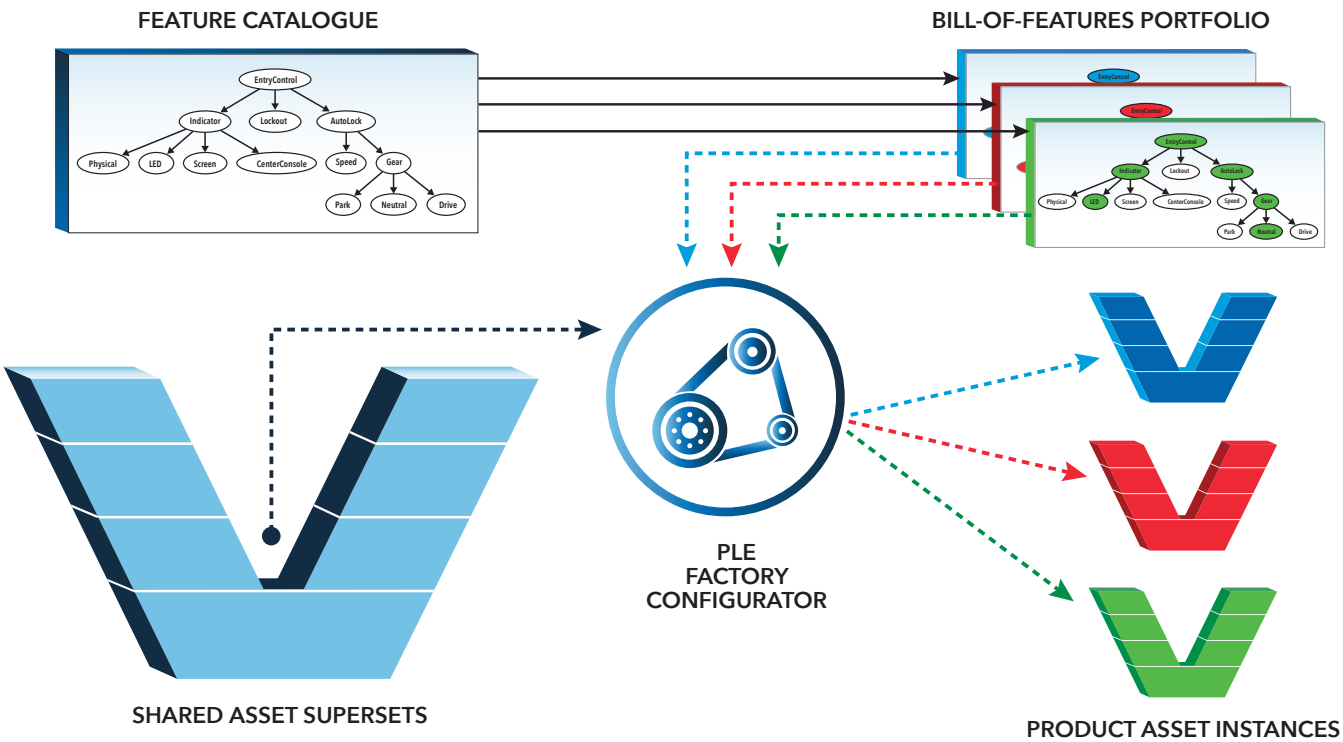
# INSIGHT

## This Issue's Feature:

## Cyber Secure and Resilient Approaches with Feature-Based Product Line Engineering




**SEPTEMBER 2020**
VOLUME 23 / ISSUE 3

This issue is sponsored by the Lockheed Martin Corporation.

**A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING**

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

# *Call for Papers*

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

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

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

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

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

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

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

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

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

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

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

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

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

http://mc.manuscriptcentral.com/SYS

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

INSIGHT

A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

SEPTEMBER 2020 VOLUME 23 / ISSUE 3



# Inside this issue

# About This Publication

## INFORMATION ABOUT INCOSE

INCOSE's membership extends to over 18,000 individual members and more than 100 corporations, government entities, and academic institutions. Its mission is to share, promote, and advance the best of systems engineering from across the globe for the benefit of humanity and the planet. INCOSE charters chapters worldwide, includes a corporate advisory board, and is led by elected officers and directors.

For more information, click here:
The International Council on Systems Engineering
(www.incose.org)

## OVERVIEW

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





## ARTICLE SUBMISSION

INSIGHT@incose.org

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

- March 2020 issue – 2 January
- June 2020 issue – 2 April
- September 2020 issue – 1 July
- December 2020 issue – 1 October

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


ISSN 2156-485X; (print) ISSN 2156-4868 (online)

## ADVERTISE

**Readership**
***INSIGHT*** reaches over 18,000 individual members and uncounted employees and students of more than 100 CAB organizations worldwide. Readership includes engineers, manufacturers/purchasers, scientists, research & development processionals, presidents and CEOs, students and other professionals in systems engineering.

| Issuance | Circulation |
|---|---|
| 2020, Vol 23, 4 Issues | 100% Paid |

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

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

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

- Asia & Australia corporatesalesaustralia@wiley.com
- Europe, Middle East & 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:**
Dan Nicholas, Associate Director – Sciences, Corporate Sales Wiley
PHONE: +1 716-587-2181
E-MAIL: dnicholas@wiley.com

## CONTACT

**Questions or comments concerning:**

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

**Advertising—*please contact:***
Susan Blessing, Senior Account Manager Sciences
Sciences, Corporate Sales
MOBILE: 24/7 201-723-3129
E-MAIL: sblessin@wiley.com

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

## ADVERTISER INDEX



*INSIGHT volume 23, no. 3 is sponsored by the Lockheed Martin Corporation.* LOCKHEED MARTIN

## CORPORATE ADVISORY BOARD — MEMBER COMPANIES

321 Gang, Inc.
Aerospace Corporation, The
Airbus
Airbus Defense and Space
AM General LLC
Analog Devices, Inc.
Analytic Services
Aras Corp
Aviation Industry Corporation of China, LTD
BAE Systems
Bechtel
Beckton Dickinson
Boeing Company, The
Bombardier Transportation
Booz Allen Hamilton Inc.
C.S. Draper Laboratory, Inc.
Carnegie Mellon University Software Engineering Institute
Change Vision, Inc
Colorado State University
Cornell University
Cranfield University
Cubic Corporation
Cummins, Inc.
CYBERNET MBSE
Defense Acquisition University
DENSO Create, Inc.
Drexel University
Eindhoven University of Technology
Embraer S.A.
ENAC
Federal Aviation Administration (U.S.)
Ford Motor Company
Fundacao Ezute
General Dynamics
General Motors
George Mason University
Georgia Institute of Technology
IBM
Idaho National Laboratory
ISAE SUPAERO
ISDEFE
ISID Engineering, LTD
iTiD Consulting, Ltd
Jacobs Engineering
Jama Software
Jet Propulsion Laboratory
John Deere & Company
Johns Hopkins University
KBR, Inc.
KEIO University
L3 Harris
Leidos
Lockheed Martin Corporation
Los Alamos National Laboratory
ManTech International Corporation
Maplesoft
Massachusetts Institute of Technology
MBDA (UK) Ltd.
Missouri University of Science & Technology
MITRE Corporation, The
Mitsubishi Aircraft Corporation (Mitsubishi Heavy Induftries Group)
National Aeronautics and Space Administration
National Security Agency - Enterprise
Naval Postgraduate School
Nissan Motor Co, Ltd
No Magic/Dassault Systems
Noblis
Northrop Grumman Corporation
Penn State University
Perspecta (formerly Vencore)
Prime Solutions Group, Inc.
Project Performance International
Raytheon Corporation
Roche Diagnostics
Rolls-Royce
Saab AB
Safran Electronics and Defence
SAIC
Sandia National Laboratories
Shell
Siemens
Sierra Nevada Corporation
Singapore Institute of Technology
Skoltech
SPEC Innovations
Stellar Solutions
Stevens Institute of Technology
Strategic Technical Services
Swedish Defence Materiel Administration
Systems Engineering Directorate
Systems Planning and Analysis
Thales
TNO
Trane Technologies
Tsinghua University
TUS Solution LLC
UK MoD
United Technologies Corporation
University of Arkansas
University of California San Diego
University of Connecticut
University of Maryland
University of Maryland, Baltimore County
University of Michigan, Ann Arbor
University of New South Wales, The, Canberra
University of Southern California
University of Texas at Dallas
University of Texas at El Paso, The
US Department of Defense, Deputy Assistant Secretary of Defense for Systems Engineering
Veoneer, Inc
Vitech Corporation
Volvo Construction Equipment
Woodward Inc
Worcester Polytechnic Institute- WPI
Zuken, Inc

# FROM THE EDITOR-IN-CHIEF

**William Miller,** insight@incose.org

*INSIGHT*'s mission is providing informative articles advancing the systems engineering practice state. The intent is accelerating knowledge dissemination closing the gap between the practice state and the research state as *Systems Engineering*, the Journal of INCOSE, also Wiley published, captures. INCOSE thanks corporate advisory board member Lockheed Martin for sponsoring *INSIGHT* in 2020 and welcomes additional sponsors, who may contact the INCOSE marketing and communications director at marcom@incose.org.

The *INSIGHT* September 2020 issue's theme is a joint INCOSE Systems Security Engineering (SSE) Working Group and Product Line Engineering (PLE) Working Group project to bring systems security into product line design. We thank theme editors Beth Wilson and Bobbi Young and the authors for their contributions. The SSE Working Group's mission is providing systems engineers and systems engineering effective sustainable system functionality means and methods under advanced adversarial attack. Their objectives are instilling systems engineering responsibility for sustainable systems functionality facing intelligent, determined, and highly competent system adversaries; facilitating responsibility assimilation and dispatch; and instigating self-sustaining cross-community involvement between systems engineers, security engineers, and system security standards. The PLE Working Group's mission is promoting PLE and related systems engineering best practices and to coordinate activities around PLE at the INCOSE level and share results. The working group's objectives are helping our members acquire knowledge comparing to the state-of-art, share concerns, experiences, good practices, and traps to avoid while providing guidelines to set up and evolve organization PLE.

