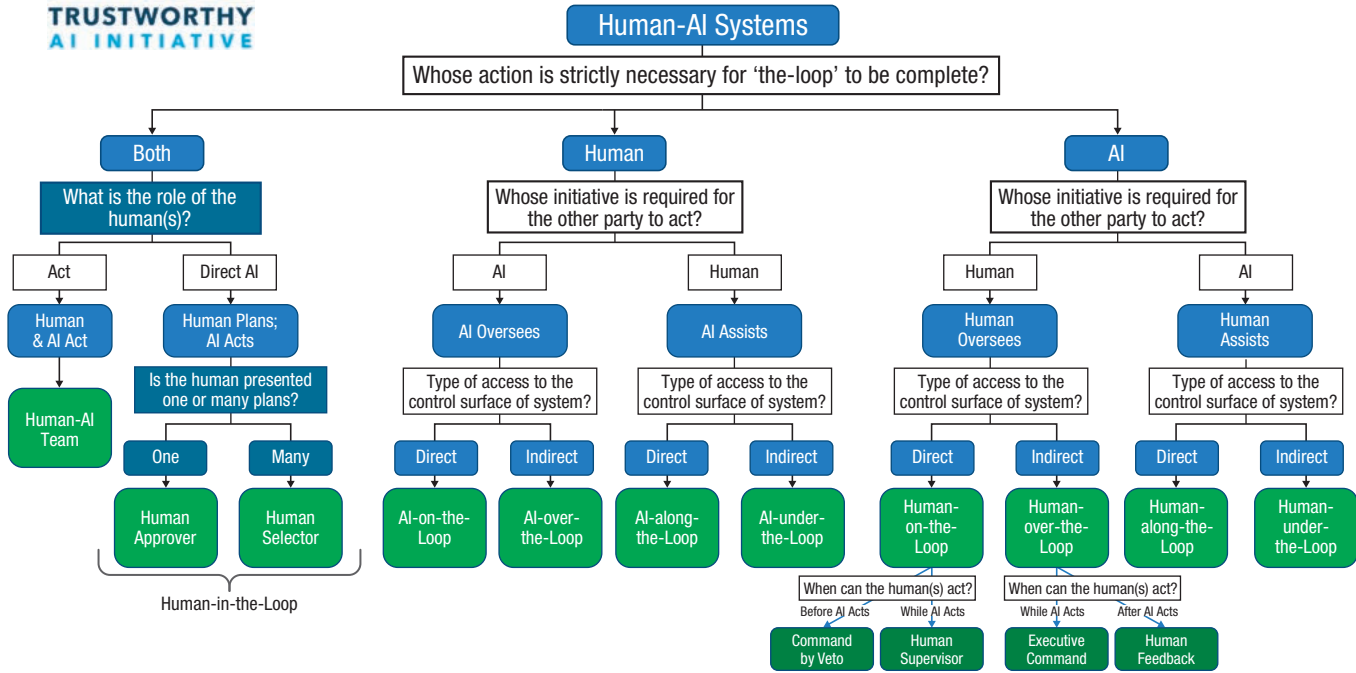


Figure 4. Human-AI architecture framework illustrating alternative control relationships between human operators and AI systems. The framework formalizes authority allocation based on control responsibility, initiative, and access, enabling systematic evaluation of how different human-AI configurations influence system behavior and trust.

reward-based exploration strategy using a multi-armed bandit framework, drawing on the systems engineering “V” illustrated in Figure 5. A multi-armed bandit framework is a sequential decision strategy in which a system repeatedly chooses among alternative options, or “arms,” while

balancing exploration of less-tested choices against exploitation of choices that have performed well in prior trials. This approach allowed the system to dynamically test and select control configurations based on observed performance. Their results produced actionable

heuristics for authority allocation. For example, human estimators were favored in complex terrains such as rocky or swampy environments, while AI components were prioritized in more predictable conditions such as sandy or wooded terrain. These findings underscore the need for flexible system architectures capable of adapting human-AI roles in response to environmental context.

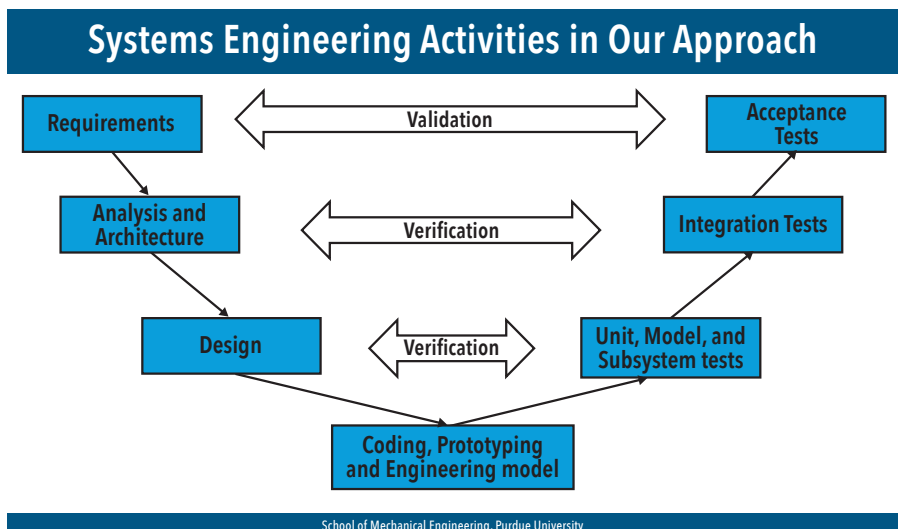


Figure 5. Dynamic human-AI role allocation framework based on environmental conditions and observed subsystem performance. By leveraging a reward-based exploration strategy, the system adapts authority between human and AI components, demonstrating the need for flexible architectures that respond to context-dependent reliability.

4.3 Reinforcement Learning Integrated with Risk Monitoring
Reinforcement learning (RL) emerged as a dominant approach for decision-making under uncertainty across the challenge. RL enables systems to adapt through environmental feedback, learning from rewards and penalties in dynamic environments. However, RL alone presents challenges for trusted systems engineering because learned policies can be difficult to interpret. An RL policy may select effective actions without providing a readily understandable rationale for why a specific action was chosen in a specific state. This opacity complicates verification, operator trust calibration, and risk assessment, particularly in safety-critical settings where users must understand not only what the system did, but why it did it.
The University of Virginia (UVA), one of the top-performing teams, addressed this

We frame trust in AI through the definition: “reliance upon others, confidence in their abilities, and consistency in behavior.”¹

Trust Framework

- System functionality requires human dependence/confidence and reliable performance
- Trust is established through AI’s ability to deliver **accurate, effective, and consistent** results
- Past performance data is utilized to improve reliability, and explainable statistics give real time risk monitoring

Tradeoff Between Speed and Trust

- Tradeoffs between time efficiency, accuracy, and user confidence must be considered for the system to support mission success.
- Using reinforcement learning provides a middle ground solution
- Risk is inevitable when there is a degree of uncertainty

Transparent and explainable statistical models are the foundation for consistency and predictability to create trust

[1] Department of the Army, “Apd 6-22: Army leadership and the profession,” Jul 2019. [Online]. Available: https://rtp.fas.org/doddir/army/adp6_22.pdf

Figure 6. Operational definition of trust used to guide reinforcement learning and system behavior. Trust is characterized in terms of reliability, confidence, and behavioral consistency, providing a foundation for integrating measurable trust metrics into decision-making processes.

limitation by tightly integrating RL with explainable statistical models and real-time risk monitoring. UVA defined trust in AI in terms of reliability, confidence, and consistency of behavior, underscored in Figure 6.

To operationalize this definition, UVA designed a reward function that incorporated distance to goal, scanning behavior, and hazard encounters, while being governed by explicitly defined trust metrics such as **accuracy avoidance ratio (AAR)** and **surrounding scan confidence (SSC)**. These metrics translated operational trust requirements into measurable system behaviors, allowing the RL policy to balance tunable tradeoffs between speed, safety, and confidence. AAR measured the percentage of UGV

moves into low-accuracy cells, defined as cells where mine detection accuracy fell below 50 percent. A lower AAR indicated that the system was successfully avoiding areas where the underlying detector was unreliable. SSC measured the average predicted accuracy of cells scanned around the UGV, providing a local measure of how well the system understood the reliability of mine detection in the UGV’s immediate operating area. The effect of UVA’s reinforcement-learning-based control policy is illustrated in Figure 7.

Similarly, Old Dominion University (ODU) focused on improving transparency in AI decision-making. Their approach centered on a modular simulation framework that generated explicit confidence scores

based on statistical analysis and terrain-dependent performance adjustments. Confidence scores are a proxy for transparency and explainability, but not sufficient by themselves. By providing visualization tools that exposed the AI’s internal confidence prior to action execution combined with the terrain context, ODU reduced the risk of automated overconfidence and improved operator trust.

These approaches demonstrate a broader trend: high-performing AI systems must be coupled with interpretable metrics and monitoring mechanisms to support trust in operational contexts.

4.4 The Expansion of Validation Infrastructure

Traditional validation methods are insufficient for AI-enabled systems operating in uncertain and dynamic environments. The 2020s saw a significant shift toward advanced simulation-based validation and digital engineering ecosystems.

Stevens Institute demonstrated the effectiveness of this approach through their Silverfish Safe Passage System. Their architecture combined advanced path planning algorithms with large-scale Monte Carlo simulation to evaluate performance across thousands of randomized scenarios.

This approach enabled empirical validation of AI performance relative to human-only baselines, demonstrating improved efficiency and resilience under high-risk conditions. More importantly, it provided a scalable method for evaluating system behavior across a wide range of operational contexts.

Stevens also emphasized that trust validation requires more than performance testing. Their system incorporated transparent, auditable logs that allowed human operators to review AI decisions post-mission, reinforcing accountability and supporting trust in system behavior.

Collectively, these approaches reflect a major shift in systems engineering practice: validation is no longer a static activity, but an ongoing, data-driven process supported by simulation, digital twins, and continuous monitoring.

5. SHAPING THE MODERN SYSTEMS ENGINEERING WORKFORCE

As systems engineering practices evolved to address AI-enabled systems, the required workforce competencies evolved in parallel. Developing, integrating, and sustaining these systems at scale demands a diverse integrated product team (IPT) with expertise that was traditionally distributed across separate disciplines.

The trusted AI challenge highlighted that the modern systems engineer must possess

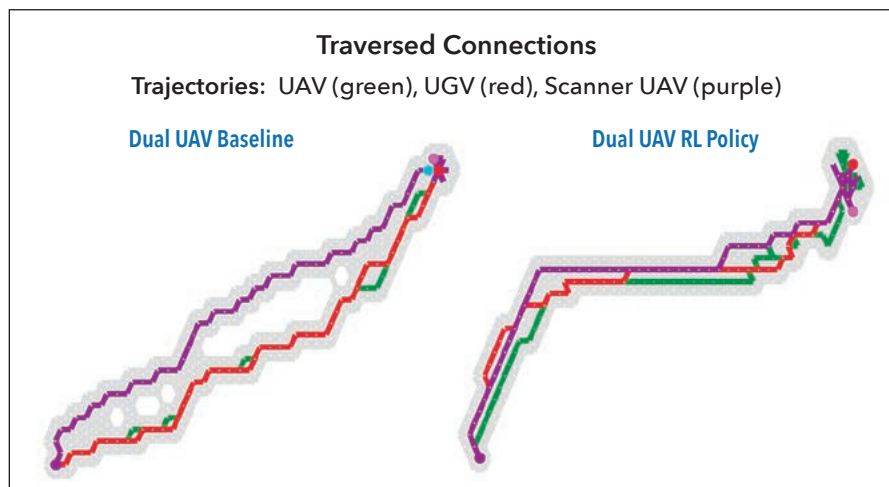


Figure 7. Comparison of navigation trajectories using baseline and reinforcement learning-based control policies. Reinforcement learning, when combined with trust-aware metrics and real-time risk monitoring, enables improved path efficiency while maintaining safety and user confidence under uncertainty.

AI systems need effective infrastructure and workforce skills/abilities to be mission successful and validate trust.

Infrastructure

- Physical Infrastructure
 - Hardware and testing/operating centers are needed to develop and maintain the system
- Digital Infrastructure
 - Includes data pipeline, cloud computing, and Model-Based Systems Engineering (MBSE) platforms to design, process, and collect data
- Governing Infrastructure
 - Establish accountability and ensure that AI systems are developed and operated responsibly

Workforce Skills and Abilities

- A diverse team and set of skills are required for an integrated product team (IPT) to develop and manage AI enabled systems
 - Systems and AI/ML Engineers
 - Data Scientists
 - Human Factors Specialists
 - Risk Analysts
 - Cybersecurity and Operational Experts
- DEVCOM can utilize existing training and certification programs

Developing and sustaining AI-enabled systems at a large scale requires a combination of infrastructure and workforce at an increasing demand but can leverage existing resources



Figure 8. Workforce readiness and skill requirements for developing trusted AI-enabled systems, from UVA team. Effective AI-enabled systems require integrated teams that combine traditional systems engineering with expertise in AI/ML, human factors, and risk management.

not only technical depth, but also cognitive agility and cross-disciplinary fluency. Teams that performed well were those that effectively integrated expertise across domains, combining traditional systems engineering with capabilities in machine learning, data analysis, human factors, and risk management. UVA's approach provides a concrete example of this integration. Their AAR and SSC metrics required the team to translate operational safety expectations into RL reward requirements, then map those requirements back to measurable system behaviors that could be interpreted by operators and evaluated during simulation. Their approach required more than model development alone but was based in the ability to connect AI behavior, statistical confidence, operational risk, and systems engineering artifacts into a coherent trust argument.

Both UVA and ODU emphasized that effective teams must blend:

- Systems engineers responsible for architecture, integration, and lifecycle management
- AI/ML engineers and data scientists responsible for model development and performance analysis
- Human factors specialists to design effective human-AI interaction and support trust calibration
- Risk analysts to evaluate uncertainty, failure modes, and operational tradeoffs.

UVA's view of workforce readiness for trusted AI-enabled systems is summarized in Figure 8.

This shift reflects a broader change in the role of the systems engineer—from coordinating well-defined subsystems to

orchestrating adaptive, data-driven, and human-centered systems. In this context, the systems engineer serves not only as an integrator, but also as a translator across disciplines, ensuring alignment between algorithmic behavior, human expectations, and mission objectives.

Ultimately, the evolution of the workforce reflects the broader transformation of systems engineering itself. As AI-enabled systems become more complex and adaptive, success depends not only on technical innovation, but on the ability of multidisciplinary teams to collaboratively engineer, evaluate, and sustain trust throughout the system lifecycle.

6. LOOKING AHEAD: IMPLICATIONS FOR 2030 AND BEYOND

The progression of the trusted AI challenge provides a useful microcosm of how systems engineering has evolved over the past decade. By enabling multiple teams to explore diverse approaches in parallel, the competition demonstrated that structured, phased experimentation can generate a richer and more transferable set of design patterns than isolated efforts. This model reflects a broader shift in the discipline toward collaborative exploration and iterative learning in the face of rapidly advancing technologies.

Looking ahead to 2030 and beyond, several key implications for systems engineering practice have emerged:

1. Trust Engineering as a Standard Lifecycle Activity

Trust is no longer an implicit outcome of system performance—it must be explicitly engineered. This includes defining trust requirements,

integrating trust considerations into system architecture, and maintaining trust through continuous runtime monitoring and feedback across technical and operational domains.

2. Human-AI Teaming as a Core Competency

Human operators remain essential, providing contextual judgment and oversight that AI systems cannot fully replicate. Designing effective human-AI collaboration—including authority allocation, information flow, and dynamic role adaptation—has become a central architectural responsibility.

3. Validation Through Digital Ecosystems

Traditional testing approaches are insufficient for AI-enabled systems operating under uncertainty. The increasing reliance on simulation-based environments, digital twins, and scenario-driven validation reflects the need to evaluate system behavior across a wide range of conditions that may be impractical or unsafe to replicate physically. This is especially critical for human-system concerns such as human-in-the-loop, human-on-the-loop, and related human-system interactions.

4. Systematic Risk Management for AI-Enabled Systems

Managing risk in AI-enabled systems requires structured, lifecycle-integrated approaches. Frameworks such as the NIST *Artificial Intelligence Risk Management Framework* (AI RMF) provide a foundation for identifying, analyzing, and mitigating risks (NIST 2023). In practice, this includes implementing real-time anomaly detection, monitoring trust degradation, and establishing recovery mechanisms to ensure resilience under adverse conditions.