Young and Wilson's article introduced the theme issue and the articles exploring the intersection between systems security engineering and product line engineering. The focus includes product line system security implementation techniques, product line architectures patterns addressing systems security, and security and resilient product line product variation management approaches. Two articles follow the lead article addressing SSE and PLE vocabulary, respectively. Young, Darbin, and Clements then describe how applying both SSE and PLE achieves a productive union, introducing the "securing the PLE factory" technique. Williams, Moss, Bataller, and Hassell describe how to apply cyber resiliency analysis to product line architectures, introducing the "cyber resiliency wheel" technique. Dove addresses product line resource variation security issue detection and mitigation p-patterns. Adejokun and Siok describe how to identify a product line design's security requirements, introducing a security profile developing and evolving a secure product line aligned with industry security standards. Teaff describes how to apply rule-based system security verification in product line variants. Agrawal describes how to leverage model-based systems engineering capturing systems security concerns while developing the product line architecture. Navas, Voirin, Paul, and Bonnet address a model-based approach to systems and cybersecurity co-engineering in a product line context. Finally, Hause describes an integrated security views set for the Unified Architecture Framework (UAF) defining security goals and requirements implemented throughout the architecture.

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

Front cover image credit: INCOSE Product Line Engineering Working Group

# Exploring Cyber Secure and Resilient Approaches with Feature-Based Product Line Engineering

**Bobbi Young,** bobbi.young@raytheon.com and **Beth Wilson,** wilsondrbeth@aol.com


**ABSTRACT**
The INCOSE Systems Security Engineering Working Group and Product Line Engineering Working Group completed a joint project exploring cyber secure and resilient approaches with feature-based product line engineering. The project output results are in this INCOSE *INSIGHT* theme issue (Volume 23, Issue 3). This article introduces the theme issue and the articles exploring the intersection between systems security engineering and product line engineering. The focus includes techniques for implementing systems security as part of a product line design, patterns for product line architectures addressing systems security, and variation management approaches for security and resilient product line products.

## INTRODUCTION

The INCOSE Systems Security Engineering (SSE) and Product Line Engineering (PLE) Working Groups launched a joint project in April 2018 to explore the intersection of SSE and PLE. Product line design represents a unique opportunity to address cybersecurity and cyber resiliency in a way that benefits all the products in the portfolio. By addressing systems security within the product line core team, the product line programs receive secure and resilient products and shared assets. When systems security repeats for every system using a product line product, it is a potentially incomplete and wasted effort. Applying systems security techniques before generating the variations is more cost effective and likely to have better outcomes.

The project vision was to bring systems security into product line design. The team focused on cyber secure and resilient approaches to feature-based management in product line implementation. The effort addressed the vision through three goals:

1. Identify and/or develop techniques for implementing systems security as part of product line design
2. Identify and/or develop patterns for product line architectures addressing systems security
3. Identify and/or develop variation management approaches for secure and resilient product line products

The articles in this theme issue address the intersection of SSE and PLE for product design, implementation, and migration over time. The authors collectively address architecture development, requirements, and verification using model-based systems engineering, SSE/PLE techniques, and patterns.

These articles show how applying systems security to product line design can provide a cost-effective focus on critical mission threads with recursive analysis at different design levels. Model-based systems engineering approaches capture systems security concerns while developing a digital environment to promote reuse and enable multi-discipline teams to co-engineer the product line.

The techniques developed by this project show how SSE analysis can apply to interim systems engineering products to address security concerns inside and outside the core product line. These product line architecture techniques provide continuous analysis of the cyber-attack surface and address critical mission threads that may be different across the product line. Feature-based variation management applied to the product line products results in security requirements in the requirements specification, threats in the operational environment description, security-related scenarios in the Concept of Operations, and test cases in the verification plan. Changes to the threat environment require evaluating the impact on the instantiated products across the product line.

## INTRODUCTIONS TO SSE AND PLE VOCABULARY

When the team started this joint project, the initial effort was to make sure SSE experts were using the correct PLE vocabulary and vice versa. As the authors drafted the initial versions of their papers, introductory material related to terminology repeated across the articles. We decided to include two introductory articles in this issue to provide essential vocabulary to better

understand the collaboration between the two working groups:

- Introduction to Systems Security Engineering Vocabulary
- Introduction to Product Line Engineering Vocabulary

### SYSTEM SECURITY ENGINEERING AND FEATURE-BASED PRODUCT LINE ENGINEERING: A PRODUCTIVE MARRIAGE

Bobbi Young, Rowland Darbin, and Paul Clements describe how to apply both SSE and PLE to achieve the best of each. They introduce the "Securing the PLE Factory" technique that creates a product line architecture to support transferring product line products designed to be cyber secure and cyber resilient to multiple programs. When the security-related content is the same for all product line members, the PLE factory can design and test these products once as part of the product line. The product line team can maintain the expertise necessary to address evolving threats benefitting all the receiving programs. The technique includes temporal baseline management to synchronize the evolution of the digital assets at each security level as development occurs over time. This allows for mixed data sharing and multiple security levels across the receiving programs. The article also addresses "Applying PLE to Security" which considers the need to address the impact of variant related security content for each possible product solution within a product line.

### ENGINEERING A CYBER RESILIENT PRODUCT LINE

Paula Moss, Susan Bataller, Patrice Williams, and Suzanne Hassell describe how to apply cyber resiliency analysis to product line architectures. They introduce the "Cyber Resiliency Wheel" technique to develop a cyber secure and cyber resilient product family architecture facilitating building block reuse of product modules. The cyber resiliency wheel's key advantage is applying the analysis on interim architecture products at different stages of product line development and maintenance.

### SECURITY ISSUE DETECTION AND MITIGATION PATTERNS FOR PRODUCT LINE RESOURCE VARIATION

Rick Dove describes how to represent the product line design as an agile architectural pattern. He addresses cyber-physical-social system products using techno-social patterns for detecting and mitigating security issues. The techno-social contract concept provides a method to implement defense-in-depth as an emergent security behavior adapting to a varying threat.

### EFFECTIVE SYSTEMS SECURITY REQUIREMENTS IN PRODUCT LINE ENGINEERING

Ademola Adejokun and Michael Siok describe how to identify security requirements in product line design. They introduce a security profile to develop and evolve a secure product line aligned with industry security standards. Beyond the confidentiality, integrity, and availability requirements addressed as security tenets, the technique also addresses requirements for configuration management and the deployment environment.

### RULE-BASED VERIFICATION OF SYSTEM SECURITY USING FEATURE-BASED PRODUCT LINE ENGINEERING

Jim Teaff describes how to apply rule-based verification of system security in product line variants. The rule base is created in the PLE factory to provide continuous cyber-attack surface analysis and digitize cybersecurity and cyber resiliency verification rules. The technique validates instantiated variants to ensure systems security capabilities delivery to the fielded systems.

### LEVERAGING A SYSTEM MODEL TO INITIATE SECURITY ARCHITECTURE DEVELOPMENT FOR PRODUCT LINES

Angel Agrawal describes how to leverage model-based systems engineering to capture systems security concerns while developing the product line architecture. He identifies security related model elements using stereotypes defining a security architecture. The directed association relationships between threat activities and countermeasure activities result in a countermeasure coverage matrix to identify coverage and gaps for product line products.

### TOWARDS A MODEL-BASED APPROACH TO SYSTEMS AND CYBERSECURITY CO-ENGINEERING IN A PRODUCT LINE CONTEXT

Juan Navas, Jean-Luc Voirin, Stephane Paul, and Stephane Bonnet describe a security-by-design co-engineering approach to product line development using the Arcadia model-based method. The technique integrates cybersecurity functions with the product line architecture to define protected services and patterns for managing sensitive data. The resulting model elements superset includes cybersecurity capabilities for configuring into the product line products.

### INTEGRATING SECURITY INTO ENTERPRISE ARCHITECTURE WITH UAF AND PLE

Matthew Hause describes an integrated security views set for the Unified Architecture Framework (UAF) defining security goals and requirements implemented throughout the architecture. The UAF security views can address cybersecurity and cyber resilience in the product line architecture capturing the variations. The UAF security measures provide a quantitative and qualitative method to analyze security alternatives. ■

### ABOUT THE AUTHORS

**Dr. Bobbi Young** is a systems engineering fellow and certified architect at Raytheon Technologies. She currently leads an internal research and development project focusing on adoption of product line engineering across the business. She is regarded throughout Raytheon Technologies as an expert in Product Line Engineering and MBSE and co-chairs an MBSE technical interchange group. She is a member of the INCOSE Product Line Engineering Working Group, Architecture Working Group, and MBSE Working Group. Bobbi is also an adjunct professor at Worcester Polytechnic Institute as an MBSE instructor and has co-authored a book on object oriented analysis and design. She is a US Navy Commander (ret).

**Beth Wilson** earned her PhD in electrical engineering from the University of Rhode Island. She is retired from Raytheon where she worked for 33 years as a design engineer, program manager, research scientist, functional manager, systems architect, risk manager, and test director on sonar, satellite, and radar programs. Dr. Wilson is currently an adjunct professor at Worcester Polytechnic Institute in their master of science in systems engineering program. She is an INCOSE Expert Systems Engineering Professional (ESEP), INCOSE Certification Advisory Group (CAG) member, and co-chair for the INCOSE Systems Security Engineering Working Group.

# Introduction to Systems Security Engineering Vocabulary

**INCOSE Systems Security Engineering Working Group,** https://www.incose.org/incose-member-resources/working-groups/analytic/systems-security-engineering


■ **ABSTRACT**
Systems security protects a system from misuse and malicious behavior and makes sure the system can deliver value even under adverse conditions in cyberspace. This article identifies essential systems security vocabulary including systems security engineering, cyberspace, cyber-attack, cybersecurity, and cyber resiliency. This terminology overview supports effective collaboration with other INCOSE working groups.

## INTRODUCTION

At the beginning of the INCOSE joint project between the systems security and product line working groups, the team spent some time reviewing vocabulary. For those experienced in systems security concepts, learning the nuances of feature-based product line engineering, and vice versa, was important to make progress. This article summarizes systems security concept nuances and highlights the essential vocabulary the team identified.

Systems security engineering is a discipline focused on ensuring the system of interest can deliver value even under adverse conditions. Table 1 from the INCOSE handbook shows a more formal definition. Using this definition, **systems security engineering** is a discipline to engineer a system of interest to ensure that system "can function under disruptive conditions associated with misuse and malicious behavior."

Before we had computers, systems security protected the system from misuse that could intentionally or unintentionally disrupt effective operation. Guards and gates kept the enemy out. They authorized people who got in (and things they carried) to be there. If someone got in who should not be there or had malicious intent, the guards made sure they could not damage anything inside or remove anything valuable.

Systems security's emphasis expanded when adversaries could attack from a distance (drop bombs from a plane) or steal something valuable without removing it (take a picture or make a recording). The focus remained on physical security and the protection of hardware-intensive systems.

*Table 1. Definitions related to systems security engineering*

| Term | Definition | Source |
|---|---|---|
| Systems security engineering | Ensuring a system can function under disruptive conditions associated with misuse and malicious behavior | INCOSE SE Handbook (Walden, et al. 2015) |
| Cyberspace | Interconnected digital environment of networks, services, systems, and processes | ISO/IEC 27102: 2019 (ISO 2019) |
| Cyber-attack | Malicious attempts to exploit vulnerabilities in information systems or physical systems in cyberspace and to damage, disrupt or gain unauthorized access to these systems | ISO/IEC 27102: 2019 (ISO 2019) |
| Cybersecurity | Safeguarding of society, people, organizations, and nations from cyber risks | ISO/IEC 27102:2019 (ISO 2019) |
| Cyber resiliency | Ability to anticipate, withstand, recover from, and adapt to adverse conditions, stresses, attacks, or compromises on systems that use or are enabled by cyber resources | NIST SP 800-160 (NIST 2019) |

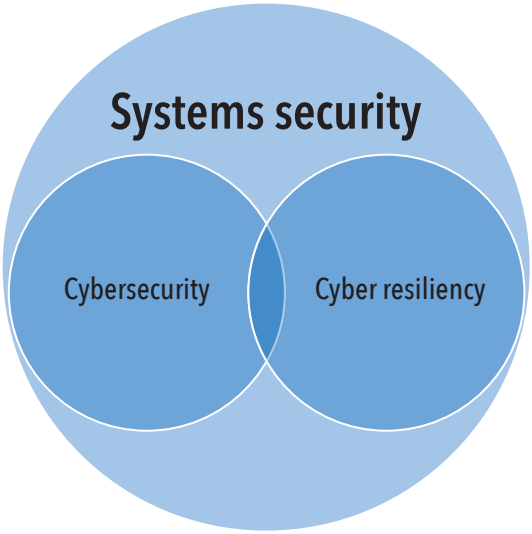



*Figure 1. Systems security engineering includes both cybersecurity and cyber resiliency*

## SYSTEMS SECURITY IN CYBERSPACE

Systems security's focus expanded again to protect the information stored in computers. The goal stayed the same—to protect the system from unauthorized entry, to protect what is valuable if there is an entry, and to resist the disruption of the system's operation. The challenge now is systems must operate in **cyberspace**, defined in table 1 as an "interconnected digital environment of networks, services, systems, and processes" representing the proposed definition in ISO 2019.

The focus on cyberspace is important. This interconnected digital environment soon became much more than computers and their software. Cyber-physical systems represent a global interaction of people, software, and hardware. The Internet of Things connects our smart phone applications to our physical devices. Delivering value under adverse conditions now needs to include **cyber-attacks**, defined in ISO/IEC 27102:2019 as "malicious attempts to exploit vulnerabilities in information systems or physical systems in cyberspace and to damage, disrupt or gain unauthorized access to these systems."

An important systems security nuance in figure 1 is showing both cybersecurity and cyber resiliency. **Cybersecurity** as defined in ISO/IEC 27102:2019 is "safeguarding of society, people, organizations, and nations from cyber risks." **Cyber resiliency**, defined in NIST 800-160, is the "ability to anticipate, withstand, recover from, and adapt to adverse conditions, stresses, attacks, or compromises on systems that use or are enabled by cyber resources." We must protect the system from a cyber-attack. We must also ensure mission success in the event of a cyber-attack.

As noted in the diagram, some overlap exists between cybersecurity and cyber resiliency. Some techniques, such as encryption and access control, help to both prevent an attack and provide resiliency if attacked.

Designing cyber secure and cyber resilient systems requires a risk-based assessment of how best to deliver value in cyberspace. We identify the critical mission threads and business use cases representing key value delivery aspects. We identify cybersecurity techniques to protect the critical system assets from potential threats. We identify cyber resiliency techniques and approaches to deliver value when the system faces adverse conditions (including a cyber-attack). ■

## REFERENCES


- ISO (International Organization for Standardization). 2019. ISO/IEC 27102:2019. Inforamtion Security Management—Guidelines for Cyber-Insurance. Geneva, CH: ISO.
- NIST 2019. "NIST SP 800-160 "Developiong Cyber Resilient Systems: A Systems Security Engineering Approach". Gathersburg, US-MD. National Institute of Standards and Technology.
- Ross, R, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid. 2019. "Developing Cyber Resilient Systems: A Systems Security Engineering Approach." *National Institute of Standards and Technology Special Publication 800-160* 2:1-229. doi:10.6028/NIST.SP.800-160v2.
- Walden, D. D., G.J. Roedler, K.J. Forsber, R.D. Hamelin, and T.M. Shortell. 2015. *INCOSE Systems Engineering Handbook*. San Diego, US-CA: Wiley. https://www.wiley.com/en-us/INCOSE+Systems+Engineering+Handbook%3A+A+Guide+-for+System+Life+Cycle+Processes+and+Activities%2C+4th+Edition-p-9781118999400.

# Introduction to Product Line Engineering Vocabulary

**INCOSE Product Line Engineering Working Group,** https://www.incose.org/incose-member-resources/working-groups/analytic/product-lines


**ABSTRACT**
Feature-based product line engineering (PLE) designs a portfolio of products using variation management to maximize value of commonality and manage differences. This article identifies essential product line vocabulary including product line engineering, shared assets, feature catalogue, bill-of-features, PLE factory configurator, and product asset instances. This terminology overview supports effective collaboration with other INCOSE working groups and summarizes the INCOSE PLE Primer.

## INTRODUCTION

Product Line Engineering (PLE) is a systems engineering discipline to engineer a portfolio of products using variation management techniques to take advantage of products' similarities while managing their differences. This approach enables large organizations to maximize a portfolio's commonality value by embracing variation as a value driver instead of a justification for divergence.

Feature-based PLE is a specific and well-documented PLE form supported by industrial-strength commercial automation and using features to express the differences among the products. Maximizing commonality requires encapsulating and abstracting the differences across the portfolio using a defined set of features enabling variation within shared assets ranging from systems development artifacts (requirements, systems models, and software) to the deployment environment (configuration parameters, bills of materials, and end-user documentation). Using these feature-based variation mechanisms enables organizations to encapsulate and abstract access to system resources from a single logical expression describing the services offered to the external environment. Ultimately, this enables a family of systems' risk management to scale with a normalized capability offering instead of each system.

## FEATURE-BASED PRODUCT LINE ENGINEERING FACTORY

Organizations utilizing Feature-based PLE adopt a *factory* approach to building their products. The factory is a conceptual construct showing how the various Feature-based PLE aspects interact with each other (INCOSE 2019). Figure 1 illustrates:

- **Shared assets** are the "soft" artifacts supporting the creation, design, implementation, deployment, and operation of products. A shared asset can be any artifact representable digitally: requirements, design models, source code, test cases, bills of materials, wiring diagrams, documents, user manuals, installation guides, and more. They either compose a product or support the engineering process to create a product. Shared assets can be configured and shared across the product line.

  A shared asset used in the product line exists as a superset and includes

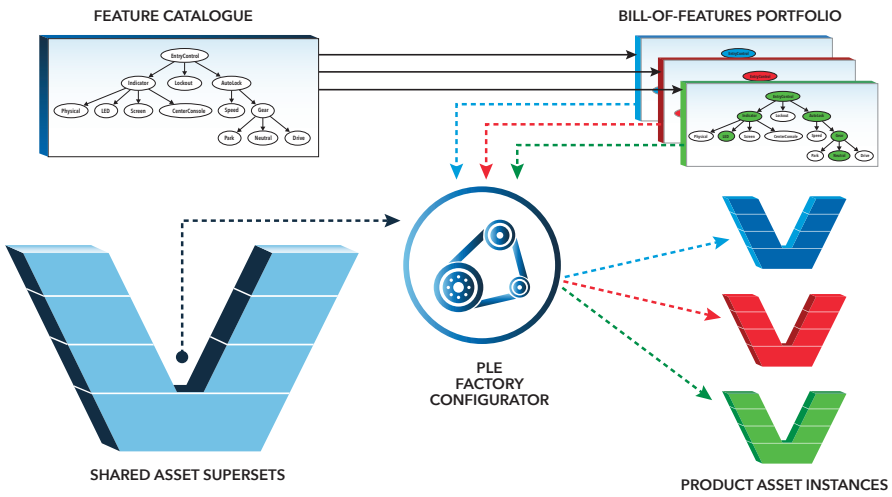



*Figure 1. The Feature-based PLE Factory*

any asset content used in any product. There is no asset content duplication or replication. This duplication elimination is where Feature-based PLE derives its savings.

The shared asset supersets contain variation points, which identify content included, omitted, or configured according to the product's feature selections. A statement of the product's distinguishing characteristics—its features—"exercises" these variation points (configuring the content associated with each variation point to meet the product needs).

A key aspect of Feature-based PLE is consistent and traceable variation treatment across all shared asset types. Feature choices are the basis of a common variation language across all disciplines and at all organization levels. This resolves the confusion brought about by different disciplines each using its own approach to variation.

- The **Feature Catalogue** captures the features available for each product to select. A feature is a distinguishing characteristic describing how the product line members differ from each other. This provides a common language and definition of the product line's variation scope for everyone throughout the organization.
- The Bill-of-Features specifies the features selected for each product in the product line.
- The **PLE Factory Configurator** is an automated software tool applying a Bill-of-Features to all shared assets. It evaluates each variation point to determine if it should include that variation point's content.
- The PLE Factory produces as output **Product Asset Instances,** each one containing only the shared asset content suited for one product in the product line. Together, they constitute the artifact set for one product in the product line. Engineers now work on the shared asset supersets, the Feature Catalogue, and the Bills-of-Features, handling change and evolution systematically through well-defined governance procedures.

Once established, the PLE Factory instantiates, rather than manually creates, engineering assets for products. Feature-based PLE transforms the task of engineering products into the much more efficient task of producing a single system: The PLE Factory itself. ■

## REFERENCES

- INCOSE International Working Group for Product Line Engineering. 2019. "Feature-based Systems and Software Product Line Engineering: A Primer." Available at https://connect.incose.org/Pages/Product-Details.aspx?Product-Code=PLE_Primer_2019

SEPTEMBER 2020
VOLUME 23/ ISSUE 3

# System Security Engineering and Feature-based Product Line Engineering: A Productive Marriage

**Bobbi Young,** bobbi.young@raytheon.com; **Rowland Darbin,** rowland.darbin@gd-ms.com; and **Paul Clements,** pclements@biglever.com



■ **ABSTRACT**

Product Line Engineering (PLE) is a systems engineering discipline to engineer a product portfolio using variation management techniques to take advantage of the products' similarities while managing their differences. It has well known cost, quality, and time to delivery improvements compared to single-system development. Systems Security Engineering (SSE) is a discipline for engineering systems proactively and reactively mitigating vulnerabilities. Do these two engineering disciplines conflict with each other? Or are they compatible, even complementary? This article will discuss the relationship between PLE and SSE and how they work together. Specific topics include managing a product line factory of products in a secure way and implementing cyber resilient frameworks within a product line addressing cyber resiliency commonality and variability in product design, implementation, and migration over time.