The trusted AI challenge also reinforced a broader conclusion: engineering trust in AI is not solely a computer science problem—it is fundamentally a systems engineering challenge. By requiring teams to balance safety, explainability, and performance in partially observable, adversarial environments, the challenge catalyzed the development of actionable systems engineering methods.

As AI-enabled systems continue to expand their autonomous footprint, the margin for operational error continues to shrink. Systems engineers remain the critical bridge between advanced computational capabilities and safe, effective real-world operation. The methodologies that emerged

over the past decade—including trust-integrated lifecycle models, dynamic human–AI architecture frameworks, and reinforcement learning approaches augmented with risk monitoring—provide a foundation for ensuring that the autonomous systems of the future are not only highly capable, but also appropriately trusted.

Taken together, these developments signal a defining shift for the discipline: systems engineering is no longer solely about integrating components—it is about integrating intelligence, uncertainty, and human judgment into cohesive, trust-worthy operational systems. ■

REFERENCES

- Choung, Hyesun, Prabu David, and Arun Ross. 2023. “Trust in AI and Its Role in the Acceptance of AI Technologies.” *International Journal of Human-Computer Interaction* 39 (9): 1727–1739.
- Passi, Samir, and Mihaela Vorvoreanu. 2022. “Overreliance on AI: Literature Review.” *Microsoft Research*.
- Robinette, Paul, Wenchen Li, Robert Allen, Ayanna M. Howard, and Alan R. Wagner. 2016. “Overtrust of Robots in Emergency Evacuation Scenarios.” In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 101–108. IEEE.
- National Institute of Standards and Technology (NIST). 2023. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. Gaithersburg, MD: NIST.
- Systems Engineering Research Center (SERC). 2025. “Trusted Artificial Intelligence (AI) Systems Engineering Challenge.” Technical Report. September 29, 2025.

INCOSE Career Center



Scan QR code to visit www.careers.incose.org

The INCOSE Career Center connects top talent with leading organizations in the field. Explore job listings tailored for systems engineering professionals, post your résumé, and find the perfect role to match your skills and aspirations.

Whether you're a job seeker or recruiter, the INCOSE Career Center is your gateway to opportunities in the systems engineering community.

Connecting talent with opportunity



Designing Human-AI Architectural Robustness to Inherently Brittle AI

Aditya Singh, asingh25@gwu.edu; and Zoe Szajnfarber, zszajnfa@gwu.edu | The George Washington University
Copyright ©2026 by Aditya Singh and Zoe Szajnfarber. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Machine learning based systems have made impressive advances in recent years, but they are still brittle, often failing when encountering new and unexpected changes to the operating environment. Humans are often partnered with AI-enabled systems to negate this brittleness, but human operators may struggle with changes to the scale of the operating environment, as cognitive fatigue can decrease the effectiveness of human oversight. Through a simulation of a notional minefield traversal system, we show how humans and AI are integrated together affects system outcomes. We propose that evaluations of human-AI systems should consider different human-AI integration architectures and that AI-enabled systems should also deploy continuous monitoring to detect shifts in the operating environment which may cause AI system performance to falter. Finally, system design should enable human operators to have sufficient situational awareness, which may require scaling up the number of human operators available.

INTRODUCTION & MOTIVATION

Environmental uncertainty is a reality of complex system design. Significant attention has been paid to architecting traditional systems that are either robust, meaning that they sustain value despite changes during operations, or changeable, meaning they can change to maintain value in new environments (Fricke and Schulz 2005). With the rise of AI-enabled systems (AIES), many of the principles developed around robustness and changeability in traditional complex engineered systems must be reevaluated because the ways in which AIES fail and the mechanisms to mitigate the impact of those failures are inherently different. To that end, this work models how changes to scale and environment might affect humans and AI differently and shows that the ways in which humans and AI are paired together can have significant effects on how robust a system is to different types of change. These findings have implications for the way that humans and AI are partnered together, how models of human integration need to

change for human-AI systems to deal with issues of scale, and the technical measures that could be deployed to help AIES deal with changes in the operating environment.

LITERATURE REVIEW

New Kinds of Brittleness

Machine learning approaches that have enabled modern advanced AI systems, like computer vision systems, are known to be brittle to natural variation, new environments, and adversarial manipulation. Natural variation occurs when observed data in deployment does not match training and testing data. This might occur if training data was not representative of the population of interest, if the population itself changes, or if data collection methods change. Many types of frontier AI models experience significant performance losses when dealing with natural variation (Taori et al. 2020; Koh et al. 2021). The issues that natural variation pose to machine learning are also applicable if AIES are deployed to new environments, especially if the new

environment does not closely match the environment the model was originally trained for (Hendrycks et al. 2021). This means that the data used to train the model should closely match the data the model uses during deployment, rather than assuming that a model can perform the same type of task in different environments. Adversarial changes occur when other parties attempt to trick AI systems. A variety of adversarial techniques have been shown to trick computer vision and large language model systems (Wei et al. 2024; Souly et al. 2025). Models that score well on internal testing and benchmark evaluation often see extreme performance drops in deployment due to natural variation, being deployed to new environments, or adversarial manipulation (Yu et al. 2022; Bolocan et al. 2025). Regardless of the reason that models see performance drops, the inherent brittleness of AIES can render systems inoperable in the face of such change and requires human operators to step in and deal with issues that AIES cannot address.

Table 1. Classification architectures

| Architecture | Description |
|----------------------|--|
| Human Only | Human makes a prediction and outputs the classification. |
| AI Only | AI makes a prediction and outputs the classification. |
| Human-AI Team | Based on an initial screening for terrain type, if terrain is {grassy, rocky} human makes prediction and outputs classification; if terrain is {swampy, sandy, wooded} AI makes prediction and outputs classification. |
| Human Approver | AI makes prediction and reports confidence in classification; Human assesses if confidence meets or exceeds human's dynamic threshold to accept. If threshold is met, human approves classification; if not, the human redoes the classification task themselves and reports their classification. |
| AI-along-the-loop | Human classifies and assesses their own confidence. If their confidence in their classification is below the static threshold value, they delegate to AI. If they both disagree, the classification made at a higher confidence level is used. |
| Human-along-the-loop | AI classifies prediction and assesses its own confidence. If its confidence is below static threshold, it delegates prediction to a human expert. If they both disagree, the classification made at a higher confidence level is used. |
| AI-on-the-loop | Both human and AI make a prediction. In cases of disagreement, if AI's confidence is 5%+ higher than the human's confidence, the AI classification is reported; otherwise, the Human's is reported. |

Table 2. Routing architectures

| Architecture | Routing |
|-------------------|--|
| AI Only | Expected traversal time of each road is calculated, and the system selects the lowest expected traversal time road that still makes progress towards the end node.
$E[T_{Travel}] = T_{Traverse} * C_{clear} + (T_{Traverse} + T_{clear}) * (1 - C_{clear})$ |
| Human Selector | System ranks available roads based on safety (classified as clear at highest confidence) and progress towards the end goal and combines the ranks into a total rank. The human selector selects the top combined rank road and selects the safest road if there is a tie. |
| Executive Command | Executive command models the ability to change AI Only routing behavior dynamically. It uses AI Only as its baseline, but if an IED is encountered, it enters cautious mode in which the system selects the highest confidence clear road that still makes progress towards the end node. If the traversal is taking twice as long as expected (calculated based of the expected time of human classification and traversal for the initial optimistic path), it enters bold mode where it selects the maximum progress making road among clear roads. |

Architecture as a Potential Means for Mitigating Brittleness

In prior work, we outlined how there are many ways humans and AI can partner together, defining human-AI architecture as the task allocation and relationship between humans and AI (Singh and Szajnfarber 2025). Few studies have tested how different human-AI architectures respond to change. One study compared two types of human-AI collaboration against a human-only baseline for UAV traversal (Griner 2012). In the human-only system, the UAV followed a path using waypoints set by the human. The first human-AI collaboration architecture allowed humans to work with an algorithm to generate paths by setting general way areas that the UAV should pass through. The second collaboration architecture had the algorithm

generate the path automatically but allowed the human to edit waypoints (Griner 2012). The first architecture performed the worst in a variety of environments. The second architecture showed similar performance to the baseline but improved as the difficulty and complexity of the environment increased (Griner 2012). In another study, AI alone performed better than both humans alone and a human-AI team on detecting fake reviews (Cabrera et al. 2023). However, the same architecture performed better than both humans and AI alone on a bird classification task, suggesting that architecture performance is dependent on the task (Cabrera et al. 2023). These two studies highlight the importance of testing human-AI architectures across both scale and environmental shift challenges. Some authors have suggested that AI models be

tested for their ability to deal with changes in the operating environment (Koh et al. 2021; Hendrycks and Dietterich 2019). We extend these calls by pushing for human-AI architectures, rather than just the models themselves, to be tested for their ability to respond to changes in the environment.

METHODOLOGY

Simulation Structure

Leveraging a simulation developed as part of a Systems Engineering Research Council challenge sponsored by the U.S. Army Armaments DEVCOM Center (Meno et al. 2025; Beling et al. 2024), we test how 21 human-AI architectures respond to changes in the operating environment of a notional unmanned minefield traversal system. In our notional system, humans or AI can predict whether the road

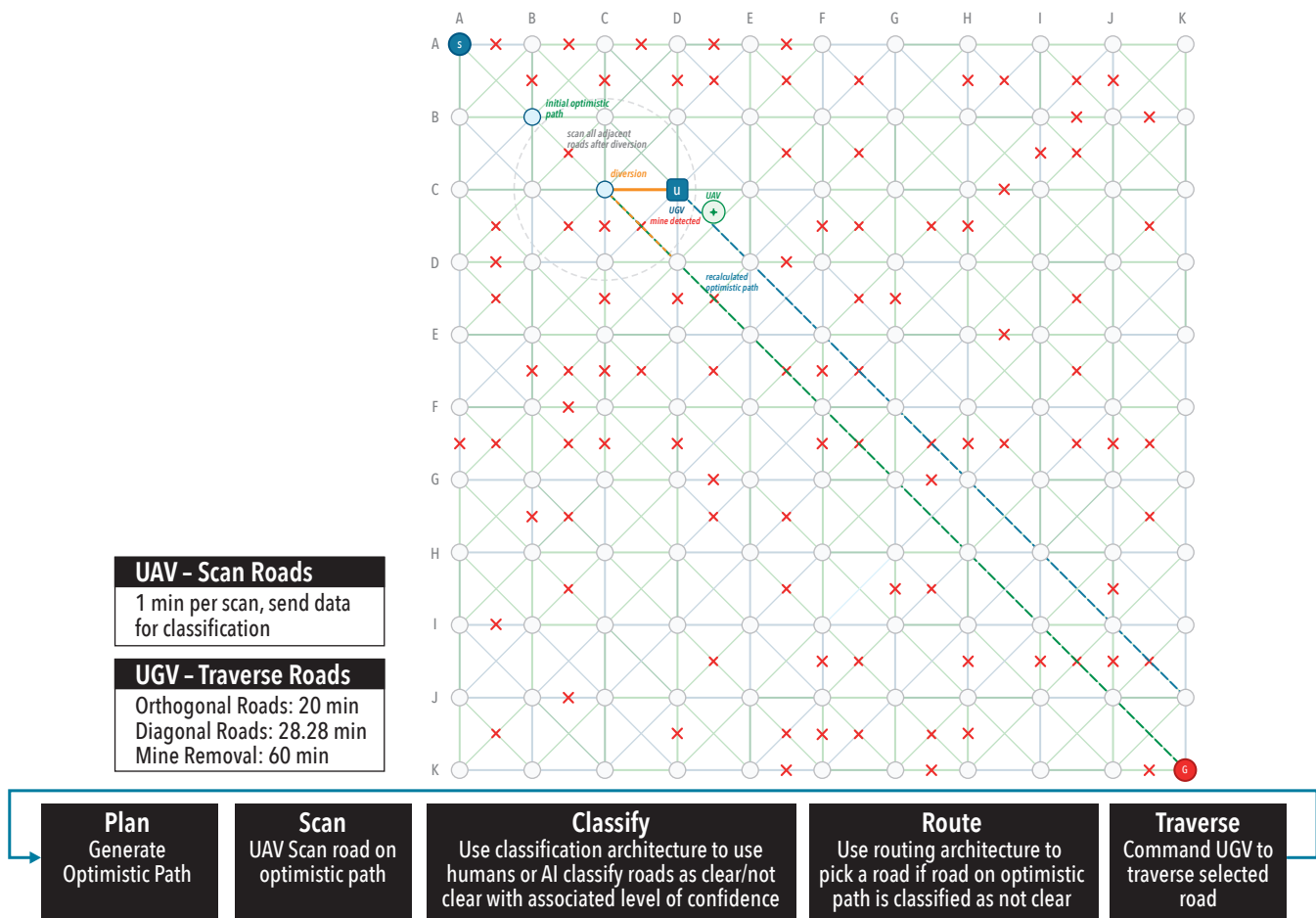


Figure 1. Simulation overview

ahead is clear or has a mine present. The different architectures for this classification task are described in Table 1. The system generates an optimistic path, which is the shortest route from the current node to the end node assuming there are no mines. We initially only classify roads on the optimistic path. If roads are classified as not being clear, we classify every adjacent road to the node we are at, and three different architectures are modeled to determine different ways of how humans and AI would decide which alternative road to take as described in Table 2. An overview of the simulation is presented in Figure 1. Note that if an IED is hit, the penalty is additional time to dismantle rather than mission failure or vehicle loss.

DESIGN OF EXPERIMENTS

The goal of this work is to assess how different human-AI architectures respond to changes in environment (where humans tend to be more robust) and scale (where AI tends to be more robust). These dimensions of change are operationalized here in terms of the three variables that define the environment of the simulation: IED density, map size, and terrain composition.

Table 3. Classification performance bounds

| | AI Accuracy Range | Human Accuracy Range |
|--------|-------------------|----------------------|
| Wooded | 55-75% | 65-75% |
| Grassy | 65-75% | 75-85% |
| Swampy | 60-70% | 65-70% |
| Sandy | 80-85% | 70-80% |
| Rocky | 55-75% | 70-85% |

IED density is a measure of the fraction of edges where a mine is present. Map size is a count of the number of nodes in the network, with links being available between all nodes, including diagonal connections. Map size and IED density change the scale of the problem but do not change its fundamental underlying nature.

Mechanisms of Human and AI Degradation

We expect that humans would struggle to cope with changes to scale as longer or more difficult missions might cognitively burden humans, degrading their performance over time. Terrain composition changes the underlying challenge for both

human and AI classifiers, but we expect that humans are better able to deal with a variety of environments. We modeled the human and AI’s classification accuracy based on terrain type, as shown in Table 3, to reflect humans being less variable in their performance both within a given environment and across environments, while AI was far more variable both within an environment and across environments. This is meant to model AIES’ ability to operate above human level performance in narrow circumstances while being quite brittle to even minor shifts within the environment and being brittle to changes in environment.