## INTRODUCTION

Product line engineering (PLE) and Systems Security Engineering (SSE) are two engineering disciplines needing little or no motivating justification. Each provides an indispensable engineering solution to an overwhelming economic impetus. The prohibitive cost PLE avoids is building and maintaining a portfolio of similar systems independently or semi-independently. SSE avoids disastrous theft, protected information disclosure, or service denial resulting from a security breakdown.

Lacking insight, a systems engineer may choose one discipline over the other and accept what he or she perceives as the lesser of two evils in a particular engineering context. Happily, however, it is not necessary to choose between the renowned efficiencies of PLE and the reassuring safeguards of security. This paper will show one can apply both disciplines together and achieve the best of each.

First, what does it mean, operationally speaking, to apply PLE and SSE together? It means accounting for a bi-directional relationship:

1. Applying SSE to PLE: Addressing security concerns while applying PLE processes on a daily basis.
2. Applying PLE to SSE: Building a product line of security-intensive systems to effectively and efficiently manage their commonality and variations.

This article will treat each of these as a section of its own.

Our goal is not to comprehensively define a new, hybrid engineering discipline but to provide examples and approaches showing how these two disciplines work together in practice. We want to show PLE and SSE are not antagonistic towards one another but help each other in order to achieve the cost avoidance in the purview of each.

## APPLYING SSE TO PLE

### *Securing the PLE Factory*

As discussed in the INCOSE primer on Feature-based PLE (INCOSE Product Line Engineering Working Group 2020), the concept of a "factory" is central. Shared assets—the engineering artifacts used to design, develop, implement, and deploy the products—are maintained as supersets with variation points and input to the factory. Also input to the factory are each product's feature-based descriptions. The PLE Factory applies one of these to the shared assets and the result is a set of asset *instances* applying specifically to the described product. Engineers in a Fea-

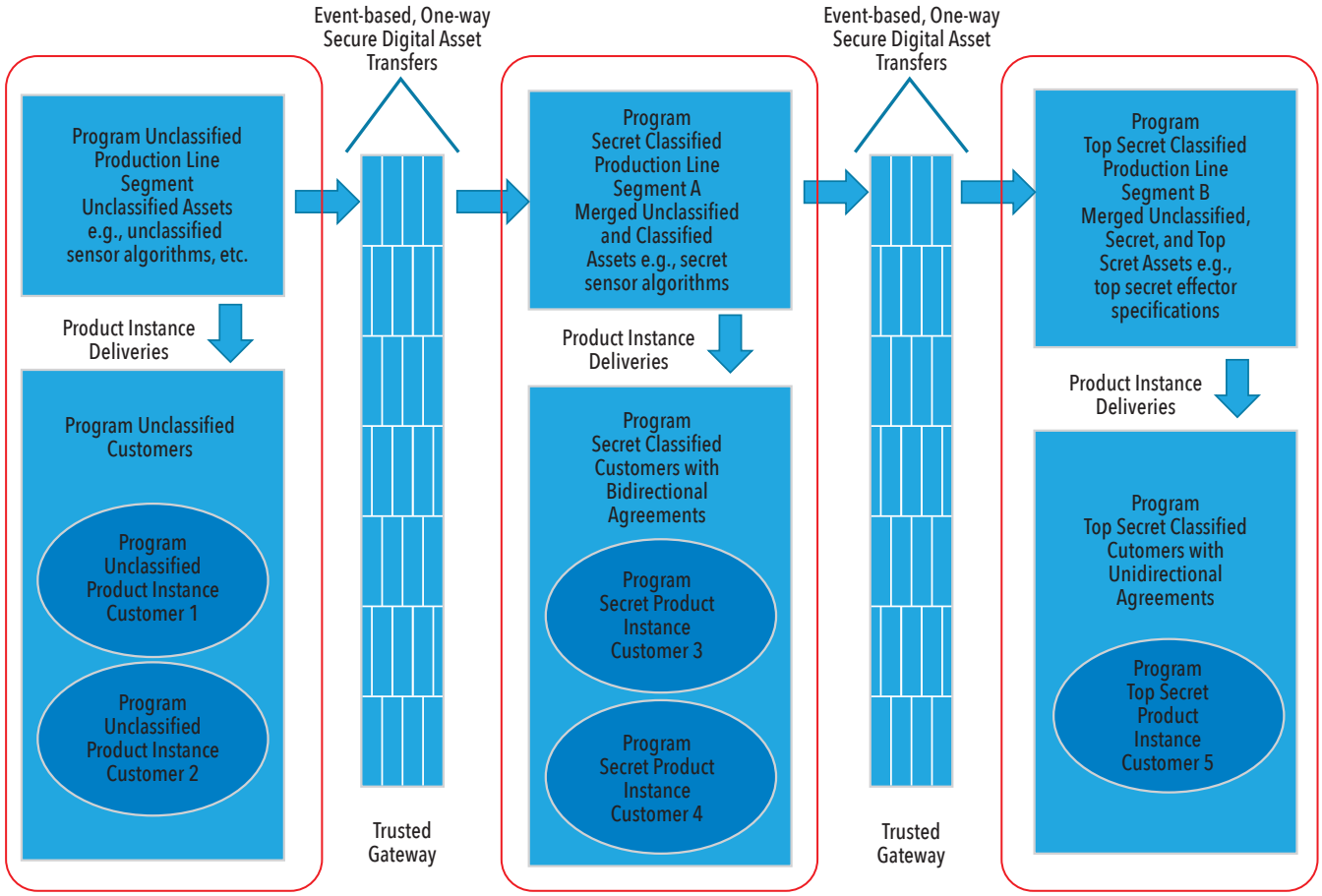



*Figure 1. Mixed Data Sharing and Multiple Security Level Daisy Chain Approach (Teaff 2019)*

ture-based PLE organization focus on the PLE Factory's care and evolution—managing and implementing evolution to the Feature Catalog, the Shared Asset Supersets, and the Bills-of-Features. The PLE Factory generates the products themselves; no development occurs in them.

In this section, we consider the PLE Factory as the entity needing securing. Under this view, Feature-based PLE is no different from any other project you want to secure; here, the PLE Factory care, feeding, building, nourishing, and evolution is the "project." The objective is making the information contained in the PLE Factory—the features, the shared assets, and the product definitions—immune to theft or disclosure to the extent required.

As a specific example, we will consider how to make the PLE Factory "project" secure across multiple security classification zones. Imagine our PLE Factory includes requirements, source code, test cases, and various documentation types. Some capabilities of our product line, or certain aspects, are classified. This means some requirements are classified, as is the source code, test cases, and the documentation portions corresponding to those classified capabilities. Specifically, a portion of each Shared Asset Superset is classified. This is the information we must safeguard.

If all product line members have a classified capability, then there will be no feature in the Feature Catalog to choose; the capability is not a distinguishing characteristic if every product has it. Let us assume, however, at least one of our classified capabilities is optional: some product managers have chosen it, others have not. Then there will be a feature in the Feature Catalog to represent it. Such features, which are generally just descriptive names accompanied by a brief explanation, may or may not be classified. Typically in practice, these features are not classified; it is their specification and implementation that is. However, if the features are classified, the portions of the Feature Catalog corresponding to the classified capabilities simply join the safeguarded information.

Given we know the classified information, the PLE Factory and the digital information developed and maintained for its use must follow the same SSE rules as when handling any "project" information across multiple security zones. Each customer may dictate the security zones depending, for example, on International Traffic in Arms Regulations (ITAR) restrictions and/or security classifications. The classified and unclassified security classification levels define security zones. The separation of these security classification levels requires managing and storing information on different Information Systems (IS). In addition, PLE Factories may have to manage digital information for different customers who may not want their information shared or commingled with other customers' information even at the same classified levels. The result may require additional information partitioning either within the same IS or on different ISs. The need to compartmentalize multiple customers' intellectual property from each other is simply another type of "security classification" scheme.

Operating a PLE Factory across multiple security classification zones follows these rules:

1. Maintain all information associated with different classification levels in separate environments, on separate servers.
2. Perform as much work as possible in the lowest-level-classification (ideally, unclassified) environment.
3. Information may move into a higher classification level environment from a lower classification level environment,

but no information flows from higher to lower classification levels.
4. Information resident at a particular level includes the portions of the Feature Catalog, Shared Asset Supersets, and Bills-of-Features classified at that level. From time to time, through natural change processes, contents at that level evolve over time.
5. Define synchronization points at convenient points in time—periodically, at specific intervals, or in support to some event such as an upcoming build-and-test milestone. Within every level, a snapshot of the level's contents is passed to the next-higher level enclave, which then performs a merge between that content and its own.

As shown in (Teaff 2019), data-sharing agreements and trusted gateways can play a role in this process:

- Brokering unidirectional data sharing agreement with the agencies involved is required to take advantage of the reuse offered by product line engineering and continuously delivers product variants to various classified customers. Data sharing agreements between these agencies allows harvesting classified assets from pre-existing programs for use in the PLE Factory, and sharing and commingling variant customer information within the production line and the same security classification levels.
- In case there needs to be information separation between different ISs, the production line architecture uses a daisy chained assembly line approach as illustrated in Figure 1. Security zones manage each production line daisy chain segment. Files are automatically and securely transferred between IS security zones via a trusted gateway. The trusted gateway provides a rule-driven file content and meta-data inspection allowing for file transfer allowed by security policies.

Temporal baseline management synchronizes the digital information's evolution at each security level as development occurs over time. In the PLE Factory, production line segment level baselines define temporal baselines comprising the superset (not the product-specific instances output by the PLE Factory) of digital information and the PLE Factory files within the daisy chain segments. Standard configuration management processes and tooling can manage the baselines at each security level.

With this process in place, information that constitutes the contents of the PLE Factory is secured, in accordance with best SSE practices for safeguarding information

Now, what about the security of the products produced by the PLE Factory?

***Security of the instances produced by the PLE Factory***

To complete the applying SSE to PLE discussion, let us consider the product instances generated from the PLE Factory as the things needing securing. For our discussion this can mean either (or both) of two things:

1. Building, testing, delivering, deploying, and operating the instantiated product in a secure manner. This necessity is no different from a product build outside of PLE. Once the PLE Factory produces the instantiated product (in the highest-classification zone necessary), it delivers from that zone in the "usual" manner. This is SSE's normal realm.
2. The instantiated product must adhere to the security restrictions imposed upon it. It must comply with rules protecting intellectual property, ITAR restrictions, and regulatory restrictions of the country of sale.

The second is the case we will consider: Meeting content restrictions in a generated product.

A *capability audit* is a quality audit focusing on the restricted content's correct use. A product manifest assists in providing checks and balances at different points within the production, providing an auditable trail to any leak sources. The Feature-based PLE Factory approach provides the ability to identify sensitive capabilities and choose to either include or exclude them in a product instance. This occurs by:

- Features representing capabilities included or excluded in some products. Restricted capabilities identified as features.
- Feature and product profiles identify the capability features to include. Profiles for a product must not include capability features representing a forbidden capability.
- Variation points inserted within the asset supersets identify the location of information related to feature specific choices. Variation points can identify protected content.
- Assertions or rules exclude illegal combinations of capability features—a rule can state a particular feature is unavailable for a particular country of sale (ITAR), or for a particular customer (IP protection).
- The PLE Factory's automated configurator supports constructing product instances by exercising the profiles, variation points, and assertions.

The configurator must also provide a product manifest for inspection ensuring the configurator exercised the appropriate profiles, variation points, and assertions to include or exclude capability features as expected. In addition to inspecting the generated product manifest during the audit, steps must ensure careful construction when defining features, profiles, assertions, variation point placement, and inspecting/testing the generated product instance prior to delivery.

To read about an example of the capability audit approach in use, see Clements 2013.

## APPLYING PLE TO SECURITY

The previous section discussed applying SSE to PLE by addressing the PLE Factory's security and the security of the products emitted by the PLE Factory.

This section will complete our treatment of PLE with SSE by considering the reverse case: How do we apply PLE to the SSE realm?

Suppose an organization produces a product line of systems under security requirements such as requirements dealing with cyber-resilience (Ross 2018). Whereas Section 2 addresses handling information securely during the product's *development*—which is the PLE Factory's realm—this section deals with a product line of products required to *operate* securely.

To apply PLE we ask the following two questions:

1. How does the need for security manifest itself as we build, deploy, and operate each system in our product line? Specifically, what artifacts does the need to implement security affect? For any specific product line under any specific security requirements set, the answer will be a content list in a specific artifacts list, but for the sake of our argument let us hypothesize the following:
   - Requirements specifications will capture the security requirements
   - Other documents will define the threat environment
   - A Concept of Operations will define security-related scenarios and the system's desired response to each
   - System models will capture the secure solution's design (such as a security architecture)
   - Software source code will implement the secure solution
   - System test cases will evaluate the solution security
   - A Bill of Materials will catalog the devices necessary to provide physical security
2. For each artifact, is their security-related content the same for all product line members or does it vary?

Suppose the security-related content of each artifact we listed in Question #1 does not vary from system to system in the product line. For example, all products share the same security architecture crafted with the entire product line in mind, rather than just a single system. Or, all systems required operation under the same threat environment. This case most easily occurs under Feature-based PLE: security-related content in each artifact from Question #1 does not vary and needs no variation points.

However, the more interesting answer from a PLE perspective is they do vary from product to product. Managing variation like this is Feature-based PLE's superpower. Each artifact listed for Question #1 will exist as a superset including the respective content for each possible solution need. Feature selections will determine which system gets which security treatment, and variation points on all security-related content in each artifact will reflect the feature selection so the correct security provisions are in the requirements, documents, Concept of Operations, models, code, tests, and Bill of Materials.

In summary, Feature-based PLE treats security-related content as it would any other content varying from product line member to product line member, thus providing the appropriate security solution for each product line member. The PLE Factory will instantiate each product in the Requirements, Concept of Operations, models, and source code which all consistently define and implement a product meeting its security requirements.

Once instantiated, the product can return to testing and certification as needed, just like any other security-intensive system.

## CONCLUSION

Systems Security Engineering focuses on building and maintaining systems remaining secure despite malice or error. This requires using appropriate security measures and controls utilizing tools, processes, and methods needed to design and implement systems mitigating vulnerability to achieve system assurance (SEBoK 2019). Feature-based Product Line Engineering provides a company the power to take advantage of wide-spectrum reuse while managing variation among products delivered to multiple customers with variantsecurity concerns, vulnerabilities, and restrictions. Feature-based PLE can incorporate system security practices and architecture frameworks addressing customer protection needs and security concerns, including protecting intellectual property as data, information, methods, techniques, and technology used to create the system or incorporated into the system while maintaining physical and operational control of information classification levels.

Systems Security Engineering's nature is much broader than the context provided in this paper, as is Product Line Engineering's application to manage shared assets in a product portfolio, as is their interaction. A well-known security problem involves aggregating data (all unclassified) in order to infer something classified. Feature-based PLE helps ameliorate that problem up front, rather than after the fact, by putting all information in supersets. The gathered data resides next to each other, allowing much up-front identification and therefore prevention and mitigation. Once identified, Feature-based PLE allows written rules, enforced by the PLE Factory Configurator automation, to ensure the data never co-habit a product at a lower classification level than that of their aggregated information.

The focused approach to applying PLE as a complementary technology to SSE in this paper, presented the risk reduction opportunity in the portfolio and the concepts while ensuring the PLE assets security necessary in the product deployment. Embracing the SSE and PLE approaches facilitates deploying complex solutions to multiple customers with variant needs across multiple domains without significantly increasing the organization's burden. This ultimately leaves more resources available for advancing the core value ensuring the organization's success. ■

## REFERENCES


- Clements, P., C. Krueger, J. Shepherd, and A. Winkler. 2013. "A PLE-Base Auditing Method for Protecting Restricted Content in Derived Products." Paper presented at the 17th International Software Product Line Conference, Tokyo, JP, 26-30 August.
- INCOSE Product Line Engineering Working Group. "Feature-based Systems and Software Product Line Engineering: A Primer," Technical Product INCOSE-TP-2019-002-03-0404, available at https://connect.incose.org/Pages/Product-Details.aspx?ProductCode=PLE_Primer_2019, downloaded 17 August 2020).
- Ross, R., R. Graubart, D. Bodeau, and R. McQuaid. 2018. "Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems." Draft NIST Special Publication 800-160 2:1-158. https://insidecybersecurity.com/sites/insidecybersecurity.com/files/documents/2018/mar/cs03202018_NIST_Systems_Security.pdf.
- SEBoK contributors. 2020. "Guide to the Systems Engineering Body of Knowledge (SEBoK)." International Council on Systems Engineering (San Diego, US-CA). SEBoK Wiki (&oldid=59187).
- Teaff, J.K., B. Young, and P. Clements. 2019. "Applying Feature-Based Systems and Software Product Line Engineering in Unclassified and Classified Environments" Paper presented at the 29th Annual INCOSE International Symposiums, Orlando, US-FL, 20–25 July.


## ABOUT THE AUTHORS


**Dr. Bobbi Young** is a systems engineering fellow and certified architect at Raytheon Technologies. She currently leads an internal research and development project focusing on adoption of product line engineering across the business. She is regarded throughout Raytheon Technologies as an expert in MBSE and co-chairs an MBSE technical interchange group. She is a member of the INCOSE Product Line Engineering Working Group, Architecture Working Group, and MBSE Working Group. Bobbi is also an adjunct professor at Worcester Polytechnic Institute as an MBSE instructor and has co-authored a book on object oriented analysis and design. She is a US Navy Commander (ret).

**Rowland Darbin** has been with General Dynamics Mission Systems for 17 years and currently leads the Product Line Engineering (PLE) Center of Excellence facilitating the adoption of PLE factory principals across project teams. Prior to his current position, Rowland was the product line manager for the consolidated product-line management program coordinating systems engineering efforts across the US Army's Live Training Transformations (LT2) family of live training systems, balancing the needs training portfolio with the needs of the individual training ranges. Rowland is also the lead co-chair of the International Council on Systems Engineering PLE International Working Group as well as a reviewer for the International Organization for Standardization for the 26580 standard.

**Dr. Paul Clements** is the Vice President of Customer Success at BigLever Software, Inc., where he works to help organizations adopt Feature-based Systems and Software Product Line Engineering. Prior to this, he was a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute, where for 17 years he worked leading or co-leading projects in software product line engineering and software architecture documentation and analysis. Prior to the SEI, Paul was a computer scientist with the US Naval Research Laboratory in Washington, D. C. He is co-author of seven books and nearly one hundred papers on software architecture and product line engineering.

SEPTEMBER 2020
VOLUME 23/ ISSUE 3

# Engineering a Cyber Resilient Product Line

**Patrice Williams,** patrice.dillon.williams@raytheon.com; **Paula Moss**; **Susan Bataller**; and **Suzanne Hassell**


## OVERVIEW OF A PRODUCT LINE STRUCTURE SUPPORTING MULTIPLE ARCHITECTURES

A product line consists of a managed core set of composable systems with scalable features and customizable variations. Critical mission threads may differ across the product line, but key product line architecture components support the implementing capabilities supporting a specific customer mission.

The choice to adopt a product line engineering strategy allows an organization to manage its assets for efficient use across business opportunities. This article uses an illustrative product line containing two separate but related architectural solutions, which include some similar and some unique hardware assets. Developing shared hardware assets conforming to both architectural constraints facilitates asset usage across the entire product line. Using the product line engineering factory configurator adapts the shared software asset supersets to these hardware assets. This approach comes from Meyer and Lehnerd 1997.