Map Size Comparison (Mixed, 25% IED, Decay Enabled)

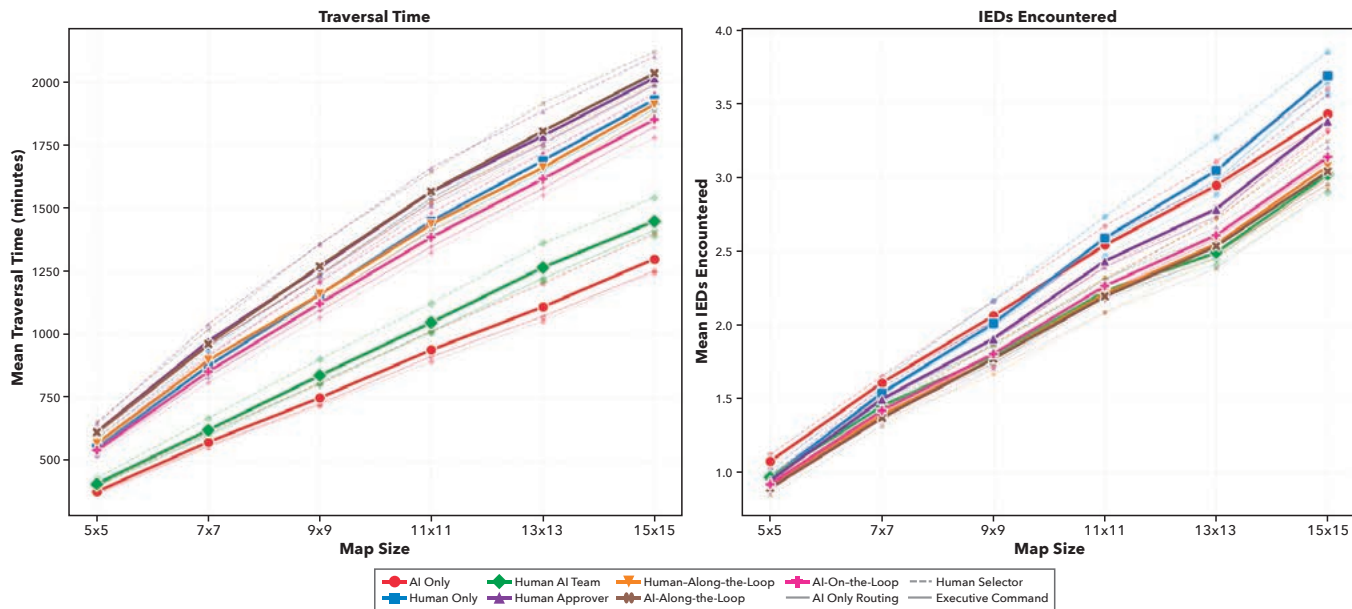


Figure 2. IEDs hit across map size

In real deployments of human-AI systems where human behavior affects system outcomes, we might expect humans to succumb to fatigue and have their vigilance fall as the scale of the problem increases (Warm et al. 2008; Wohleber et al. 2019). Fatigue might arise because humans are cognitively overloaded with tasks or because they are bored (Desmond and Hancock 2000). Both causes of fatigue present challenges to how humans can deal with issues of scale. Fatigue and drop offs in vigilance should not be taken as guaranteed as there is a body of science dedicated to assisting humans in maintaining vigilance and situational awareness (Hancock 2026), but the aim of this paper is to show that fatigue can degrade human-AI architecture performance even when the technical specifications and interaction patterns remain fixed. To model how human fatigue might manifest, we modeled a slight degradation of human accuracy by dropping the lower end of the human accuracy range by 0.3% and the upper range by 0.2% with every decision the human makes, making them less accurate and more varied as time goes on. This equates to a 0.5% cumulative penalty per decision, but the total penalty is capped

at 35%. Additionally, we modeled humans being more willing to delegate to AI or to approve AI classification as time goes on by decreasing the relevant thresholds by 0.5% with every human decision made, signaling the desire to offload more cognition as vigilance falls. Together, the simulation captures the potential brittleness of AIES and the cognitive limitations that humans face.

Changes in Scale: To model changes in scale, we varied the map size and enabled human vigilance and attention decaying over time. We present six map sizes with 25% IED density throughout: 5x5, 7x7, 9x9, 11x11, 13x13, 15x15. We conducted 5000 runs per environment. This experiment seeks to understand if the shifts in performance and risk we expect to see as map size increases occur uniformly across architectures.

Changes in Environment: To model changes in the nature of the environment, we tested uniform terrain maps for each terrain type, meaning the maps were entirely wooded, sandy, grassy, swampy, or rocky, along with the mixed terrain baseline map. The goal of this experiment is to study how changes in the underlying environment affect the performance and risk profile of the human-AI architectures.

RESULTS

Effect of Changes to Scale

Figure 2 plots the traversal time and IEDs hit across the various map sizes for each architecture. In the smaller map sizes, there is a tight clustering of architectures in terms of IEDs hit. The primary tradeoff is that while humans are slow at classifying, they should be more reliable and less variable in their performance. Meanwhile, AI is fast but more varied in its performance. However, as map sizes increase and humans grow more fatigued, Human Only, indicated in blue, and Human Approver, indicated in purple, classification architectures see their risk profiles worsen, while AI Only classification architecture, indicated in red, remains steady in how it is affected by increases in map size. While Human Only and Human Approver classification architectures proved to be sensitive to changes in human performance, Human-AI Team was fairly robust to changes in map size, and the remaining three classification architectures were not nearly as sensitive as Human Only and Human Approver. These results show that how humans and AI are partnered together can mitigate or exacerbate human fatigue. Our simple model of human fatigue

Table 4. Difference in differences testing across terrain

| | Grassy | | Rocky | | Sandy | | Swampy | | Wooded | |
|---|-------------|------------|-------------|------------|-------------|-------------|-------------|------------|------------|------------|
| | Time | IED | Time | IED | Time | IED | Time | IED | Time | IED |
| <i>Pairs that are Statistically Significant and have at least a small effect size</i> | 135 (64.3%) | 71 (33.8%) | 135 (64.3%) | 92 (43.8%) | 161 (76.7%) | 146 (69.5%) | 137 (65.2%) | 26 (12.4%) | 52 (24.8%) | 38 (18.1%) |

and attention decay did not include humans slowing down or being unavailable, which would worsen how negatively changes in scale affect human operators.

Effect of Changes to the Nature of Problem

Changing the terrain composition of the map changes the underlying nature of the problem given how different classification accuracy is across terrains. We wanted to test whether or not differences in human-AI architecture led to differences in how system risk and performance changed when the nature of the environment changed. We found that architectures responded differently to changes in terrain composition. Table 4 shows the results of a Welch's t-test used to determine the difference in differences for each architecture going from the mixed terrain map to a single terrain map, tested at the 0.01 level of significance with Bonferroni corrections. In all but the wooded terrain, at least 60% of pairs had statistically significant difference in differences in traversal time with at least a small effect size as measured by Cohen's *d*. In all but the swampy and wooded cases, at least a third of pairs had statistically significant difference in differences in IEDs encountered with at least a small effect size as measured by Cohen's *d*. Figure 3 shows the performance and risk of each architecture across the five single terrain maps and the mixed terrain map.

Human Only and AI-along-the-loop classification architectures were the most robust in terms of differences in performance and risk across the single terrain maps, while Human-along-the-loop and Human Approver architectures were the most sensitive. Human-AI Team architectures which achieve high performance and risk mitigation by delegating classification to whichever classifier has a comparative advantage, were highly variable to environmental shift, because there were no multi-terrain maps that they could apply their comparative advantage logic to. By collapsing environments to be the same terrain, the effectiveness of Human-AI Team was eliminated.

The human-AI architectures that dominate in one terrain type may not be the optimal choice in another terrain, and we found that the architectures that were the most robust, Human Only and AI-along-the-loop, were not the ones that were the universally the fastest or safest across terrains. Instead, different human-AI architectures created tradeoffs in terms of specializing in one type of environment versus being robust to a variety of environments. This means that the choice of which human-AI architecture to select is highly dependent on the expected operating environment. If the expected operating

environment is well-understood ahead of time, designers may be able to deploy the optimal human-AI architecture for that environment, but if there is uncertainty in the potential operating environments, designers may choose to deploy a human-AI architecture that is more robust to a variety of environments.

DISCUSSION & CONCLUSION

We set out to understand how different human-AI architectures responded to changes in the environment. We modeled one case of scale change and one case of change in the nature of the problem. We found that human-AI architectures responded differently to the types change imposed, in a more complex way than a 'humans are better at, machines are better at' framework might predict. Architectures that rely on humans doing the classification or oversight struggled to deal with increased scale as human cognition declined. Conversely, architectures that rely on AI doing the classification were generally the least robust in terms of IEDs encountered when switching environments. Surprisingly, some architectures that are intended to leverage the strengths of both humans and AI, like Human-along-the-loop were the least robust in terms of performance and risk when environments where changed. In total, we found that different architectures that rely more heavily on humans vs. AI respond differently to different changes. To understand how to address this issue, we discuss the role of human-AI architecture in dealing with changes to the environment, the technical measures being deployed to address AIES brittleness, and the actions needed to mitigate human fatigue.

Human-AI Architecture as a Response to AI Fragility

Analyzing both scale and nature changes, we show that certain human-AI architectures may be better suited to some environments than others. This creates the opportunity for either system robustness or system changeability. For example, higher fidelity versions of the type of modeling and simulation shown in this paper may reveal that one architecture, such as a Human-AI Team, yields acceptable performance across a range of expected operating conditions even if it is not dominant in any. Such a human-AI architecture would create a relatively robust system, even if the model itself sees its performance change significantly. In the inverse situation, human-AI architecture changes are likely to be far easier to implement than changes to an AI model. If pre-deployment testing and evaluation of different human-AI architectures can reveal which architectures or

family of architectures is likely to perform best in different operating environments, human-AI architecture could change to match the environment. This may enable a system to be changeable even if the AI model itself is not.

Learning how to better partner humans and AI together may create better long-term outcomes faster than research on improving AI model robustness or changeability can deliver on its own. There is currently not well-developed theory or empirical analysis to draw on to understand the facilitators or limitations of how human operators can easily switch architectures to match environments. Early longitudinal studies seem to suggest that users of LLMs match different interaction patterns with associated desired outcomes, seemingly using intuitive knowledge of the system's capabilities developed over time (Ammari et al. 2026; Long et al. 2024). LLMs provide both a wide range of capability through their use of software tools and adaptability through their ability to take in feedback through natural language. Developing systems that are broadly capable and easily adaptable by end users may be key design features of AI-enabled systems that can change human-AI architectures quickly and effectively in the field.

Continuous Monitoring

The findings that architectures' performance and risk shift greatly when environments shift signifies a need to test human-AI architectures across expected operating environments to discover any potential tradeoffs between robustness across environments and specialization within environments in the manner that this testbed does. Additionally, the simulation results support that idea that shifts in context need to trigger retest. Context shifts may not be as readily detectable as switching operating environments. Subtle changes in the population of interest over time can change how AI models perform without such changes being obvious to users. There is ongoing research around how to monitor shifts that cause performance loss in machine learning AIES (Koh et al. 2021; Koch et al. 2024; Hendrycks and Dietterich 2019), but these methods are not yet fully mature and may struggle to identify all relevant shifts that can degrade AIES performance. Machine learning vulnerability to adversarial manipulation is also especially concerning in safety critical domains like transportation and defense. These issues are why humans should remain partnered with AI systems for the foreseeable future and are also why some form of environmental monitoring is vital to guard against AIES brittleness. In our simulation, the clearest analogous

Effect of Terrain Type on Architecture Performance

(Ranked by Average of Time + IEDs Sensitivity Rank: Most Robust at Top)

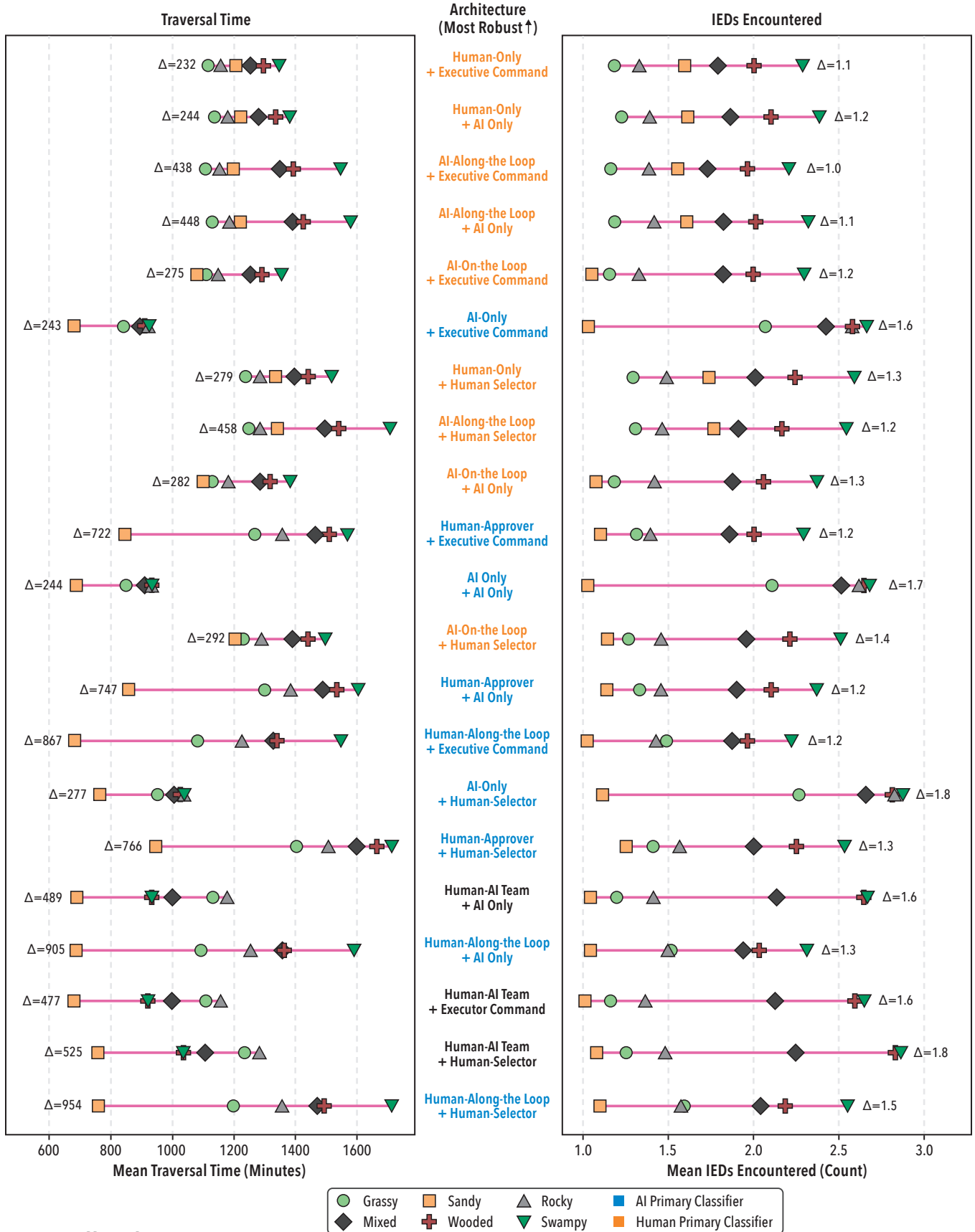


Figure 3. Effect of terrain composition

case to this is testing for some of the terrain types and then deploying the system to a terrain type not originally tested for. Identifying shifts was relatively easy in our model, but across a variety of systems, work should be done to identify parameters that might shift during deployment to understand what changes might need to trigger re-test. If those parameters shift during deployment, the performance-risk profile of a system may degrade significantly.

Situational Awareness & Scaling Up

Along with testing for context shifts, test and evaluation should include analysis of how well human operators can perform oversight tasks if facing an edge case scenario. Human operators face tremendous difficulties in maintaining vigilance when asked to oversee or partner with AIES for extended periods of time. Lessons from automation suggest that situational awareness of the task at hand and of the capabilities of the system are crucial for human operators to perform the way we expect them to (Endsley 2023). AIES may pose new challenges due to their more

opaque and stochastic nature. We modeled a case of perfect human and AI calibration where confidence was exactly equal to accuracy. Even in this idealized case, we saw that human-AI architectures often struggle to cope with changes in environment. In the real world, AI models are often convincingly wrong, over projecting confidence even when presented with nonsensical problems (Nguyen et al. 2015). Large language models, for example, have been shown to present falsified reasoning to users (Scheurer et al. 2024) and high performance on benchmarks might be due to spurious correlation or reward hacking (DeGrave et al. 2021), which may lull users into a false sense of security. Proper training on how the relevant AIES operate and can fail will be vital for helping humans calibrate their expectations of the AIES they are intended to oversee and partner with.