Figure 1 illustrates an example dual architecture product line for cryptographic solutions, supporting a core and an adjacent market. Each architecture solution supports the requirements and

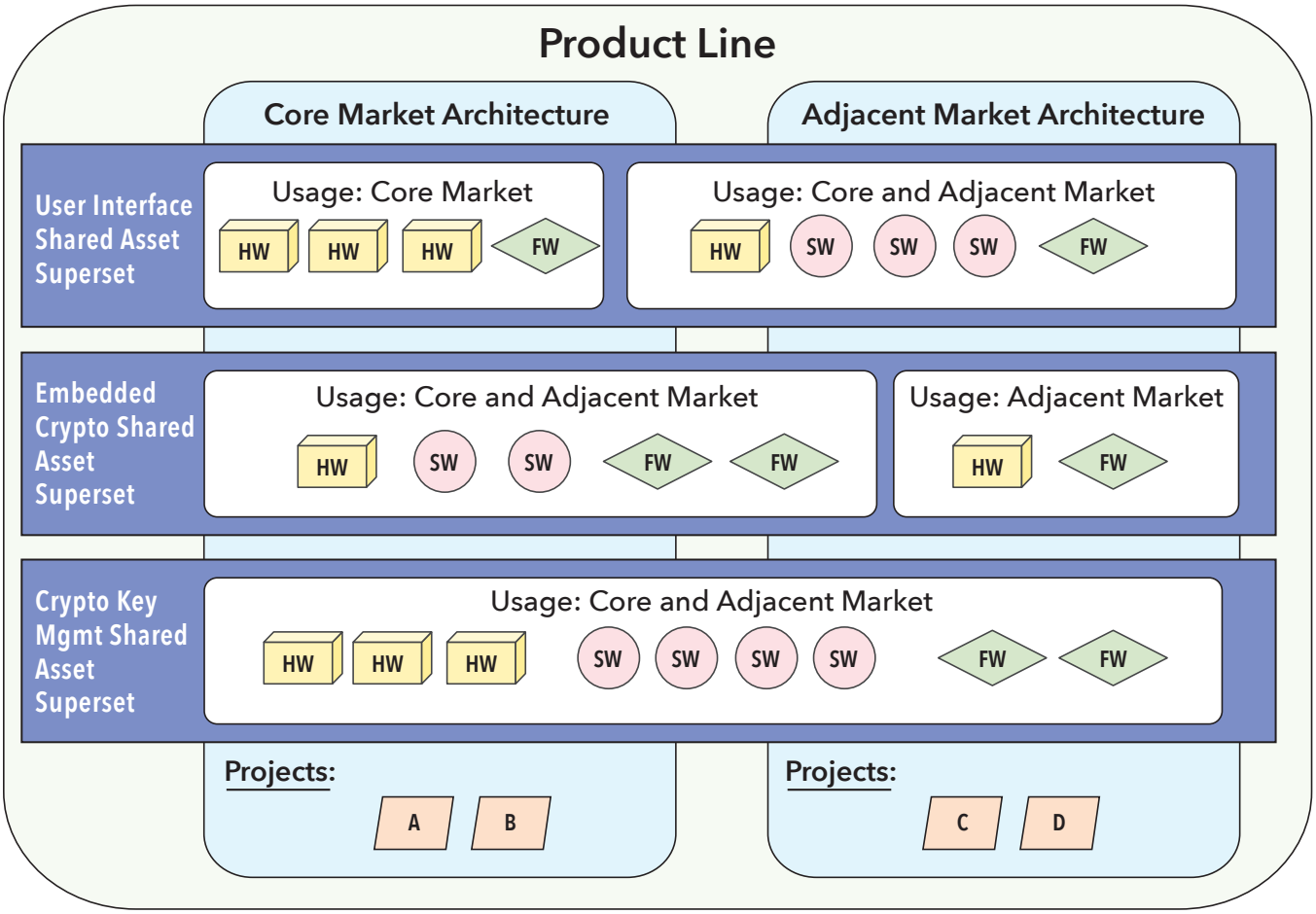



*Figure 1. Example product line structure*

environmental considerations of the business opportunities within that market. The product line has organized its shared asset supersets into separate functional areas to create embedded cryptographic modules, crypto key management systems, and user interfaces supporting both architectures. Within each shared asset superset is a set of hardware, software, and firmware assets supporting the core's architecture and/or adjacent business opportunities. Finally, each architecture supports several projects, with each being a unique architecture instantiation.

A product line can provide a structured approach to effectively managing commonality and diversity in its product offerings. An organization may decide to stand up a product line based on its business forecasts in its core markets. Alternatively, once established in a core market, an organization may wish to leverage its existing products to establish a presence in identified adjacent markets. The potential cost savings associated with effectively leveraging existing projects in the new market is a significant motivator for adopting a product line approach.

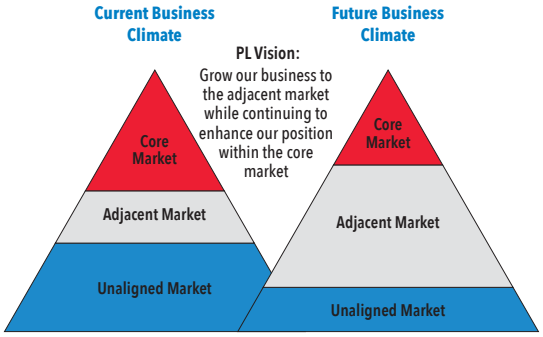



*Figure 2. Product line business climate*

Figure 2 shows the business climate for our example product line. The organization has an established business presence in its core market. The future business climate shows increasing opportunities in an adjacent market, with a corresponding decrease in core business opportunities. The organization anticipates reusing technologies and products developed in the core market will provide competitive advantages in the newly expanded adjacent market. Described below are the key components needed to establish a product line.

*Vision*: Create the vision for the product line, which identifies and binds its scope. This includes identifying the business opportunities within the current and adjacent markets the product line will support, and the business opportunities not in product line's scope.

*Product Line Guiding Practices*: Identify overarching practices used across the product line. These include identifying development and change control strategies, funding models, and strategies for shared asset development with defined variation mechanisms.

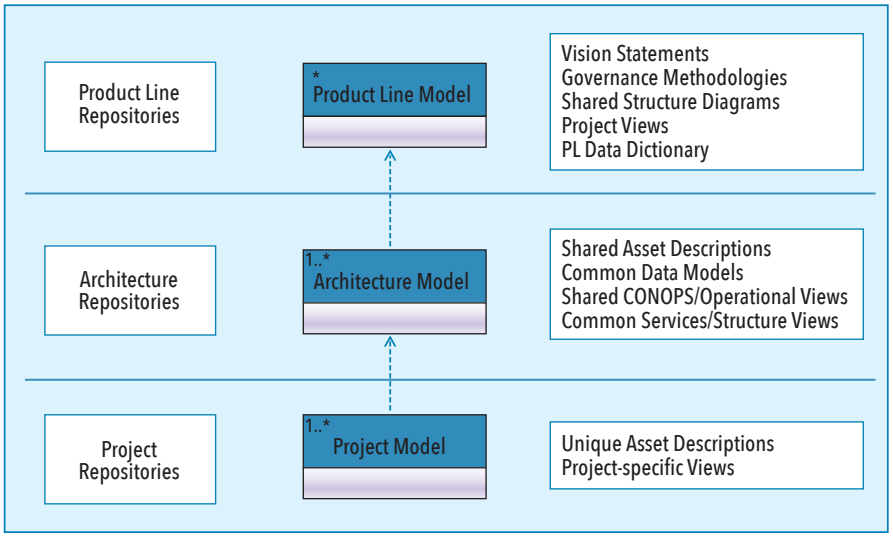



*Figure 3. Example product line artifact organization*

*Digital Environment*: Establish a digital environment supporting the product line, which organizes digital models and repositories to promote artifact reuse. Defining, characterizing, organizing, and managing the product line artifacts is essential for efficient governance. Creating customizable templates for common product line documents can reduce the effort for each supported project.

Figure 3 provides a layered artifact organization for the example product line. The product line repository stores artifacts defining the product line structure, established governance methodologies, project portfolio views, and product line evolutionary plans. The architecture repositories, one for each identified architecture, store information common to that architecture, such as architecture views, data models, operational views, and shared asset inventory. Finally, each project creates a project-specific repository to contain its unique views and inventory of project-unique assets.

The following recommended activities define each architecture within the product line.

*Characterize the Architecture:* Characterize the architecture by

*Table 1. Building block approach*

| QAs Supporting Building Block Approach | |
|---|---|
| **Modularity** | Composed of discrete components; minimized impact of change propagation |
| **Interoperability** | Ease by which one asset can exchange data with another; different operating environments |
| **Adaptability** | Ability to adjust to new conditions; the ability to adapt to new use or purpose |

*Table 2. Secure architectures*

| QAs Supporting Secure Architectures | |
|---|---|
| **Confidentiality** | Not disclosing information to unauthorized individuals or processes |
| **Integrity** | Not modifying or deleting protected data in an unauthorized and undetected manner |
| **Availability** | Ability to adjust to new conditions; modification ability for new use or purpose |
| **Resiliency** | The ability to adapt to changing conditions and to withstand and recover rapidly from disruptions. |

identifying the architectural principles and quality attributes which will define the architecture. Table 1 and Table 2 below show quality attribute examples (QA) .

*Create Common Architectural Views:* Common operational, data, and system views define the common architecture supporting the member projects.

*Identify Shared Architectural Constraints*: Shared architectural constraints facilitate building coherent systems from the shared asset superset. Examples include:

- Using a Modular Open System Approach (MOSA) and associated support for a specific Open System Architecture (OSA)
- Hardware architecture constraints, such as size, weight, and power restrictions
- Software architecture constraints, such as layered architecture, service-oriented architecture, specific middleware, and programming languages

*Identify Shared Security Attributes and Capabilities:* Security capabilities provided by the architecture may exceed the protection's scope required by any one project.

Creating a project as part of a defined architecture has advantages for both the project and the product line. The following depicts the typical steps undertaken when adding a new project to an architecture solution within the product line. The project benefits by inheriting a defined architecture, and a set of shared assets available to provide the required capability. The project identifies available shared assets, and the variation points used to create the architecturally compatible asset instance. Developing new assets within a shared asset superset can provide value to other projects in the product line.

Figure 4 shows an example project's product line use. Project 'A' belongs to the Core Market business, and therefore inherits its overall architecture. The project develops two functions, for stand-alone cryptography, and for key generation and distribution. For the stand-alone cryptography function, assets identified within the crypto and the user interface shared asset supersets are good fits for the new function. The project also identifies the need for new assets, developed within the shared asset superset. Similarly, the key generation and distribution function will use existing shared assets from the crypto key management shared asset superset. Since project 'A' has some unique requirements for this function, unique assets will provide this functionality.

A key concern in a crypto architecture is ensuring the resulting system can achieve mission success in a cyber-compromised environment. This is also known as 'Cyber Resiliency,' associated with the "Resiliency" product line quality attribute. Section II describes the approach for applying cyber resiliency analysis within the product line.