Additionally, new staffing models that ensure there are enough situationally aware human operators available for increases in scale should be explored when deploying AIES in safety critical contexts. For example, Waymo has remote support staff that

can help guide the vehicle out of problems it cannot solve on its own. This oversight structure has fewer humans than vehicles, which is typically not an issue since every vehicle does not need assistance at any given moment. However, if the large portions of the fleet experience issues, as was the case when power outages in San Francisco stalled many Waymo vehicles, some surge staffing or other mechanism may need to be in place to handle such issues (Roy and Roy 2025; Cummings et al. 2025). In our simulation, we assumed that there is always a human available to delegate to, but this assumption may not hold in deployment. For many of these human-AI architectures to be successful, a certain level of human availability must be guaranteed when inherently brittle AIES fail and fallback to their human operators. Understanding the tradeoffs of different potential human-AI architectures, deploying environment monitoring, and ensuring human operator availability and alertness can help designers take advantage of the tremendous potential of AIES while being prepared for such systems to fail. ■

REFERENCES

- Ammari, Tawfiq, Meilun Chen, S. M. Mehedi Zaman, and Kiran Garimella. 2026. “Learning to Live with AI: How Students Develop AI Literacy Through Naturalistic ChatGPT Interaction.” arXiv:2601.20749. Version 1. Preprint, arXiv, January 28. <https://doi.org/10.48550/arXiv.2601.20749>.
- Beling, Peter, Tom McDermott, Tyler Cody, and Stephen Adams. 2024. *Trusted Artificial Intelligence Systems Engineering Challenge*. SERC-2024-ITR-001_V2. Virginia Tech, Stevens Institute of Technology, Systems Engineering Research Center. <https://sercuarc.org/trusted-artificial-intelligence-challenge-for-armaments-systems-engineering/>.
- Bolocan, Vlad-Octavian, Alexandru Mitoi, Oana Nicu-Canaireca, Maria-Luiza Băean, Cosmin Medar, and Gelu-Adrian Popa. 2025. “Artificial Intelligence in Prostate MRI: Current Evidence and Clinical Translation Challenges—A Narrative Review.” *Journal of Imaging* 11 (10). <https://doi.org/10.3390/jimaging11100335>.
- Cabrera, Ángel Alexander, Adam Perer, and Jason I. Hong. 2023. “Improving Human-AI Collaboration With Descriptions of AI Behavior.” *Proceedings of the ACM on Human-Computer Interaction* 7 (CSCW1): 1–21. <https://doi.org/10.1145/3579612>.
- Cummings, Mary, Victoria Chibuogwu Nneji, and Songpo Li. 2025. “Predicting Workload of Dispatchers Supervising Autonomous Systems.” In *Interdependent Human-Machine Teams*. Academic Press. Academic Press. <https://doi.org/10.1016/B978-0-443-29246-0.00015-8>.
- DeGrave, Alex J., Joseph D. Janizek, and Su-In Lee. 2021. “AI for Radiographic COVID-19 Detection Selects Shortcuts over Signal.” *Nature Machine Intelligence* 3 (7): 610–19. <https://doi.org/10.1038/s42256-021-00338-7>.
- Desmond, Paula A., and Peter A. Hancock. 2000. “Active and Passive Fatigue States.” In *Stress, Workload, and Fatigue*. CRC Press.
- Endsley, Mica R. 2023. “Supporting Human-AI Teams: Transparency, Explainability, and Situation Awareness.” *Computers in Human Behavior* 140 (March): 107574. <https://doi.org/10.1016/j.chb.2022.107574>.
- Fricke, Ernst, and Armin P. Schulz. 2005. “Design for Changeability (DfC): Principles to Enable Changes in Systems throughout Their Entire Lifecycle.” *Systems Engineering* 8 (4): 342–59. <https://doi.org/10.1002/sys.20039>.
- Griner, Alina. 2012. “Human-RRT Collaboration in Unmanned Aerial Vehicle Mission Path Planning.” Thesis, Massachusetts Institute of Technology. <https://dspace.mit.edu/handle/1721.1/76913>.
- Hancock, P. A. 2026. “Human Vigilance in the Age of Intelligent Machines: Challenges and Prospects.” *Ergonomics* 0 (0): 1–17. <https://doi.org/10.1080/00140139.2025.2609813>.
- Hendrycks, Dan, Steven Basart, Norman Mu, et al. 2021. “The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization.” arXiv:2006.16241. Preprint, arXiv, July 24. <https://doi.org/10.48550/arXiv.2006.16241>.
- Hendrycks, Dan, and Thomas Dietterich. 2019. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.” arXiv:1903.12261. Preprint, arXiv, March 28. <https://doi.org/10.48550/arXiv.1903.12261>.
- Koch, Lisa M., Christian F. Baumgartner, and Philipp Berens. 2024. “Distribution Shift Detection for the Postmarket Surveillance of Medical AI Algorithms: A Retrospective Simulation Study.” *Npj Digital Medicine* 7 (1): 120. <https://doi.org/10.1038/s41746-024-01085-w>.
- Koh, Pang Wei, Shiori Sagawa, Henrik Marklund, et al. 2021. “WILDS: A Benchmark of in-the-Wild Distribution Shifts.” *Proceedings of the 38th International Conference on Machine Learning*, July 1, 5637–64. <https://proceedings.mlr.press/v139/koh21a.html>.
- Long, Tao, Katy Ilonka Gero, and Lydia B. Chilton. 2024. “Not Just Novelty: A Longitudinal Study on Utility and Customization of an AI Workflow.” arXiv:2402.09894. Preprint, arXiv, May 31. <https://doi.org/10.48550/arXiv.2402.09894>.

- Nguyen, Anh, Jason Yosinski, and Jeff Clune. 2015. "Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images." arXiv:1412.1897. Preprint, arXiv, April 2. <https://doi.org/10.48550/arXiv.1412.1897>.
- Roy, Abhirup, and Abhirup Roy. 2025. "Waymo's San Francisco Outage Raises Doubts over Robotaxi Readiness during Crises." Autos & Transportation. *Reuters*, December 27. <https://www.reuters.com/business/autos-transportation/waymos-san-francisco-outage-raises-doubts-over-robotaxi-readiness-during-crises-2025-12-27/>.
- Scheurer, J r my, Mikita Balesni, and Marius Hobbhahn. 2024. "Large Language Models Can Strategically Deceive Their Users When Put Under Pressure." arXiv:2311.07590. Preprint, arXiv, July 15. <https://doi.org/10.48550/arXiv.2311.07590>.
- Singh, Aditya, and Zoe Szajnfarder. 2025. "Architecting Human-AI Systems for Effective Collaboration and Oversight: Making Sense of Human/AI-in/on/Over/Under/Along-the-Loop." *Systems Engineering* 29 (2): 337–53. <https://doi.org/10.1002/sys.70024>.
- Souly, Alexandra, Javier Rando, Ed Chapman, et al. 2025. "Poisoning Attacks on LLMs Require a Near-Constant Number of Poison Samples." arXiv:2510.07192. Preprint, arXiv, October 8. <https://doi.org/10.48550/arXiv.2510.07192>.
- Taori, Rohan, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. 2020. "Measuring Robustness to Natural Distribution Shifts in Image Classification." *Advances in Neural Information Processing Systems* 33: 18583–99. <https://proceedings.neurips.cc/paper/2020/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html>.
- Warm, Joel S., Raja Parasuraman, and Gerald Matthews. 2008. "Vigilance Requires Hard Mental Work and Is Stressful." *Human Factors* 50 (3): 433–41. <https://doi.org/10.1518/001872008X312152>.
- Wei, Hui, Hao Tang, Xuemei Jia, et al. 2024. "Physical Adversarial Attack Meets Computer Vision: A Decade Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46 (12): 9797–817. <https://doi.org/10.1109/TPAMI.2024.3430860>.
- Wohleber, Ryan W., Gerald Matthews, Jinchao Lin, et al. 2019. "Vigilance and Automation Dependence in Operation of Multiple Unmanned Aerial Systems (UAS): A Simulation Study." *Human Factors: The Journal of the Human Factors and Ergonomics Society* 61 (3): 488–505. <https://doi.org/10.1177/0018720818799468>.
- Yu, Alice C., Bahram Mohajer, and John Eng. 2022. "External Validation of Deep Learning Algorithms for Radiologic Diagnosis: A Systematic Review." *Radiology. Artificial Intelligence* 4 (3): e210064. <https://doi.org/10.1148/ryai.210064>.

A Systems Architecture for Data-Constrained Environments: Leveraging the Astronaut Digital Twin to Improve Spaceflight Adaptation in Humans

Caleb Schmidt, cmschmidt@patternanalysis.org; and Steve Simske, steve.simske@colostate.edu

Copyright ©2026 by Caleb Schmidt and Steve Simske. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Spaceflight exposes humans to a uniquely harsh operating environment, producing highly individualized physiological responses that existing monitoring and countermeasure paradigms have limited struggle to predict or control. This challenge is compounded by small sample sizes, limited observability, fragmented evidence, and shifting baselines, creating a discordant epistemic environment where conventional data-driven, mechanistic, and hybrid models' methods are not well-suited. Resolving this gap is becoming increasingly urgent: the expanding scope, duration, and population diversity of human spaceflight missions have outpaced the predictive and personalization capability of space medicine now. This article proposes the astronaut digital twin (ADT), a biodigital twin architecture designed to help address three coupled problem classes: inference of physiological state under partial observability, reconciliation of heterogeneous evidence within a causal framework, and individualized countermeasure control under uncertainty. The ADT integrates systems physiology models, digital populations, and individualized digital twins within a Bayesian, constraint-based architecture that continuously updates state estimates and generates uncertainty-bounded, mechanistically interpretable recommendations. By formalizing expert knowledge into a testable human-machine co-learning system, the ADT is intended to enable a transition in space medicine from reactive monitoring toward predictive, personalized decision support. It also serves as a structured hypothesis engine for countermeasure development across diverse physiological profiles. More broadly, the ADT is positioned to engage with open SE4AI questions centered around interpretable reasoning, predictable and diagnosable behavior, and model validity under changing deployment conditions. Although developed for astronaut health, the architecture has the potential to generalize to other high-consequence domains characterized by sparse data, population diversity, and non-stationary baselines, suggesting its potential as a reference framework for trustworthy AI systems in complex operational environments.

INTRODUCTION

Spaceflight is one of the harshest operating environments humans have entered, exposing individuals simultaneously to numerous spaceflight hazards. The space-associated stressors, collectively called spaceflight risks, have significant effects on fluid regulation, muscle mass, bone density, cardiovascular function, neurovestibular control, circadian alignment, immune function, and behavioral

performance, among others (Antonsen et al. 2022a; Antonsen et al. 2023). Over time, these multiple perturbations interact, amplify, and compound differently across individuals. While generalities apply, each individual responds differently to the space environment and to the countermeasures applied, with no strong predictive capacity to determine who will best respond to which approach (Schmidt and Goodwin 2013).

A central challenge is how to rapidly and effectively move beyond the model of monitoring astronauts and deploying broad countermeasure packages toward a paradigm of precisely and preemptively engineering astronaut health through personalized countermeasures (Carnell 2021). The operational objective is to assess each individual's spaceflight risk and performance potential profile in order to

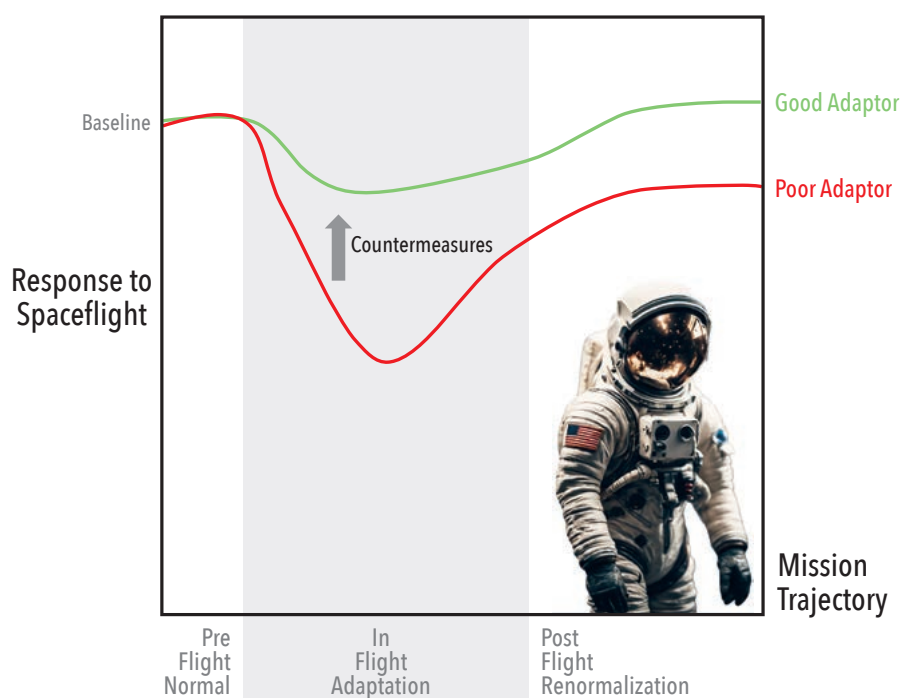


Figure 1. Shifting the spaceflight adaptation curve. Response trajectories for two theoretical adaptation phenotypes, Good Adaptor (green) and Poor Adaptor (red), are plotted across three mission phases: Pre-Flight Normal (baseline), In-Flight Adaptation, and Post-Flight Renormalization. Both trajectories originate at a common pre-flight baseline. During the in-flight adaptation window (shaded region), physiological response diverges between phenotypes, with the Poor Adaptor exhibiting a pronounced decline relative to baseline. Countermeasures (upward arrow) applied during this window differentially modulate each trajectory. Following mission completion, the Good Adaptor recovers to or exceeds baseline, while the Poor Adaptor stabilizes at a persistently reduced level, indicating incomplete renormalization. The y-axis represents a generalized index of physiological response to spaceflight; the x-axis represents mission trajectory as a continuous temporal dimension.

optimize adaptation, transforming poor responders into good responders and good responders into optimal performers, as shown in Figure 1.

Achieving success in this paradigm of personalized medicine in space depends on efficacious and targeted countermeasures (Sishc et al. 2022). Thus, a second challenge is the realization of those countermeasures, which, in turn, requires the rapid development and deployment of a diverse arsenal capable of addressing the most complex challenges in the environment. As we enter an era of increasingly routine and exploration-class human spaceflight, this need becomes increasingly urgent. Missions are getting longer, more autonomous, and more physiologically demanding, while the growth of commercial spaceflight is broadening the diversity of the astronaut population beyond a historically small and highly selected cohort (Shelhamer et al. 2023).

Digital twin technologies have demonstrated broad utility across diverse applications in industry, defense, and other

technically complex domains (Iranshahi et al. 2025). They are digital representations of a target entity with data connections that enable convergence between the physical and digital states at an appropriate rate of synchronization (ISO and IEC 2023). A special class of digital twins, called biodigital twins, offers a unique set of attributes purpose-built for biologically complex systems (Funk et al. 2026). This paper advances the position that the astronaut digital twin (ADT), as a biodigital twin implementation, offers a comprehensive and robust architecture for addressing the needs described above in the spaceflight community.

Importantly, the ADT is defined as an AI system in the formal sense via its integration of probabilistic inference engines, computational systems models, and Bayesian updating to continuously synthesize heterogeneous data and generate personalized, actionable countermeasure recommendations. This architecture satisfies consensus definitions of AI systems as machine learning-based systems that pro-

cess inputs to produce outputs influencing real or virtual environments (OECD 2024; ISO and IEC 2022). Critically, it places the ADT squarely within the SE4AI discourse concerning ensuring trustworthy AI, maintaining robust operation under partial observability, and supporting lifecycle adaptability (SERC-UARC 2026). Further, this framing of grounding the ADT in established AI system architecture means its methodology is not narrowly bound to space medicine, the same architectural logic applies wherever real-time, evidence-driven decision support is required under data constraint and uncertainty.

THE CHALLENGE OF ACCURATELY MODELING HUMAN BIOLOGY IN SPACE

Space medicine is a rigorous, well-developed field with noteworthy achievement and an outstanding safety record in a harsh operating environment that significantly affects the normal human functions (Hurly 2025). Translating the ambition of personalized spaceflight medicine that can engineer better adaptation to space into operational reality runs into two tightly coupled problems that current methods have not resolved.

First, even though the empirical foundation of space medicine rests on a broad and mature field, research remains difficult, fragmented, and heterogeneous (Sibonga and Spector 2024; Kunitzskaya et al. 2022). Publications are substantial but often limited in scope and interoperability, internal physiological states are not always directly observable during flight (Alkhoury et al. 2025), individual baselines diverge substantially, and countermeasure decisions must be made under sustained uncertainty with high operational consequence (Antonsen et al. 2022b). While there continues to be important progress in understanding how human biology responds to the space environment, especially at the multi-omic, molecular level (Garrett-Bakelman et al. 2019; Jones et al. 2024), a significant gap persists in our ability to move from observation and general countermeasures to engineering astronaut health in a way that rapidly iterates and adapts interventions in the field.

Second, the field of space medicine is comprised of highly trained subject matter experts (SMEs) with extensive real-world operational experience who guide assessment and intervention decisions. This expertise is extraordinarily difficult to transfer, reflecting a rare and highly developed body of knowledge that takes years to acquire and presents a steep learning curve for scaling across the discipline (Schmidt 2026). Moreover, the current expert model lacks a rigorous mathematical framework for predicting responses to assessment data

Table 1. Tiered constraint landscape in space medicine with generalized constraints (Afshinneko et al. 2020; Van Puyvelde 2025; Rahimzadeh et al. 2023)

| Constraint Category | Spaceflight-specific Constraint | Generalized Constraint |
|--|---|---|
| T1: Constraints on the Subjects of Study | The astronaut population is small, unrepresentative, and insufficient for conventional data-driven methods | Data scarcity and distributional non-representativeness |
| | Individual variability in stress response causes substantial signal divergence across subjects | High inter-subject heterogeneity/non-ergodicity |
| | Simultaneous exposure to the spaceflight hazards of microgravity, radiation, isolation, confinement, altered circadian cues, and increasing distance from Earth creates nonlinear hazard interactions that resist causal attribution | Strongly coupled, nonlinear, multi-factor interactions |
| T2: Constraints on Observation Capability | Critical biological variables are latent and must be inferred from sparse, heterogeneous proxies under uncertainty | Partial observability and latent state inference |
| | Imaging and invasive procedures are prohibited or severely restricted, narrowing the observable state space | Sensing and instrumentation constraints |
| T3: Constraints on the Evidence Base | Critical operational knowledge is concentrated in SMEs and not encoded in formal predictive systems, limiting the reproducibility, scalability, and quantitative testability of the evidence base | Tacit knowledge bottleneck and knowledge formalization gap |
| | Continuously shifting mission conditions render prior baselines inadequate for characterizing future physiological states | Non-stationarity and domain shift |
| | Ethical and regulatory constraints on experimental design restrict precisely the study designs most capable of generating strong evidence, so knowledge accumulates slowly and asymmetrically | Restricted intervention space and limited ground truth generation |
| T4: Constraints on Intervention and Action | Due to the fundamental shift in the response of the human physiological system, the evidence base of many countermeasures have different space profile when applied to the same system. For instance, spaceflight alters drug metabolism, absorption, and distribution in poorly characterized ways, destabilizing the pharmacological counterfactuals clinicians rely on | Model invalidation under context shift / intervention uncertainty |
| | Limited ability to safely test interventions <i>in situ</i> | High-cost, high-risk action space |
| | Institutional pressures and fragmented communication between operational and scientific communities add systemic filtering on top of the constraints above (out of scope) | Sociotechnical coupling and information degradation |

or for forecasting how those responses will change when countermeasures are applied.

Given these limitations, accomplishing both objectives requires computational approaches that act as systems integrators of myriad data forms, yielding a deep and accurate quantification of physiological state, continuous updating of that state across the mission life cycle, and the ability to apply the right countermeasures at the right time to steer the human system toward better adaptation (Saccetti 2021; Siddiqui et al. 2026). Such a computational approach must be designed not only to

encode individual SME knowledge but also to capture and redistribute the collective expertise of the spaceflight medicine community, functioning as a flight surgeon level reasoning partner in critical conditions and enabling scalable, community level decision support under uncertainty.

THE TIERED CONSTRAINT LANDSCAPE IN SPACE MEDICINE

The gap between the urgent need and the ground state reality in spaceflight medicine is not incidental but structural, shaped by four-tiered categories of compounding

constraints on: T1) subjects of study, T2) observation capability, T3) the evidence base, and T4) intervention and action (Afshinneko et al. 2020; Van Puyvelde 2025; Rahimzadeh et al. 2023). These constraints are shown in Table 1 along with generalized constraints.

Collectively, these constraints constitute a discordant epistemic environment. That is, one requiring an architectural response equal to its complexity. Some constraints are inherent to space medicine and difficult to change, while others endure because current computational systems are insufficient

to address the problem's scope and scale and find it difficult to integrate diverse, sparse data streams across the mission lifecycle. The ADT is designed to address the latter directly.

WHY MANY ARCHITECTURES FAIL IN THIS ENVIRONMENT

The T1–T4 constraints define a discordant epistemic environment requiring principled inference under partial observability, heterogeneous evidence reconciliation within a causal framework, and uncertainty-bounded control decisions auditable by human operators. No existing architectural class satisfies all three, especially when they are rendered in the context of complex mission operational parameters (i.e., design reference missions or DRM: a notional, end-to-end mission scenario that defines the environment, objectives, and sequence of operations).

Purely data-driven systems learn surface correlations but have no native representation of the mechanistic underpinnings of physiological states that define astronaut health risk (Nam et al. 2024; Shojaee et al. 2025). They are fragile under distribution shift and find it hard to support counterfactual reasoning, which is the foundation of individualized countermeasure selection. The T1 constraint is independently disqualifying: the corpus of data that has been gathered from the astronaut population is too small and fragmented to constitute a robust training distribution from which most data-driven systems can generalize with confidence (Afshinnekoo et al. 2020).

Purely mechanistic models encode causal biology explicitly and support scenario simulation, but they operate with fixed parameters calibrated to canonical subjects (Ramakrishnan et al. 2022; Thiele et al. 2013). They produce point predictions rather than individualized, uncertainty-aware state estimates and have no principled mechanism for assimilating sparse, time-evolving observations as an astronaut progresses through a mission. A model that cannot sequentially revise its internal state, especially under the T2 and T3 constraints, is not fully fit to the challenges in spaceflight medicine.

Hybrid data-driven, mechanistic systems improve predictive performance and sometimes quantify uncertainty, but they stop short of solving the full problem set (Arrieta et al. 2020). Evidence integration remains ad hoc rather than causally grounded, and they rarely embed the full operational stack space medicine requires: continuous calibration, decomposed verification of inference and control modules, and interpretable recommendations that support human-machine teaming.

THE CORE PROBLEMS THE ADT MUST SOLVE

The T1–T4 constraints don't just diagnose the gaps in spaceflight medicine, they define the three coupled problem classes any credible computational solution must solve: inference, reconciliation, and control.

The inference problem is estimating internal physiological state when the variables that matter are exceedingly hard to observe in-flight, if at all (Goswami 2026; NASA 2023). The ADT must infer bone remodeling balance, musculoskeletal degradation, and cardiovascular adaptation from incomplete, delayed, noisy proxies, within a context of varying ground-truth at decision time (Hardy et al. 2025). A credible solution must therefore maintain a continuously updated, uncertainty-quantified representation of an astronaut's physiological state. It must be one that improves as observations accumulate rather than degrading as mission conditions shift away from prior baselines.

The reconciliation problem is integrating heterogeneous, incomplete, and sometimes conflicting evidence into a single coherent causal representation when no single dataset is sufficient. Space medicine data are scattered across in-flight physiological sensors, multi-omic profiles, imaging and densitometry, endocrine and biochemical assays, behavioral and neurocognitive assessments, nutritional and exercise logs, analog environment studies (bed rest, head-down tilt, isolated confinement), animal models, and terrestrial clinical studies, each describing only part of the underlying biology (Krittanawong et al. 2022). A valid approach needs to go beyond data fusion by testing evidence against causal mechanisms, capturing population differences, integrating data continuously, and adapting to new findings.

The control problem is translating uncertain state estimates into individualized, auditable countermeasure decisions. The system must determine which astronaut is drifting toward adverse adaptation, which countermeasure to intensify, add, or remove, and how different combinations of exercise, nutrition, and drug therapeutics are likely to affect a specific person at a specific mission phase, all under sustained uncertainty, tight timelines, human-in-the-loop authority, and increasing autonomy as communication latency grows (Antonson et al. 2023). Critically, because the relevant population is too small and unrepresentative to support conventional cohort extrapolation, and because N-of-1 inference is unavailable (it requires crossover designs and repeated within-person measurements spaceflight lack), any solution must produce individualized predictions without relying on either statistical framework (Duan et al. 2022).

THE ADT BIODIGITAL TWIN ARCHITECTURE

The ADT is an implementation of a mature biodigital twin platform whose architecture has emerged from decades of convergent work across systems physiology, computational medicine, translational science, and systems engineering (Funk et al. 2026). Its three additive layers, the systems physiology model, the digital population, and the digital twin each map onto the inference, reconciliation, and control requirements established above, with constraint logic embedded across all three. Together, they produce a system in which physiological implausibility is rejected at the level of governing equations, Bayesian inference operates over a biologically constrained hypothesis space, and every output is traceable to a named assumption or data input.

The systems physiology model addresses the causal representation requirement of the inference and reconciliation problems. Built on systems of nonlinear differential equations that represent multi-scale feedback loops from the cellular level through whole-astronaut functional performance, it encodes the physics-informed dynamic equilibrium of the human system and its responses to nutritional, utilization, perfusion, and environmental perturbations such as microgravity. Model behavior is rigorously validated against high-quality clinical data from both Earth and space, establishing the mechanistic envelope within which all downstream inference is bounded.

The digital population addresses the reconciliation problem's requirement for cross-context evidence integration and the control problem's requirement for individualized inference without cohort extrapolation (Ippolito et al. 2024). By generating variants in systems physiology model parameters to create a high-dimensional hypothesis space, constrained by population data and calibrated until the generated population reproduces chosen validation statistics, this layer serves as an evidence integrator that captures inter-individual variability while keeping every hypothesis within biological possibility. The small-N generalization problem is structurally reframed: rather than extrapolating from population averages, the ADT ranks physiological state hypotheses for a specific individual against this constrained prior.

The digital twin closes the inference loop. It uses the probability-weighted hypotheses from the digital population as a Bayesian prior, then refines it continuously using an individual astronaut's data, with each updated state serving as the prior for the next inference cycle. This satisfies the inference problem's requirement for a continuously

updating, non-degrading state representation, because sequential Bayesian updating allows prior states to evolve with the astronaut rather than anchoring to launch-day baselines that mission conditions have changed. Operationally, this enables assessment of the current and evolving physiological state, simulation of counterfactual countermeasure scenarios, and generation of uncertainty-bounded forecasts to support countermeasure control, including selection and timing.

A useful analogy for this process is that of a hurricane forecast, which maps cleanly onto the three-layer architecture (Arachchige et al. 2025). Just as a forecast system encodes governing atmospheric physics, generates an ensemble of probability-weighted trajectories, and narrows its cone as new observations arrive, the ADT encodes governing biology and populates a constrained hypothesis space through the digital population. It then continuously sharpens its physiological trajectory as individual astronaut data are ingested. In both cases, implausible states are rejected not because the output looks wrong, but because it violates the underlying equations.

Constraint logic is the architectural property that binds the three ADT layers into a coherent response to the inference, reconciliation, and control problems. Because physiologically implausible trajectories are rejected at the level of governing equations rather than filtered post hoc, anomalous outputs can be traced to a named assumption, parameter boundary, or data input. This makes the system diagnosable by design and distinguishes it from a high-dimensional statistical model. Funk et al. describe this as a testable architecture grounded in homeostasis, where the ability to expose knowledge gaps is a first-class architectural feature rather than a debugging convenience (Funk et al. 2026).

Importantly, the ADT can incorporate virtual countermeasures by linking specific interventions to mechanistic parameters or nodes within the digital twin platform and using counterfactual test how different actions, doses, durations, and frequencies may affect an individual astronaut over time (Ippolito et al. 2024). As evidence-based countermeasures, such as exercise, nutrition, and drug methods, accumulate, the ADT evaluates and ranks combinations of interventions for a given astronaut, then updates those predictions through a closed-loop learning process informed by observed responses rather than trial-and-error alone. This approach enables adaptive optimization of countermeasure strategies and supports early alerts when the preferred intervention profile changes.

TRUST AND MODULARITY IN THE ADT

The ADT's trustworthiness depends on both its internal constraint logic and its external verification and validation process (Viceconti 2021). Constraint logic ensures that output remains physiologically coherent, while the V&V lifecycle ensures that each layer of the architecture is tested, calibrated, and updated against evidence over time. In this sense, architectural validity and lifecycle assurance are complementary rather than redundant.

Developed as an agile lifecycle digital twin, the ADT is iteratively refined from concept through operational deployment via the SME-validated integration of space research and mission data (Gadkar et al 2016). This agile methodology is necessitated by the fact that the system of interest (i.e., the human system) was realized through biological evolution, rather than specification-first design, and, as such, it forces reconstruction from observed data and continuous updating as new evidence is acquired. Verification and validation are therefore distributed across the ADT lifecycle rather than confined to a single end-stage review. Each new data-ingestion event is another test of model adequacy. If posterior estimates diverge from observation beyond uncertainty bounds, the model must be revised. In this way, model inadequacy becomes visible during active development or deployment rather than remaining hidden inside an opaque model. These discrepancies then drive updating, retesting, and revalidation.

The architecture is intentionally modular. Musculoskeletal adaptation, bone remodeling, cardiovascular deconditioning, radiation effects, and behavioral performance must each be built as separate submodules with clear interfaces to the full system. This lets verification evidence accumulate by domain without forcing full-system revalidation every time one module is updated.

HUMAN-MACHINE CO-LEARNING IN THE ADT

The ADT promises to help reframe expert knowledge from concentrated, individually held operational judgment into explicit, computable, and continuously testable model structure. Flight surgeons and domain experts determine which mechanistic relationships are modeled, what population data is used for supporting variability and validation, and how countermeasures are represented. Their experience is not external input to the architecture, rather, it is the architecture. This means that all data is always considered within the context of how it was acquired and how it relates to the larger system, allowing SME expertise to flow through the architecture as it converts it to physiological hypotheses,

intervention strategies, and expected responses that can be quantitatively evaluated and iteratively refined.

When fully realized within the parameters of a DRM or reference mission, the ADT extends this formalization into an operational decision support capability that complements rather than supplants clinical authority. This capability allows the flight surgeon's role to shift into the domain of personalized medicine and precision health engineering. The ADT continuously infers physiological state from heterogeneous data streams and assesses forward risk trajectories, absorbing the otherwise intractable burden of signal integration (Schmidt et al. 2025). Outputs are generated through real-time Bayesian updating across a digital population of probability-weighted physiological hypotheses, ensuring each recommendation reflects an individual's biological dynamics rather than population-average extrapolation. Every output is mechanistically traceable, bounded within a validated physiological envelope, and accompanied by explicit uncertainty quantification, enabling the flight surgeon to interrogate, override, or act with confidence while offloading inferential workload without relinquishing clinical authority. Additionally, layering agentic AI onto the digital twin stack further extends accessibility, streamlining interrogation of complex dynamics and generating highly context-aware visual outputs.

HYPOTHESIS GENERATION, FILTERING, AND COUNTERMEASURE DEVELOPMENT

Beyond individual clinical decision support, the ADT functions as a hypothesis engine for countermeasure discovery, filtering, and refinement. Because the digital population encodes a high-dimensional, biologically constrained parameter space spanning the known inter-individual variability, the system can be interrogated in silico across that space to systematically evaluate biological questions of interest or novel countermeasure strategies. For example, candidate countermeasures, such as nutritional, drug-based, exercise-based, or combinatorial approaches, can be deployed across thousands of simulated physiological variants simultaneously, enabling rapid identification of strategies that are robust across heterogeneous adaptation profiles as well as those that are effective only within specific subpopulations. This capability introduces an explicit filtering layer in which mechanistic plausibility and population-level consistency determine which interventions advance to empirical testing. In contrast to the current paradigm, where countermeasure development is slow, sample-limited, and operationally

constrained by small astronaut cohorts (Wang et al. 2022), the ADT promises the capability of a high-throughput, iterative screening pipeline that compresses the cycle between hypothesis generation and validation significantly. This function is not ancillary to the ADT's clinical role, it is a primary mechanism by which the system advances both operational readiness and the scientific foundation of space medicine.

ADT ALIGNMENT, TRANSFERABILITY, AND CONTRIBUTION TO SE4AI OPEN QUESTIONS

The SERC SE4AI roadmap identifies three pretendant open research concerns: interpretable reasoning for human-machine teaming, predictable and diagnosable behavior in safety-critical settings, and model validity under changing deployment conditions (SERC-UARC 2026). The ADT addresses all three by design: (1) every recommendation traces to SME-validated priors and data inputs, giving operators a probability-weighted argument rather than an unexplained output; (2) biological and physical laws encoded as hard constraints bound predictions within a certified physiological envelope; and (3) continuous Bayesian updating keeps the twin calibrated throughout the mission, adapting to new data rather than anchoring

to invalidated launch-state baselines. As such, the architectural considerations of the ADT may have the potential to serve as a reference implementation for SE4AI systems more broadly, with direct applicability to any domain requiring interpretable reasoning, bounded behavior, and adaptive validity under non-stationary conditions.

CONCLUSION

Spaceflight medicine presents a uniquely discordant epistemic environment, defined by small and unrepresentative populations, constrained observables, fragmentary evidence, and high-consequence decisions under sustained uncertainty. Within this setting, the ADT's three-layer biodigital twin architecture is best understood not as a replacement for existing computational approaches, but as a structured attempt to address a class of problems that conventional data-driven, mechanistic, and hybrid methods each only partially resolve under these constraints. By grounding causal representation in systems physiology, integrating heterogeneous evidence through a biologically constrained hypothesis space, and applying sequential Bayesian inference to generate individualized, uncertainty-bounded countermeasure recommendations,

the ADT provides a unified framework for approaching the coupled problems of inference, reconciliation, and control. These capabilities are designed to be interpretable, diagnosable, and adaptable across the mission lifecycle, though their effectiveness depends on verification, calibration, operational testing, and continued validation.