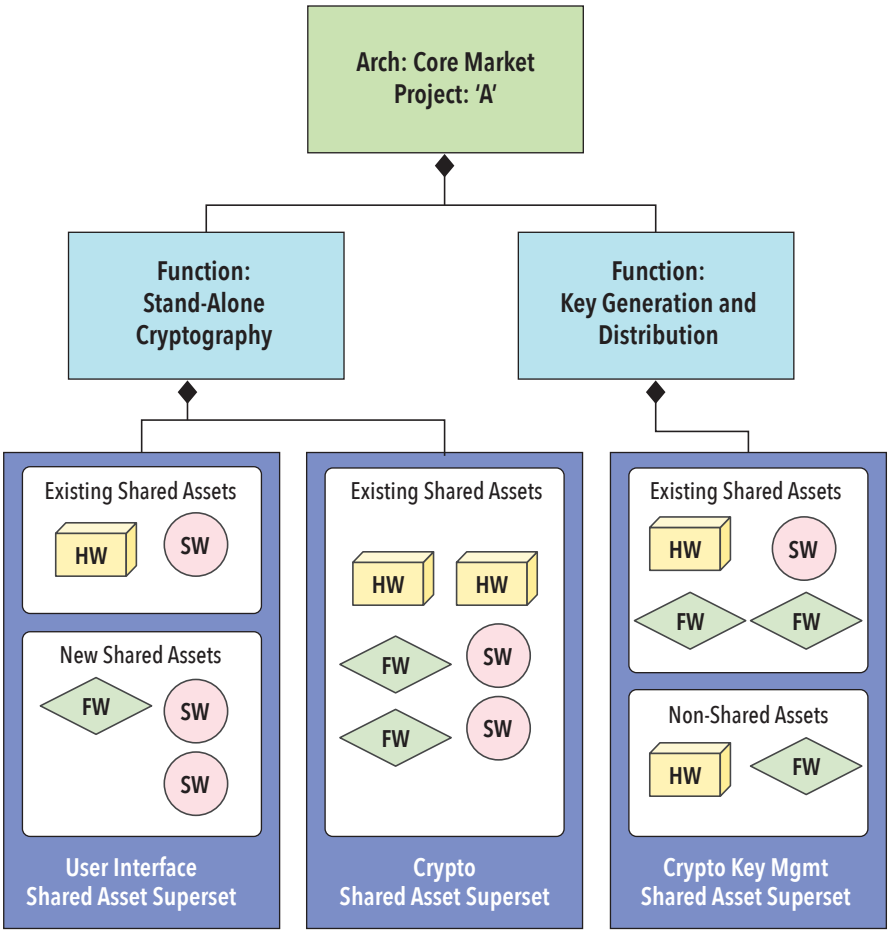



*Figure 4. Example project use of shared and unique assets*

## CYBER RESILIENCY AND PRODUCT LINES

Cyber Resiliency is achieving mission success in a cyber-compromised environment. It anticipates a compromised system. Cyber hardening of systems is insufficient to ensure systems continue operating in a cyber-compromised environment. Brittle systems may result in unreliable system performance and failed missions in an environment with ever changing threats. Reusing Commercial-Off-the-Shelf (COTS) and Government-Off-the-Shelf (GOTS) hardware, software, and firmware has created a vast attack surface including undiscovered or unpatched vulnerabilities. Vulnerabilities can also invade the system at any point in the system supply chain. "Resilient computer network defense must anticipate the emergence of new vulnerabilities, take action to exploit these vulnerabilities, and disrupt the actions of successful intruders to increase their work factor and minimize their impact. The focus of resilience is the assumption that attackers are inside the network, we cannot detect them, and yet engineers must ensure mission survival (Hassell 2015)."

"Cyber secure and cyber resilient approaches focus on both protection from and reaction to a cyber threat. Cyber secure approaches focus on keeping the adversary out of the system. Cyber resilient approaches focus on mission success if an adversary can get into the system. The cyber resiliency wheel applies these techniques to interim system architecture products demonstrating the architecture decisions made to improve cyber resiliency (Hassell, Wilson, and Williams 2020)."

Cyber Resiliency analysis addresses key concerns assumed to happen during system operation. You may not know the specific cause but anticipate the resulting effect on the Systems of Systems. Cyber Resiliency has a focus on key Mission Threads and their associated Key Performance Parameters, Technical Performance Measures and Measures of Effectiveness. The Mission Thread analysis is an architecture-based tabletop analysis of customer concerns based on known or anticipated attacker effects and capabilities engineers have applied or will develop and field offsetting those concerns. This tabletop analysis includes

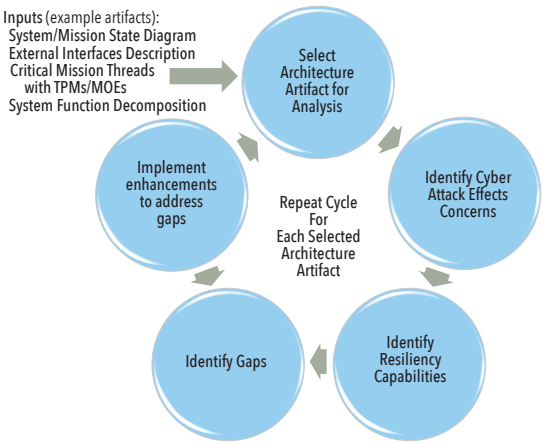



*Figure 5. Resiliency wheel*

a cross functional team including Operational subject matter experts, Software Engineers, Architects, and System Security Engineers. The Mission Thread binds the analysis timeframe to the time period during specific Mission Thread execution by the Systems of Systems. When identifying gaps, resources strategically allocated to implement enhancements close the gaps and any not closed track as a program risk. Figure 5 shows the process for performing the resiliency analysis.

The cyber resiliency analysis is made tractable by focusing only on key mission threads. Improving key mission thread component resiliency increases other mission threads' resiliency if they exercise the functionality in the improved components. If the improved components are a part of a product family, the resiliency lift applies across the product family.

Resiliency Concerns describe cyber attack effects on the System of Systems resulting from a cyber exploit. Architecturally, Use Cases describe normal system behavior. Misuse Cases describe resiliency concerns. A Misuse Case example is what happens to the Systems of Systems when it is under a Denial of Service attack. Table 3 provides a Resiliency Concerns list. It does not include all concerns applying from Systems of Systems inception to retirement.

Resiliency Capabilities can be capabilities built into the Systems of Systems or training and processes established for the Systems of Systems offsetting Resiliency Concerns. Resiliency Capabilities are proactive. They adhere to sound architecture principles such as "separation of concerns" and understanding and maximizing Quality Attributes such as "Trust." Table 4 derives, with some modifications, capabilities identified by Harriet Goldman (2010) of MITRE.

## APPLYING THE RESILIENCY WHEEL: CRYPTOGRAPHY EXAMPLE

The following example uses the 5-step Resiliency Wheel to analyze the cryptography architecture.

*Step 1:* Using the Critical Mission Threads, Key Performance Parameters, and Measures of Effectiveness for the domain; determine the applicable architecture artifacts for the analysis. These include the Concept of Operations diagram, System Block Diagram, Activity Diagrams, Sequence Diagrams, and State Diagrams.

For our cryptography example, the critical mission thread is the Crypto Key Management Product Platform. This includes creating embedded cryptographic modules, crypto key management systems, and user interfaces.

The Customer Key Performance Parameters and Measures of Effectiveness for the Cryptographic System are:

- User Interface
- Key Distribution
- Key Management

The architecture information supports architecture tabletop discussions with the stakeholders. The stakeholder group includes the customer, architect, safety, and security engineers, and key system developers.

*Table 3. Resiliency concerns*

| Resiliency Concerns (Effects of Exploit) | | |
|---|---|---|
| **Data Exfiltration** | **False Representation** | **Physical Effects** |
| **Disrupt Connection** | **Force Code Execution** | **Social Engineering** |
| **False Information** | **Force Supervisor Protected State** | **Software Exfiltration** |

*Table 4. Resiliency capabilities*

| Resiliency Capabilities | | |
|---|---|---|
| **Adaptive** | **Diversity** | **Non Persistence** |
| **Containment** | **Forensics** | **Pre-emption** |
| **Cyber Modeling** | **Integrity** | **Prioritization** |
| **Deception** | **Least Privilege** | **Pro-active** |
| **Detection** | **Monitoring** | **Randomness/Unpredictability** |
| **Distributedness** | **Cyber Maneuver** | **Reconstitution** |
| | | **Redundancy** |

*Table 5. Cryptography system resiliency concerns*

| Resiliency Concerns (Effects of Exploit) | |
|---|---|
| **Disrupt Connection** | Inappropriate storage of keys—Keys easily recovered by an attacker |
| **Data Exfiltration** | Key Re Use—Allows the attacker to crack the key |
| **False Representation** | Insider threat—Employees have access to keys |

*Table 6. Cryptography system resiliency capabilities*

| Resiliency Capabilities | |
|---|---|
| **Adaptive** | Audit log of key management |
| **Containment** | Policy to prevent reuse of keys—lifecycle management |
| **Least Privilege** | Role-based access to keys |
| **Detection** | Plan to detect key misuse within software |
| **Diversity** | Key rotation |

*Step 2*: Identify Resiliency Concerns. Table 5 describes the resiliency concerns resulting from the tabletop discussion.

*Step 3*: Identify Resiliency Capabilities mitigating the Concerns. Table 5 describes the resiliency concerns resulting from the tabletop discussion.

*Step 4*: Identify Gaps:

False information: Attacker cracks and manipulates keys (inaccurate information, malicious content attached).

Disrupt Communications: Keys stored improperly.

False Representation: Observe Operations for future malicious intent.

*Step 5*: Implement enhancements to the system(s) to mitigate the Resiliency Concerns.

The Resiliency Wheel should repeat when there are significant design changes to the system or changes to the operating environment raise new threat vectors. Cyber resiliency awareness should be an integral part of program system engineering activities.

Increasing cyber resiliency has emerged as a significant concern for both commercial and defense systems. When related systems belong to a product family, the effectiveness of adding resiliency to product modules accrues across the product family, reducing cost, schedule, and, most importantly ensuring mission success.