The broader significance of this work extends beyond the astronaut domain. The same coupled problem classes of partial observability, heterogeneous evidence integration, and individualized control under small-N conditions arise in other high-consequence settings. Spaceflight concentrates these challenges in a form that is tractable for formal analysis, making it a useful testbed for exploring what trustworthy AI systems may achieve under genuinely discordant data conditions. While architectural principles developed here are likely transferable, their applicability will depend on the presence of sufficiently stable underlying system dynamics and the availability of mechanistically meaningful constraints. As such, the ADT should be viewed not as a general solution, but as a reference architecture that helps define both the opportunities and the open questions for SE4AI systems in comparable operational environments. ■

REFERENCES

- Afshinnekoo, Ebrahim, Ryan T. Scott, Matthew J. MacKay, et al. 2020. "Fundamental Biological Features of Spaceflight: Advancing the Field to Enable Deep-Space Exploration." *Cell* 183 (5): 1162–84. <https://doi.org/10.1016/j.cell.2020.10.050>.
- Alkhoury, Bassam, Joseph D. Strock, and Santiago Schnell. 2025. "Current State and Open Problems in Universal Differential Equations Applied to Systems Biology." *npj Systems Biology and Applications* 11:86. <https://doi.org/10.1038/s41540-025-00550-w>.
- Antonsen, Erik L., Robert J. Reynolds, Avalon Monti, Jacqueline Charvat, Devan Petersen, Erin S. Connell, and Mark Shelhamer, et al. 2022a. "Directed Acyclic Graph Guidance Documentation." NASA Technical Memorandum NASA/TM-20220006812. Houston: National Aeronautics and Space Administration, Johnson Space Center. <https://ntrs.nasa.gov/citations/20220006812>.
- Antonsen, Erik L., Jerry G. Myers, Lynn Boley, et al. 2022b. "Estimating Medical Risk in Human Spaceflight." *Npj Microgravity* 8 (1): 8. <https://doi.org/10.1038/s41526-022-00193-9>.
- Antonsen, Erik L., Erin Connell, Wilma Anton, Robert J. Reynolds, Daniel M. Buckland, and Mary Van Baalen. 2023. "Updates to the NASA Human System Risk Management Process for Space Exploration." *Npj Microgravity* 9 (1): 72. <https://doi.org/10.1038/s41526-023-00305-z>.
- Arachchige, Sameera Maha, Biswajeet Pradhan, and Hyuck-Jin Park. 2025. "A Critical Review of Hurricane Risk Assessment Models and Predictive Frameworks." *Geoscience Frontiers* 16 (3): 102012. <https://doi.org/10.1016/j.gsf.2025.102012>.
- Barredo Arrieta, Alejandro, Natalia Díaz-Rodríguez, Javier Del Ser, et al. 2020. "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI." *Information Fusion* 58: 82–115. <https://doi.org/10.1016/j.inffus.2019.12.012>.
- Carnell, Lisa S. 2021. "Spaceflight Medical Countermeasures: A Strategic Approach for Mitigating Effects from Solar Particle Events." *International Journal of Radiation Biology* 97 (sup1): S125–31. <https://doi.org/10.1080/09553002.2020.1820603>.
- Duan, Naihua, Daniel Norman, Christopher Schmid, Ida Sim, and Richard L. Kravitz. 2022. "Personalized Data Science and Personalized (N-of-1) Trials: Promising Paradigms for Individualized Health Care." *Harvard Data Science Review*. <https://doi.org/10.1162/99608f92.8439a336>.
- Funk, Cory C., Tom Paterson, Alex Bangs, et al. 2025. "Mining the Gaps: Deciphering Alzheimer's Biology through AI-Driven Reconciliation." *The Journal of Prevention of Alzheimer's Disease* 13 (1): 100402. <https://doi.org/10.1016/j.tjpad.2025.100402>.
- Gadkar, K., Dc Kirouac, De Mager, Ph Van Der Graaf, and S. Ramanujan. 2016. "A Six-Stage Workflow for Robust Application of Systems Pharmacology." *CPT: Pharmacometrics & Systems Pharmacology* 5, no. 5: 235–49. <https://doi.org/10.1002/psp4.12071>.
- Garrett-Bakelman, Francine E., Manjula Darshi, Stefan J. Green, et al. 2019. "The NASA Twins Study: A Multidimensional Analysis of a Year-Long Human Spaceflight." *Science* 364 (6436): eaau8650. <https://doi.org/10.1126/science.aau8650>.
- Goswami, Nandu, Jerry Joseph Batzel, Yaopeng J. X. Ma, Per Morten Fredriksen, David Andrew Green, and Plamen Ch Ivanov. 2026. "Network Physiology in Space: Vision and Perspectives on Exploring Physiological Networks during Spaceflight for the Benefit of Life on Earth." *Frontiers in Network Physiology* 6. <https://doi.org/10.3389/fnetp.2026.1817815>.
- Hardy, John G., Corey A. Theriot, Thomas Oswald, and Gilles Clément. 2025. "Spaceflight Standard Measures Is a Multidisciplinary

- plinary Study That Systematically Monitors Risks to Astronaut Health and Performance.” *Npj Microgravity* 11(1): 78. <https://doi.org/10.1038/s41526-025-00532-6>.
- Hurly B. 2025. “The Evolution and Future of Space Medicine” *NewSpace Economy*, June 19. <https://newspaceconomy.ca/2025/06/19/the-evolution-and-future-of-space-medicine/>.
 - International Organization for Standardization and International Electrotechnical Commission. 2023. *Digital Twin — Concepts and Terminology*. ISO/IEC 30173:2023. Geneva: ISO. <https://www.iso.org/standard/81442.html>.
 - International Organization for Standardization and International Electrotechnical Commission. 2022. *Information Technology—Artificial Intelligence—Artificial Intelligence Concepts and Terminology*. ISO/IEC 22989:2022. Geneva: ISO. <https://www.iso.org/standard/74296.html>.
 - Ippolito, Alberto, Hanwen Wang, Yu Zhang, Vahideh Vakil, and Aleksander S. Popel. 2024. “Virtual Clinical Trials via a QSP Immuno-Oncology Model to Simulate the Response to a Conditionally Activated PD-L1 Targeting Antibody in NS-CLC.” *Journal of Pharmacokinetics and Pharmacodynamics* 51 (6): 747–57. <https://doi.org/10.1007/s10928-024-09928-5>.
 - Iranshahi, Kamran, Joshua Brun, Tim Arnold, Thomas Sergi, and Ulf Christian Müller. 2025. “Digital Twins: Recent Advances and Future Directions in Engineering Fields.” *Intelligent Systems with Applications* 26 (June): 200516. <https://doi.org/10.1016/j.iswa.2025.200516>.
 - Jones, Christopher W., Elishah G. Overbey, Jerome Lacombe, et al. 2024. “Molecular and Physiological Changes in the SpaceX Inspiration4 Civilian Crew.” *Nature* 632 (8027): 1155–64. <https://doi.org/10.1038/s41586-024-07648-x>.
 - Kunitskaya, Alina, James M. Piret, Nicole Buckley, and Etienne Low-Décarie. 2022. “Meta-Analysis of Health Research Data from Greater than Three Months International Space Station Missions.” *Acta Astronautica* 201: 420–30. <https://doi.org/10.1016/j.actaastro.2022.09.019>.
 - Krittanawong, Chayakrit, Nitin Kumar Singh, Richard A. Scheuring, et al. 2022. “Human Health during Space Travel: State-of-the-Art Review.” *Cells* 12 (1): 40. <https://doi.org/10.3390/cells12010040>.
 - Nam, Yonghyun, Jaesik Kim, Sang-Hyuk Jung, et al. 2024. “Harnessing Artificial Intelligence in Multimodal Omics Data Integration: Paving the Path for the Next Frontier in Precision Medicine.” *Annual Review of Biomedical Data Science* 7 (1): 225–50. <https://doi.org/10.1146/annurev-biodatasci-102523-103801>.
 - NASA. 2023. “Counteracting Bone and Muscle Loss in Microgravity.” NASA, November 30. <https://www.nasa.gov/missions/station/iss-research/counteracting-bone-and-muscle-loss-in-microgravity/>.
 - Organisation for Economic Co-operation and Development. 2024. *Explanatory Memorandum on the Updated OECD Definition of an AI System*. OECD Artificial Intelligence Papers 8. Paris: OECD Publishing. <https://doi.org/10.1787/623da898-en>.
 - Parshin Shojae, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. “The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity,” *Apple Machine Learning Research*. <https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf>.
 - Rahimzadeh, Vasiliki, Jennifer Fogarty, Timothy Caulfield, et al. 2023. “Ethically Cleared to Launch?” *Science* 381 (6665): 1408–11. <https://doi.org/10.1126/science.adh9028>.
 - Saccenti, Edoardo. 2021. “Specialty Grand Challenge: Data and Model Integration in Systems Biology.” *Frontiers in Systems Biology* 1: 800894. <https://doi.org/10.3389/fsysb.2021.800894>.
 - Schmidt, M. 2026. Personal Communication—Email Message to Author, May 21.
 - Schmidt, Caleb M., Tom Paterson, and Michael A. Schmidt. 2025. “The Astronaut Digital Twin: Accelerating Discovery and Countermeasure Development in the Optimization of Human Space Exploration.” In *Building a Space-Faring Civilization: Advancing the Renaissance of Science, Medicine, and Human Performance in Civilian Spaceflight*, edited by Michael A. Schmidt and Marianne J. Legato, 245–256. Cambridge, MA: Academic Press. <https://doi.org/10.1016/B978-0-443-13850-8.00022-1>.
 - Schmidt, Michael A., and Thomas J. Goodwin. 2013. “Personalized Medicine in Human Space Flight: Using Omics Based Analyses to Develop Individualized Countermeasures That Enhance Astronaut Safety and Performance.” *Metabolomics* 9 (6): 1134–56. <https://doi.org/10.1007/s11306-013-0556-3>.
 - SERC-UARC. 2026. 2025 AI4SE – SE4AI Final Report, v2.1. February. <https://sercuarc.org/wp-content/uploads/2026/02/2025-AI4SE-SE4AI-Final-Report-v2.1.pdf>.
 - Shelhamer, Mark, Erik Antonsen, and Robert Reynolds. 2023. “The Need for a Systems Approach in Spaceflight Health and Medicine.” 2023 *IEEE Aerospace Conference*, March 4, 1–10. <https://doi.org/10.1109/AERO55745.2023.10115773>.
 - Sibonga, Jean D., and Elisabeth R. Spector. 2024. “Evidence Report: Risk of Bone Fracture due to Spaceflight-Induced Changes to Bone.” HRP-F07-ERFT.R2. Houston, US-TX: NASA Johnson Space Center, Human Research Program, Human Health Countermeasures Element. https://humanresearchroadmap.nasa.gov/Evidence/reports/Bone%20Fracture%20Evidence%20Report_07-18-2024.pdf.
 - Siddiqui, Ruqaiyyah, Rizwan Qaisar, Adel Elmoselhi, and Naveed Ahmed Khan. 2026. “Digital Twin Modelling in Microgravity: A Framework for Predictive and Personalised Space Medicine.” *Life Sciences in Space Research* 48: 37–39. <https://doi.org/10.1016/j.lssr.2025.11.004>.
 - Sishc, Brock J., Janice Zawaski, Janapriya Saha, Lisa S. Carnell, Kristin M. Fabre, and S. Robin Elgart. 2022. “The Need for Biological Countermeasures to Mitigate the Risk of Space Radiation-Induced Carcinogenesis, Cardiovascular Disease, and Central Nervous System Deficiencies.” *Life Sciences in Space Research* 35: 4–8. <https://doi.org/10.1016/j.lssr.2022.06.003>.
 - Thiele, Ines, Neil Swainston, Ronan M. T. Fleming, et al. 2013. “A Community-Driven Global Reconstruction of Human Metabolism.” *Nature Biotechnology* 31 (5): 419–25. <https://doi.org/10.1038/nbt.2488>.
 - Van Puyvelde, Martine, Nicholas H. Van Den Berg, Lara Stas, et al. 2025. “Beyond the Lab Coat: Methodological Challenges in Space Life Sciences.” *Frontiers in Physiology* 16: 1663701. <https://doi.org/10.3389/fphys.2025.1663701>.
 - Viceconti, Marco, Francesco Pappalardo, Blanca Rodriguez, Marc Horner, Jeff Bischoff, and Flora Musuamba Tshinanu. 2021. “In Silico Trials: Verification, Validation and Uncertainty Quantification of Predictive Models Used in the Regulatory Evaluation of Biomedical Products.” *Methods* 185 (January): 120–27. <https://doi.org/10.1016/j.jymeth.2020.01.011>.
 - Wang, Z., C. Gao, L. Glass, and J. Sun. 2022. “Artificial Intelligence for In Silico Clinical Trials: A Review.” *ArXiv, abs/2209.09023*. <https://arxiv.org/abs/2209.09023>.

ABOUT THE AUTHORS

Caleb Schmidt is with the Department of Systems Engineering, College of Engineering, Colorado State University, Fort Collins, US-CO; and Sovaris Aerospace, Boulder, US-CO.

Steve Simske is with the Department of Systems Engineering, College of Engineering, Colorado State University, Fort Collins, US-CO.

From Models to Systems: The Evolution of T&E for Artificial Intelligence from 2020 to 2030

Tyler Cody, tcody@virginia.edu; and Peter Beling

Copyright ©2026 by Tyler Cody and Peter Beling. Permission granted to INCOSE to publish and use.

■ ABSTRACT

Test and evaluation (T&E) for artificial intelligence is entering a new phase. In 2020, the central challenge was to treat machine learning models as system functions rather than as ordinary software components. By 2025, that view had expanded: AI-enabled systems had to be evaluated in operational, organizational, and lifecycle context, with attention to trust, adaptation, resilience, and human-AI teaming. Looking toward 2030, an even larger shift is emerging. AI is no longer only part of the delivered capability; it is increasingly part of the engineering machinery that produces requirements, code, tests, models, documentation, and assurance evidence. This article traces that progression from model-centric assurance to lifecycle-centered assurance and, ultimately, to T&E for AI-generated systems. It argues that future assurance will depend not only on testing delivered systems, but also on evaluating the provenance, governance, and stability of the generative pipelines that create and modify them.

INTRODUCTION

In 2020, the systems engineering community was still working through a basic but consequential realization: machine learning models were no longer just analytical aids on the side of engineering practice. They were becoming system functions. If a perception model, recommender, planner, or classifier could materially affect mission performance, safety, or human decision making, then it had to be treated as an engineered element inside the system of interest. That meant test and evaluation (T&E) could no longer remain focused only on conventional software correctness, hardware certification, and subsystem verification. It had to confront a new type of component: one whose behavior depended on data, training, operating context, and probabilis-

tic performance rather than only explicit, deterministic logic. That realization was already visible in the March 2020 INCOSE *INSIGHT* special issue on AI and Systems Engineering (McDermott et al. 2020; Freeman 2020; Cody et al. 2020; Collopy et al. 2020) and in the first Systems Engineering Research Center (SERC) AI4SE/SE4AI workshop (SERC 2020), both of which framed the moment as a transformation in systems engineering methods, procedures, and tools for systems with embedded AI.

Five years later, by 2025, the question had changed. The problem was no longer simply how to engineer a system with an AI component embedded inside it. AI capabilities had become more general, more pervasive, and more entangled with human organizations, operational environments,

and digital engineering infrastructure. Foundation-model-era AI was not merely another subsystem (Bommasani et al. 2021). It was increasingly a mediating capability that shaped interfaces, workflows, trust relationships, and patterns of adaptation across larger socio-technical systems. The SERC workshop sequence from 2022 through 2025 shows this progression clearly: AI had to be tested in system context; digital twins and lifecycle evidence became more important; trust, human-AI teaming, adaptive T&E, and resilience became central concerns; and by 2025 workshop participants were discussing cognitive assistants, generative tools, autonomous test generation, model-based systems engineering (MBSE) integration, and the formalization of trust

and governance as system properties (SERC 2022; SERC 2023; SERC 2024; SERC 2025).

Looking toward 2030, the next shift is now visible. We are moving from **AI-enabled systems** to **AI-generated systems**. In the first regime, AI is part of the delivered operational capability. In the second, AI is also part of the productive machinery that creates the system artifacts themselves: requirements drafts, software modules, interface logic, test cases, mission threads, models, documentation, and perhaps even binaries. That shift changes what T&E must evaluate. It is no longer enough to test the delivered system alone. We must also assess the generative processes, model chains, evidence structures, and governance mechanisms by which the system was produced and may continue to change (NIST 2023). This is where the SERC roadmaps' emphasis on digitalization, lifecycle adaptation, trust, and engineering velocity becomes especially important (McDermott et al. 2020; McDermott 2025).

In this article, we trace the progress of AI and automation in systems engineering across the decade from 2020 to 2030, depicted in Figure 1. Two definitions help clarify this progression. An **AI-enabled system** is a system whose operational capability depends materially on one or more AI components, but whose requirements, architecture, interfaces, implementation baseline, and assurance case are still primarily produced through conventional human-led engineering processes. An **AI-generated system** is a system in which AI is used not only as an operational component, but also as a productive mechanism for creating or modifying system artifacts themselves. In this second regime, the T&E burden expands: the enterprise must evaluate the system's behavior, the provenance and control of generated artifacts, the stability of the generation pipeline, and the validity of evidence assembled across an evolving digital thread. This distinction is not a statutory one; it is a systems-engineering abstraction proposed here to make sense of a decade-long shift that is already visible in SERC's workshop themes and in broader generative-AI practice.

This distinction can also be understood as a change in the relationship between AI4SE and SE4AI. Traditionally, AI4SE has referred to the use of AI to improve systems engineering activities, while SE4AI has referred to the use of systems engineering methods to develop, assure, and govern AI-enabled systems. Generative AI makes this relationship more recursive. AI4SE no longer merely supports the engineering process from the outside. It becomes defining of the engineered system.

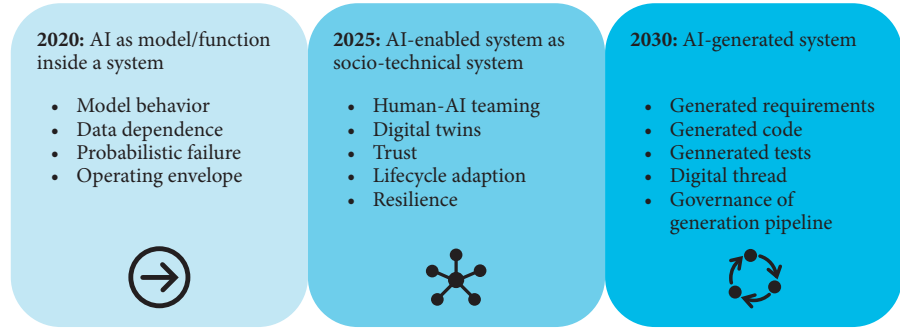


Figure 1. Evolution of the T&E object from model-centric assurance in 2020, to system- and lifecycle-centric assurance by 2025, to pipeline- and provenance-aware assurance for AI-generated systems by 2030

Otherwise put, in AI-generated systems, the generative process itself becomes part of the system of interest. T&E must therefore evaluate not only AI components and their operational behavior, but also the AI-enabled production machinery that creates, modifies, and justifies system artifacts over time. In this sense, the SE4AI problem area, including T&E for AI, must be reconsidered because AI4SE is beginning to reshape the very objects that SE4AI is meant to assure.

2020: AI AS A SPECIAL KIND OF SOFTWARE INSIDE THE SYSTEM

The 2020 baseline now looks almost simple by comparison. At that point, the major conceptual task for systems engineering was to absorb AI into the system lifecycle without pretending it behaved like ordinary software. The March 2020 *INSIGHT* special issue gathered that challenge into a coherent agenda (McDermott et al. 2020; Freeman 2020; Cody et al. 2020; Collopy et al. 2020). The issue framed “AI4SE” and “SE4AI” as dual transformations: using AI to improve systems engineering practice, and evolving systems engineering so that systems with embedded AI could be engineered to be fit for purpose and to avoid unintended harm. The inaugural SERC workshop in 2020 echoed that view and established a roadmap whose long-term outcome was human-machine co-learning, signaling early that AI would strain lifecycle models built around static baselines and fixed design assumptions (SERC 2020).

For T&E, the immediate implication was straightforward but demanding. If a machine learning model could perform a system function, then evidence of suitability could no longer be derived solely from source inspection, deterministic requirements satisfaction, or regression testing in the familiar software sense. Laura Freeman’s 2020 *INSIGHT* article on T&E for AI argued that AI systems still require testable, repeatable, auditable decisions,

but that the methods for establishing such evidence must broaden to include data dependencies, probabilistic performance, novel failure modes, and evaluation across broader operating envelopes (Freeman 2020). Around the same time, SERC workshop materials and related presentations highlighted challenges such as continuous learning, uncertainty in configuration management, the need for new analysis and certification approaches for training datasets, and the role of enabling sensors and context in shaping system behavior (SERC 2020).

This first phase can therefore be described as the era in which systems engineering learned to say: *this is not merely software*. It is a new kind of software-linked capability whose behavior cannot be fully understood apart from data, training, interfaces, users, and context. That was the first major contribution of the SERC community from 2020 onward: to force a systems-level interpretation of AI before the rest of the engineering enterprise had fully adjusted to it.

2021-2023: FROM COMPONENT ASSURANCE TO LIFECYCLE ADAPTATION

The second stage of the decade was a broadening of scope. In 2021, the second SERC AI4SE/SE4AI workshop explicitly tied the roadmap to a future in which humans and machines co-adapt under dynamic operational conditions (SERC 2021). The workshop framing emphasized that systems engineering would have to prepare for worlds in which trust, teaming, adaptation, and lifecycle evolution are not edge cases but normal conditions. Even the 2020 report, in its roadmap discussion, had already suggested that T&E processes would have to adapt to co-learning environments and that trust would become critical in joint human-machine decision settings. By 2021, that direction was no longer speculative; it had become a research agenda.

The 2022 workshop sharpened the point in especially useful language: **AI is a system.** That formulation matters. It marked an evolution from “AI is a special component” to “AI behavior must be understood in the context of larger system structure and guardrails.” The 2022 report states that mitigating AI uncertainty and guarding against emergent events must begin from the start of system design and be shared by designers, requirement setters, and systems engineers (SERC 2022). It also argues that testing of AI needs to be done in the context of a system and that digital twins are becoming indispensable for managing the lifecycle of AI systems because they support real-time learning, testing, refinement, and trust building. This is a decisive shift from isolated model validation toward model-based, lifecycle-centered system assurance.

By 2023, the center of gravity had moved again. The workshop report and updated roadmap emphasized robust testing in dynamic environments, dataset quality, explainability, trust metrics, infrastructure standardization, active learning, and adaptive systems (SERC 2023). The updated roadmap included themes such as T&E as a continuum, lifecycle adaptation and trust, AI system architecting, standards of trust, AI risk analysis, AI specification, and human-AI team testbeds (McDermott et al. 2023). In parallel, SERC-affiliated research on T&E with explainable AI and counterfactuals showed how the community was moving beyond aggregate accuracy and into methods for identifying performance envelopes, exposing hidden decision structure, and probing responses to unmodeled uncertainty (Freeman et al. 2022; Cody et al. 2022; Seif et al. 2024; Gregory and Salado 2024; Chandrasekaran et al. 2024). The message was that T&E for AI could not be reduced to benchmark scores; it required richer forms of system identification and evidence.

This 2021–2023 period is therefore best understood as the moment when the community stopped asking only whether AI components were accurate enough and started asking whether larger AI-enabled systems could be trusted, adapted, and governed over time. The object of assurance expanded from the model to the lifecycle.

2024–2025: AI-ENABLED SYSTEMS AS SOCIO-TECHNICAL SYSTEMS

The present state, as reflected in the 2024 and 2025 SERC workshop reports, is one in which AI-enabled systems are clearly treated as socio-technical systems embedded in digital ecosystems rather than as isolated technical artifacts (SERC 2024; SERC 2025). The 2024 report

emphasizes that AI-enabled systems are multidisciplinary, that reliability in critical situations is essential, that explainability and measurement of trust remain active priorities, and that T&E should be understood as a continuous activity aimed at predicting performance in evolving environments rather than simply validating fixed requirements. The 2025 report goes further by discussing cognitive assistants, generative models accelerating systems-model development, AI and MBSE integration, ontology-enabled digital ecosystems, and autonomous test generation. It also frames governance, trustworthiness, and resilience as properties that must be formalized and managed across the lifecycle.

That evolution supports a stronger claim: by 2025, systems engineers are increasingly dealing with AI not just as a subsystem, but as a relational property of complex systems of systems. AI mediates how humans interact with data, each other, and the world; how models are updated; how digital threads are maintained; how evidence is produced; and how system behavior changes under shifting context. In such a setting, the engineering problem is not simply to certify a model artifact. It is to manage a network of functions, interfaces, data pipelines, decision rights, operational assumptions, and trust relationships that collectively determine performance (Shadab et al. 2023; Shadab et al. 2024). This is why resilience, continuous T&E, and digital engineering appear so persistently in the SERC research: they are not optional enhancements but responses to the fact that the assurance object has become dynamic and distributed (McDermott 2025; Sandman et al. 2025).

2030: THE RISE OF AI-GENERATED SYSTEMS

The next five years are likely to intensify this trajectory. The most important conceptual shift for T&E may be the transition from AI-enabled systems to AI-generated systems. The reason is simple: generative AI is increasingly being applied to the production of engineering artifacts themselves. This is not simply an extension of engineering automation. It changes the object of assurance. In conventional AI4SE, AI may accelerate or improve engineering work while the resulting artifacts remain the primary objects of configuration management and T&E. In AI-generated systems, however, the artifacts may be continuously produced, revised, and recombined by AI-enabled processes. The T&E problem therefore expands from evaluating only products to evaluating product-generating processes.

SERC-related 2025 workshop material

described cognitive assistants and generative models as accelerants for lifecycle functions such as language generation, project tracking, model development, and traceability. Related presentations also pointed to agentic architectures for lifecycle traceability and to digital-thread concepts that connect concept, requirements, design, implementation, test, operations, and retirement using context-aware automation (SERC 2025). Outside the SERC corpus, current system-card practice for frontier models further indicates that generative AI now arrives with system-level safety evaluations, operational constraints, and governance scaffolding rather than lightweight product descriptions (Hurst et al. 2024; Anthropic 2025).

When AI helps produce software, tests, models, and even executable artifacts, T&E inherits a more difficult problem. The enterprise must evaluate not only the delivered artifact, but the confidence one should place in the artifact-generation process. Was the generated requirement traceable? Did the generated code preserve assumptions from the architecture? Did the generated test set meaningfully span the operational envelope, or merely restate nominal behavior? Did a model-driven assistant produce evidence that is structurally complete but epistemically thin? These are not speculative questions. They are the logical extension of the same issues the SERC community has been raising since 2020: data quality, lifecycle adaptation, trust calibration, model governance, and continuous assessment. The difference is that by 2030 those issues will attach not only to deployed AI components but to the engineering machinery that builds the system in the first place.

In that environment, “AI-enabled” may become too weak a label. If AI is ubiquitous in design environments, code generation, model construction, evidence management, and operational adaptation, then many future systems will be AI-generated whether or not their end-user functionality is branded as “AI.” This matters because current policy and acquisition structures were largely shaped for programs in which system artifacts, baselines, and evidence products change more slowly than the generative pipelines now emerging. Recent U.S. Title 10 language already points toward a need for more rapid, dynamic, and iterative requirements processes for software, artificial intelligence, and data, and it references automated approaches for testing, evaluation, and deployment throughout acquisition and sustainment processes (U.S. Congress 2020). That direction is important, however, the deeper issue is whether the T&E enterprise can preserve rigor

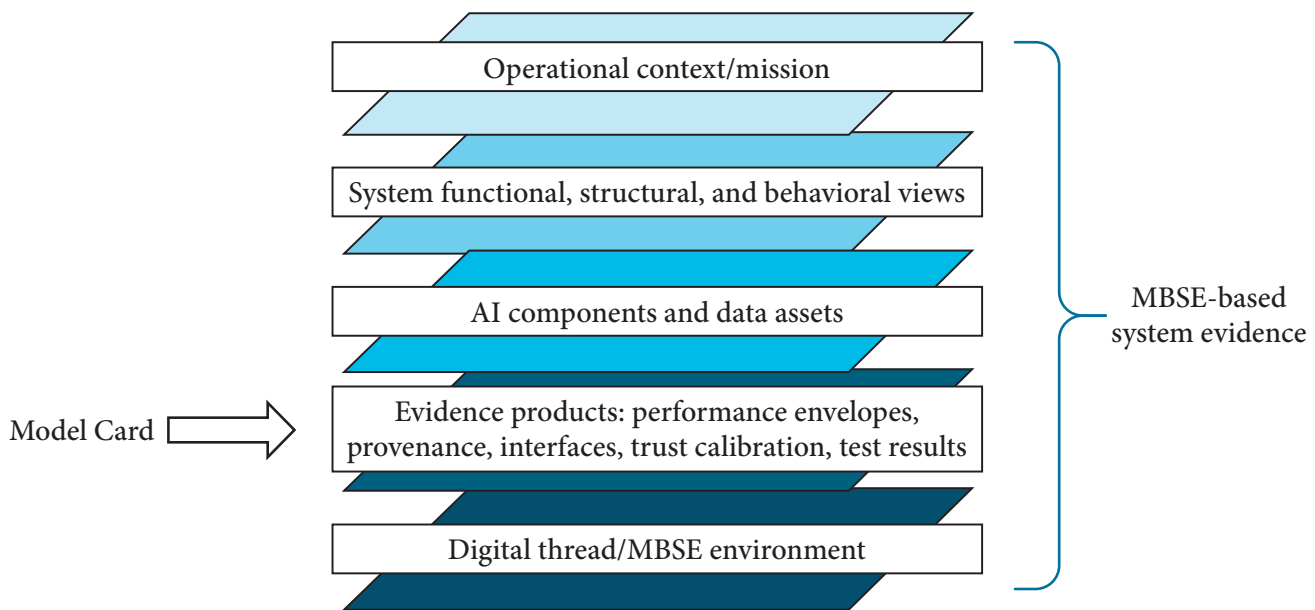


Figure 2. Conceptual transition from stand-alone model reporting toward model-based system evidence, in which AI performance is preserved across functional, structural, behavioral, and lifecycle views

when the system under test and the process producing it are both adaptive.

FROM MODEL CARDS TO SYSTEM MODELS

One of the clearest technical implications of this decade-long shift concerns documentation and evidence. In the late 2010s, model cards were proposed as short documents accompanying trained machine learning models, intended to summarize use cases, evaluation context, and performance characteristics (Mitchell et al. 2019). That was an important step toward transparency. But the model card abstraction now appears insufficient for many high-consequence systems. Modern practice already admits this in the renaming from “model” to “system” card for frontier models. System cards for frontier models have grown into lengthy technical documents covering capabilities, limitations, safety mitigations, evaluation methods, deployment controls, and sometimes third-party assessments (Hurst et al. 2024; Anthropic 2025). This expansion reflects the fact that the object being documented is no longer just a model in isolation. It is a system operating within a broader technical and governance context. Yet, system cards are still in the form of extensive model cards, not system models in the systems engineering sense, and they push the limits of the model card paradigm (Mitchell et al. 2019).

For systems engineering, the lesson is not simply that model cards need to be longer. The lesson is that AI evidence should increasingly be organized under a model-based engineering paradigm, depicted

in Figure 2. AI capabilities must be represented in functional, structural, and behavioral terms; tied to inputs, outputs, interfaces, relationships, constraints, and operating envelopes; and connected to the other systems, humans, and data infrastructures on which performance depends (Cody et al. 2020; Cody and Beling 2023). In other words, evidence of AI performance must migrate from an artifact-centric reporting style to a system-evidence architecture embedded in MBSE and digital engineering (Arndt et al. 2025). SERC research strongly supports this move through its repeated emphasis on digital twins, AI system architecting, model governance, system context, and interoperable digital ecosystems.

That transition will reshape how evidence is developed, presented, and preserved. Instead of asking for a single “card” that describes a model, T&E will increasingly need linked evidence products: operationally grounded performance envelopes, data provenance and curation records, interface contracts, adaptation logic, mission-level measures, trust calibration results, testbed outcomes, and digital-thread traceability across lifecycle stages. This is not a rejection of model cards; it is their absorption into a systems engineering worldview. What model cards were trying to do for standalone models, MBSE-style evidence frameworks will need to do for AI-generated systems.

CONCLUSION

From 2020 to 2030, T&E of AI can be read as a story of expanding assurance scope. In 2020, the community had to learn

how to treat machine learning models as system functions and to incorporate them into the engineering and T&E process as a special kind of software-linked component. By 2025, the community had learned that this was not enough: AI had to be treated in system context, across lifecycle adaptation, human-AI teaming, governance, resilience, and digital engineering ecosystems. By 2030, the next step is likely to be even more disruptive. As AI becomes a productive force in the creation and modification of system artifacts themselves, T&E will have to address not only AI-enabled systems but AI-generated systems: systems in which the generative processes that produce requirements, code, tests, models, evidence, and updates become part of the lifecycle being assured. That will require continuous T&E, stronger model-based evidence structures, lifecycle-aware governance, and a concept of engineering velocity that is disciplined rather than merely accelerated.

Seen this way, the SERC community’s contribution across the decade is not only a sequence of workshop topics or publications. It is a conceptual progression. First, it helped the systems engineering enterprise recognize AI as an engineered object inside the system. Then it helped reconceive AI-enabled systems as dynamic socio-technical systems requiring trust, adaptation, and continuous evidence. The task ahead is to complete the next transition: to build a T&E enterprise capable of assuring systems whose functions, evidence, and even component artifacts may increasingly be generated by AI. That is the frontier on which systems engineering now stands. ■

REFERENCES

- Anthropic. 2025. "System Card: Claude Opus 4 & Claude Sonnet 4." *Claude-4 Model Card*.
- Arndt, Craig, Valerie Sitterle, and Jeremy Doerr. 2025. *Knowledge based Metrics for Test and Design*. Acquisition Research Program.
- Bommasani, Rishi, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein et al. 2021. "On the Opportunities and Risks of Foundation Models." *arXiv preprint arXiv:2108.07258*.
- Chandrasekaran, Jaganmohan, Tyler Cody, Nicola McCarthy, Erin Lanus, Laura Freeman, and Kristen Alexander. 2024. "Testing Machine Learning: Best Practices for the Life Cycle." *Naval Engineers Journal* 136, no. 1-2: 249-263.
- Cody, Tyler, and Peter A. Beling. 2023. "A Systems Theory of Transfer Learning." *IEEE Systems Journal* 17, no. 1: 26-37.
- Cody, Tyler, Stephen Adams, and Peter A. Beling. 2022. "Empirically Measuring Transfer Distance for System Design and Operation." *IEEE Systems Journal* 16, no. 3: 4962-4973.
- Cody, Tyler, Stephen Adams, and Peter Beling. 2020. "Motivating a Systems Theory of AI." *INSIGHT* 23, no. 1: 37-40.
- Collopy, Paul, Valerie Sitterle, and Jennifer Petrillo. 2020. "Validation Testing of Autonomous Learning Systems." *INSIGHT* 23, no. 1: 48-51.
- Freeman, L., J. Kauffman, D. Sobien, T. Cody, and E. Lanus. 2022. "Best Practices for Addressing New Challenges in Testing and Evaluating Artificial Intelligence Enabled Systems." *The ITEA Journal of Test and Evaluation*.
- Freeman, Laura. 2020. "Test and Evaluation for Artificial Intelligence." *INSIGHT* 23, no. 1: 27-30.
- Gregory, Joe, and Alejandro Salado. 2024. "dTEMP: From Digitizing to Modeling the Test and Evaluation Master Plan." *Naval Engineers Journal* 136, no. 1-2: 134-146.
- Hurst, Aaron, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, A. J. Ostrow et al. 2024. "GPT-4o System Card." *arXiv preprint arXiv:2410.21276*.
- McDermott, Tom, Dan DeLaurentis, Peter Beling, Mark Blackburn, and Mary Bone. 2020. "AI4SE and SE4AI: A Research Roadmap." *INSIGHT* 23, no. 1: 8-14.
- McDermott, Tom, Kara Pepe, and Megan Clifford. 2024. "The Updated SERC AI and Autonomy Roadmap 2023." In *INCOSE International Symposium*, vol. 34, no. 1, pp. 1135-1148.
- McDermott, Tom. 2025. Review of *SE in the Era of Human-Machine Teaming Roadmap for AI and SE*. June 25. https://sercuarc.org/wp-content/uploads/2025/06/0840_Tom-McDermott_Updated-SERC-Roadmap-for-AI-and-SE.pdf.
- Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. "Model cards for model reporting." In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 220-229.
- NIST. 2023. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. Gaithersburg, US-MD: National Institute of Standards and Technology.
- Sandman, B., P. Wach, A. Salado, J. Gregory, T. Topcu, G. Kerr, and T. Sherburne. 2025. "Digital Transformation of Testing and Evaluation: Introducing Operation Safe Passage." In *Conference on Systems Engineering Research*, Long Beach, US-CA.
- Seif, Robert J., Zichong Yang, Ziran Wang, Laura Freeman, and Jitesh H. Panchal. 2024. "A Multi-Fidelity Approach to Testing and Evaluation of AI-Enabled Systems." In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 88353, p. V02BT02A044. American Society of Mechanical Engineers.
- SERC. 2020. Rep. *Research Workshop: AI4SE & SE4AI Report*. Systems Engineering Research Center.
- SERC. 2021. Rep. *Research Workshop: AI4SE & SE4AI Report*. Systems Engineering Research Center.
- SERC. 2022. Rep. *Research Workshop: AI4SE & SE4AI Report*. Systems Engineering Research Center.
- SERC. 2023. Rep. *Research Workshop: AI4SE & SE4AI Report*. Systems Engineering Research Center.
- SERC. 2024. Rep. *Research Workshop: AI4SE & SE4AI Report*. Systems Engineering Research Center.
- SERC. 2025. Rep. *Research Workshop: AI4SE & SE4AI Report*. Systems Engineering Research Center.
- Shadab, Niloofar, Tyler Cody, Alejandro Salado, and Peter A. Beling. 2024. "A Systems-Theoretical Formalization of Closed Systems." *IEEE Open Journal of Systems Engineering* 2: 26-37.
- Shadab, Niloofar, Tyler Cody, Peter Beling, and Alejandro Salado. 2023. "Product Herding for Intelligent Systems." In *Conference on Systems Engineering Research*, pp. 397-409. Cham: Springer Nature Switzerland.
- U.S. Congress. House. 2020. *Information technology acquisition: planning and oversight processes*. 10 U.S.C. § 4571. 116th Congress, 2nd session.

Viewpoint

Who Gets to Decide? An Organizational Decision-Making Lens for AI Governance

Jimmie G. McEver, jimmie.mcever@jhuapl.edu

Copyright ©2026 by Jimmie G. McEver. Permission granted to INCOSE to publish and use.

Systems engineers integrating AI into operational settings need better ways to think about delegation, oversight, and accountability. Command-and-control theory offers a practical framework — one that changes what you actually design, not just how you document it.

Organizations deploying AI are increasingly serious about governance. Responsible AI principles, human-in-the-loop requirements, model cards, and bias testing represent genuine attempts to address the consequences of integrating AI into operational environments. Even the most thoughtful governance efforts, though, tend to focus on a second-order question: how to manage this technology safely and ethically. The first-order question runs deeper: how should decision-making authority be structured in systems that include both human and artificial actors. Decision-making authority encompasses not just who gets to decide what, but how those decision-makers interact with each other and what information flows among them.

When this structural question is addressed clearly, the technology governance questions become more tractable. When it remains implicit, even sophisticated AI governance can miss the most important organizational changes. The system that needs to be designed here is not the

AI model or the platform it runs on; it is the organizational structure of authority, interaction, and information within which the technology operates. Our community is well-aligned to design at that level, but only if we recognize that the design problem in front of us is organizational rather than purely technical.

AI governance discussions have produced a wide range of useful tools, and many organizations are applying them seriously. Still, most focus on properties of the model or controls around its use. They are less effective at addressing the design question that underlies successful implementation: how should decision rights, interactions, and information flows be structured in a system that includes both human and artificial actors? A model card tells you what the model can do. It does not tell you who should be allowed to act on its outputs, under what conditions, or what happens when those conditions change.

Command and control (C2) is at its core a way of thinking about this problem — how decision-making is organized in complex, dynamic, high-stakes environments. Alberts and Hayes (2006) describe a C2 approach along three dimensions: the allocation of decision rights among actors, the patterns of interaction among those actors, and the distribution of information

across the enterprise. Change any of those, and you change how the organization senses, decides, and acts. For systems engineers, that three-part framework is useful well beyond national security command structures: it provides a practical way to think about collective decision-making in any complex sociotechnical system.

The U.S. national security community learned this lesson once already. When networked communications transformed military operations in the late 1990s, the most important effect went well beyond enabling existing command structures to move faster. It was that entirely new structures of coordination and authority became viable — structures that could not have functioned without shared situational awareness and distributed information (Cebrowski and Garstka 1998). The lesson was not about technical communications networks *per se*. It was that technology can expand the space of organizational designs, and that institutions need frameworks to navigate that expanded space deliberately rather than stumbling into or reacting to it (Alberts et al. 2010).

The AI challenge is structurally similar. AI does not simply make existing decision processes faster; it changes where decisions can be made, by whom, and on what informational basis. When an AI-enabled

system is integrated into an operational environment — whether in healthcare, logistics, finance, defense, or engineering activities — we are making a design choice about command and control, whether we recognize it or not. We are deciding how decision rights are allocated among human and machine actors, how those actors interact, and what information each has access to. The technology is new, but the underlying systems question is not.

Every consequential AI deployment embeds a command-and-control design choice. The question is whether you make that choice deliberately.

This reframing has immediate practical implications. Discussions of AI autonomy often collapse into a binary: either a human is in the loop or the system is autonomous. Authority is rarely structured so simply. Between full manual control and unconstrained autonomy lies a wide design space: systems in which humans approve, monitor, override, set intent, handle exceptions, or retain decision authority only for specific classes of action. The real question is not whether delegation occurs. It is how delegation is structured so that performance remains aligned with intent as conditions change.

Conditions do change, and when they do, the allocation of decision authority between human and AI actors may need to shift. A C2 structure for an AI-enabled system is not something you configure once and leave in place. It is a dynamic design that must account for transitions — moments when authority needs to move, when interaction patterns need to change, or when information flows need to be redirected. Those transitions are themselves decisions, and they need to be part of the system's design from the start. An organization that expects its people to reallocate decision authority under operational pressure, but has never practiced doing so, is in the same position as a military unit whose mission rehearsal leaves out the hard parts: the rehearsal builds false confidence precisely where real confidence is most needed.

Consider an example: A maintenance enterprise uses AI to prioritize inspections and repairs across a distributed fleet. A conventional responsible-AI review would begin with the model: is it accurate, fair, documented, and tested for the intended use? Those are necessary questions, but they are not enough. A C2 approach asks a different set of questions and produces different design choices. First, decision rights: the AI recommends a prioritized repair sequence, but a fleet manager retains authority to override when operational

commitments conflict with model-optimal sequencing—and those overrides are logged, not because the manager is untrusted, but because the pattern of overrides is itself a signal about where the model's operating assumptions may be breaking down. Second, interaction pattern: rather than a simple approve/reject interface, the system is designed so that when model confidence drops below a threshold, the interaction shifts from monitoring to active approval — the human is brought into the loop precisely when the AI's basis for recommendation weakens. That transition — from monitoring to active approval — is exactly the kind of reallocation that needs to be designed explicitly and rehearsed, so that when it happens under operational pressure, the people involved know what is expected of them and why. Third, information distribution: maintenance technicians see not just the recommendation but the key factors driving it, so they can flag cases where local conditions (a part that has been running hot, a known environmental exposure) are not reflected in the model's inputs. Each of these is a design choice that a responsible-AI checklist would not surface but that a C2 framework makes explicit.

The same logic applies to AI-enabled systems more broadly. The right question is not 'autonomous or supervised?' but 'what command-and-control structure fits this system, in this operational context, at this level of consequence and uncertainty?' Some decisions should be delegated aggressively because AI can act at the necessary speed or scale. Others should remain with human decision-makers not because AI is technically incapable, but because the accountability structure, ethical burden, or operational context requires a human to own the decision. For practitioners, that is a systems design problem, not just a policy overlay.

A C2 perspective also reframes what organizations mean when they say they need to 'trust' an AI system. In most governance discussions, trust collapses into one of two things: a feeling of general confidence that the system works, or a technical judgment that the system has been adequately trained, tested, and validated. Both matter, but neither is the real question. Whether an AI system is technically ready for a given role is a technical question with technical answers. Whether the organization should delegate that role to the AI is a command-and-control question. The first informs the second but does not answer it. Organizations already make this distinction for human decision-makers. A surgeon may be fully trained and credentialed, but the hospital still decides what procedures

that surgeon is authorized to perform independently, which require a second opinion, and what triggers a review. The same structural logic applies to AI. What decisions is the system trusted to make, and under what conditions? What evidence supports that judgment, and what happens when the evidence changes? Trust, framed this way, is not a sentiment to be cultivated or a technical threshold to be cleared. It is an organizational decision about the allocation of authority, one that should be specified, bounded, monitored, and, when necessary, revoked.

None of this eliminates the technical challenges. AI systems do fail in ways that differ from human failure: they can be confidently wrong, brittle outside their training or testing conditions, and difficult to interpret when they fail. Those differences matter and require serious technical attention, but do not solve the underlying governance problem. The practical challenge is still how to structure authority, accountability, and information flow in systems that include both human and artificial decision-makers. Our natural instinct is to reach for the tools we know — requirements, verification procedures, architecture frameworks, and others — and those tools will be needed. But they address the wrong level of the problem if the organizational decision-making structure has not been designed first. That is exactly where a C2 framework is useful — and where a purely model-centric governance approach falls short.

For systems engineers, the most useful way to think about human-AI teaming is not as a choice between human and machine, or even between supervised and unsupervised operation. It is as a question of system design: what decision rights should be assigned to which actors, how should those actors interact, what information should each have, and how should performance and accountability be monitored over time? Command-and-control theory does not answer those questions automatically. It does provide a disciplined way to ask them — and a way to recognize when an organization's current answers are inconsistent, incomplete, or no longer appropriate for the operational conditions it faces. At a moment when many organizations are rushing to operationalize AI, that is more than a useful starting point. It may be the most important design discipline they are missing.

THREE QUESTIONS FOR PRACTITIONERS DESIGNING AI-ENABLED SYSTEMS

- What decision rights are being delegated, and which remain with humans — and under what conditions should that allocation shift?

- What interaction pattern governs the human-AI relationship: approval, monitoring, exception handling, override, or intent-setting - and how does it adapt when model confidence or operational conditions change?
- What information does each actor have, and where could information asymmetry create brittle or unsafe behavior — or mask a degradation in system performance? ■

REFERENCES

- Alberts, David S., and Richard E. Hayes. 2006. *Understanding Command and Control*. Washington, US-DC: CCRP Publication Series.
- Alberts, David S., Reiner K. Huber, and James Moffat. 2010. *NATO Network Enabled Capability C2 Maturity Model*. Washington, US-DC: CCRP Publication Series. The author was a contributing member of the NATO SAS-065 research group that produced this work.
- Cebrowski, Arthur K., and John J. Garstka. 1998. "Network-Centric Warfare: Its Origin and Future." *Proceedings of the U.S. Naval Institute* 124 (1): 28-35.

ABOUT THE AUTHOR

Jimmie G. McEver holds a PhD in physics from Georgia Institute of Technology and a master of public policy (MPP) from Harvard University's John F. Kennedy School of Government. He is a principal scientist in the asymmetric operations sector of the Johns Hopkins University Applied Physics Laboratory, deputy director for Technical Operations at INCOSE, and deputy director of the AIAA Information Systems Group, and chair of the AIAA Defense Working Group. His work focuses on the application of complex systems science to cyber risk, command and control, and the organizational and operational implications of emerging technologies.

MBSE YOU CAN TRUST

Enterprise-Grade Security & Compliance

- SOC 2 Type 2
- ISO 27001
- ISO 9001

INNOSLATE[®]
Developed By **SPEC INNOVATIONS**

[Learn More](#)

MBSE with SysML V2 & CATIA Magic

Expert Webinar Series.

On Demand and Live. With Live Q&A Sessions



November 6, 2025 -
December 31, 2026

DS DASSAULT
SYSTEMES

Alignment is the Advantage

Train for Skills.
Align for Capabilities.

Caltech CTME designs custom learning that aligns engineers, leaders, and organizations around shared systems thinking—so teams move faster, smarter, together.

NEW CERTIFICATE PROGRAMS

Transitioning Models to SysML v2

MBSE Fundamentals Using SysML v2

AI-Assisted MBSE

Caltech

Center for Technology &
Management Education

CTME.CALTECH.EDU



THE BEST ENGINEERS ALLOW FOR A LITTLE GIVE.

Become an INCOSE volunteer today!

incose.org/volunteer