## SUMMARY

Product line engineering provides a tremendous opportunity for organizations. Utilizing proven practices and technology allows an organization to focus on enhancements and features benefiting their customers. While many benefits to engineering a product line exist, adding Cyber Resilient practices need attention. These Resiliency measures help ensure mission success in a cyber-compromised environment. Applying the cyber resiliency wheel techniques and focusing on critical mission threads throughout the system, helps engineers evaluate the organization's most vital needs. Although, it is impossible to build a product line hardened against every cyber-attack, it is possible to build a product line with confidence using Cyber Resiliency techniques. ■

## REFERENCES


- Goldman, H. 2010. "Building Secure, Resilient Architectures for Cyber Mission Assurance." Paper presented at the 201 Secure and Resilient Cyber Architectures Conference, McLean, US-VA, 29 October.
- Hassell, S. 2015. "Using DoDAF and Metrics for Evaluation of the Resilience of Systems, Systems of Systems, and Networks Against Cyber Threats." *INSIGHT* 18 (1): 26-28.
- Hassell, S., B. Wilson, and P. Williams. 2020. "Cyber Secure and Resilient Techniques for Architecture." Paper presented at the INCOSE International Symposium, online virtual event, 20-22 July.
- Meyer, M. and A.P. Lehnerd. 1996. *The Power of Product Platforms: Building Value and Cost Leadership*. New York, US-NY: The Free Press.


## ABOUT THE AUTHORS


**Patrice Williams** is a system security engineer with a focus on cyber security, MBSE, DevSecOps and secure architecture. Patrice joined Raytheon as a software engineer in 2009. She is a Raytheon cyber security resiliency architecture framework subject matter expert and has contributed to several cyber red team activities. She is currently participating in the 2020 CODE Center Cyber Rotational Engineering Program. Patrice has a BA in computer science and a Masters in cyber security with a focus in digital forensics.

**Paula Moss** is an engineering fellow at Raytheon Technologies and is a Raytheon certified enterprise architect. She has extensive experience with software, systems engineering, and architecture for command and control systems and is a cyber resiliency subject matter expert.

**Susan Bataller** is a senior engineering fellow at Raytheon Technologies and is a Raytheon certified enterprise architect. She has work extensively on military satellite communications systems in the systems, architecture, and software domains. Susan has been principally involved in conceptualizing, developing, and maintaining a product line architecture for Raytheon's military SATCOM products.

**Suzanne Hassell** is a principal engineering fellow at Raytheon Technologies and is a Raytheon certified enterprise architect. She has been at Raytheon since 2005, and is the Raytheon cyber resiliency subject matter expert and Raytheon Technologies Raytheon Intelligence and Space Chief Cyber Architect. She was the principal investigator for the US Army CERDEC Morphing Network Assets to Restrict Adversarial Reconnaissance (MORPHINATOR) program and led Raytheon resiliency research projects. Prior to coming to Raytheon, Suzanne did systems engineering, architecture, and software research and development in the communications industry for 23 years. She has 12 US patents.

# Security Issue Detection and Mitigation Patterns for Product Line Resource Variation

**Rick Dove,** dove@parshift.com


■ **ABSTRACT**
Product Line Engineering (PLE) builds upon an Agile Architectural Pattern—with reusable resources, evolving resource variations, and a standardized interconnect and sustainment infrastructure. Commercial PLE systems attract alternative resource suppliers, such as automotive parts. Defense PLE systems typically result from acquirers encouraging Open System Architectures to enable alternative resource suppliers. Alternative resource suppliers are a major resource variation source in product line engineered systems. Product line engineered systems are systems of systems with potential for complex interactions and unintended emergent behaviors. This article focuses on PLE cyber-physical-social system products, the security issues resource variation can introduce, and security patterns for detecting and mitigating these security issues. Resource variations may cause security issues unintentionally, but intentional introduction is also possible by malicious alternative resource suppliers, supply chain interdiction, and insiders. This article assumes malicious intent in resource variation as its security issue base line, as patterns for effective detection and mitigation of malicious variation intent encompass unintentional occurrences.

■ **KEYWORDS:** social contract; techno-social contract

## CONTEXT

Product line engineering (PLE) can be fractal. Military radios produced as a product line can be assembled for specific features from an inventory of electronic circuit board components pooled for variations on general capabilities. One pool may have variations in sensor signal processing, another in transmission encryption. These radios may become part of an aircraft avionics system with an open systems architecture structured as a product line to accommodate different radio and other avionic devices. The avionics systems in turn may provide variations for use in an aircraft product family.

This article broadly addresses product components architected and designed as a product line part, and focuses on detection and mitigation of security issues introduced by component variation. Components have cyber, physical, and techno-social interactions with other components collectively configured as a product. Components adapt internally to facilitate variation in component features fit for different purposes. Variation may occur in cyber, physical, and techno-social features.

More specifically, this article focuses on PLE variation security from a techno-social pattern point of view. The techno-social viewpoint centers on the *social contract* concept among components. The social contract concept, introduced by the French philosopher Jean-Jacque Rousseau (Rousseau 1762), recognizes humans aggregate as communities for mutual preservation. A social contract is an implicit cultural agreement or contract among society members that "essentially binds the members into a community that exists for mutual preservation" (SparkNotes 2005).

We propose PLE variation security is more effective when there is a techno-social contract of mutual protection among product components, we discuss why this is, and we show ten patterns useful for social contract compliance.

## TECHNO-SOCIAL CONTRACT CONCEPT

A product assembled from PLE component variants, built with intent to serve a specific user need (or desire), is a component collection which works to satisfy a user's total need. Ultimately, a component's task is to perform its intended functionality. A security issue in any other component it interacts with may affect its ability to function securely, but a security issue in any component may affect the ability of other components to function securely.

In a sense, we have a component community participating collectively to deliver total product satisfaction for the user. If the radio continues working in a disabled car the user experiences considerable dissatisfaction. This degrades the radio's participa-

tive intent. We propose the radio (and other components) should have a community sense specifically in the security domain, complying with a collective social contract for mutual protection.

## WHY TECHNO-SOCIAL ASPECTS ARE USEFUL

Long considered truth, no unit or system testing, certification, and standards compliance can guarantee secure product operation. These practices are necessary, but insufficient. Accidental or insider-malicious security issues may occur undetected at engineering time, but damage manifests at operational time. Operational time is when unexpected emergent behaviors can occur.

An Original Equipment Manufacturer (OEM) initially assembles PLE products and, we suggest, provides a mutual protection social contract among OEM components. However, in deployed operation, 3rd party components may replace the OEM product components, or additional components may add to the product from 3rd party sources. Nevertheless, the OEM remains the producer of record, and bears the responsibility of product security failures. If a 3rd party component has a security issue not affecting other product components, the OEM is innocent, but if a 3rd party component acts as a gateway for security issues spreading to OEM components the OEM is at fault.

The 3rd party issue underscores the need for PLE product components to distrust other components. The OEM does not control the deployed product configuration.

A malicious OEM insider, the OEM's supply chain when procuring parts for product components from a malicious source, or the OEM ships a safe product component but interdiction and malicious alteration occurs in transit to the product integration point, or when installation or maintenance personnel maliciously introduce a component variation are various security issue introductions.

The malicious intent issue underscores the need for distributed real-time operational behavior assessment by OEM components.

There are many ways to introduce a vulnerable variation. Attempting to preclude such introductions before deploying a product is a form of perimeter defense. Implementing a product techno-social contract is a form of defense in depth.

For the reasons above, the operation must actively detect and mitigate variation vulnerability in a component-distributed manner sensitive to abnormal operational behavior. If the OEM product can detect and mitigate the uncontrollable 3rd party security issues and security issues introduced with malicious intent, then it encompasses unintended OEM engineering issues evading discovery.

## TECHNO-SOCIAL PATTERNS

Peyton Quinn will provide a conceptual example. Peyton has a conscience, or so it seems. A voice saying you did something probably causing others some problems, and you ought to confess. Peyton resides in a gated community with a mutual protection social contract. A bit like neighborhood watch, but a lot more. The gate did not stop an intruder, evidenced by a mess made where a neighbor's keys were kept. Payton's conscience gets the upper hand and notifies the community association as well as the neighbors. The association responds shortly thereafter with a community broadcast saying a few residents have noticed security problems and recommends all go on high alert. Peyton double locks the doors, increases surveillance by cutting back on editing some videos as planned, and calls the cleaners to fix the mess the intruder made. Peyton Quinn is a blazing fast hardware/software techno-social device—pay a ton for edits, quintuple what software would have cost.

A social focus has patterns to consider from mutual security practices in human and animal social groups. The social focus in this article is technology-technology relationships rather than relationships involving humans or animals. Our concern is with components of a cohesive techno-social product community. The following ten patterns come from a paper discussing security in the Future of Systems Engineering (Dove, Willett 2020).

### *Self-Protection*

When a techno-social contract is present there is an obligation for components to perform the contract, seemingly benefiting others but it is a contract for optimizing self-protection. Self protection is an encompassing macro-pattern including the nine following patterns and more.

### *Self Aware*

Techno-social capabilities rely on self awareness, as socialness is a relationship between self and others. How much self awareness does a component need? At least awareness of the functional exchanges establishing interactive relationships with other components warranting attentive interest. Maximally, perhaps, as follows.

### *Self Behavior Judgement*

This is like a conscience, an independent local agent evaluating behavior for expected norms and deviations constituting abnormality. This approach does not rely on other components' sustained integrity to make judgement, it distributes watchfulness diversely and widely and is independent of potentially aberrant functional mechanisms, regardless of cause. Such an agent might exist within the component or as a separate component-dedicated companion. See Horowitz 2015 for a functional example.

### *Self Behavior Mitigation*

A self judgement may have different confidence levels. Some may be sufficient for unilateral immediate action. An extreme example proposed for ad hoc networks includes the ability for a component (node) to commit suicide for the greater good. Another component type might call for a wipe and reload. A less confident judgement may call for consensus among peer components or appeal to a higher authority, perhaps a component functioning as community overwatch attentive to multiparticipant appeals, or a human.

### *Peer Behavior Judgement*

Peer-behavior monitoring and judgement occurs naturally and constantly in social animals. Each group member evaluates the others for adherence to social norms and threats to social coherence and security. Humans monitor others' behavior in more sophisticated and more complex ways than animals of lesser cognitive capability. A techno-social component interacts with other components through communication and observed behavior, can learn what to expect as normal, and vet for normalcy before, during, or after acting upon it. A two-part journal paper in Dove 2009a and 2009b reviews literature supporting concepts and methods for peer behavior monitoring among unmanned autonomous systems. "Trust but verify" might be a polite operable phrase but is fundamentally about the need for distrust.

### *Peer Behavior Mitigation*

Rogue elephants are the result of banishment for unacceptable behavior. Social insects restrain and even kill group members that overstep certain social bounds. One of the 911 planes had passengers who took preventive action against the attackers. Nodes in some ad hoc networks will take a vote on questionable communication behaviors experienced with specific nodes and take collective action to refuse further interaction with a node receiving bad vote results.

### *Peer Collaboration*

Vehicular communication systems are computer networks in which vehicles and roadside units are the communicating nodes, providing each other with

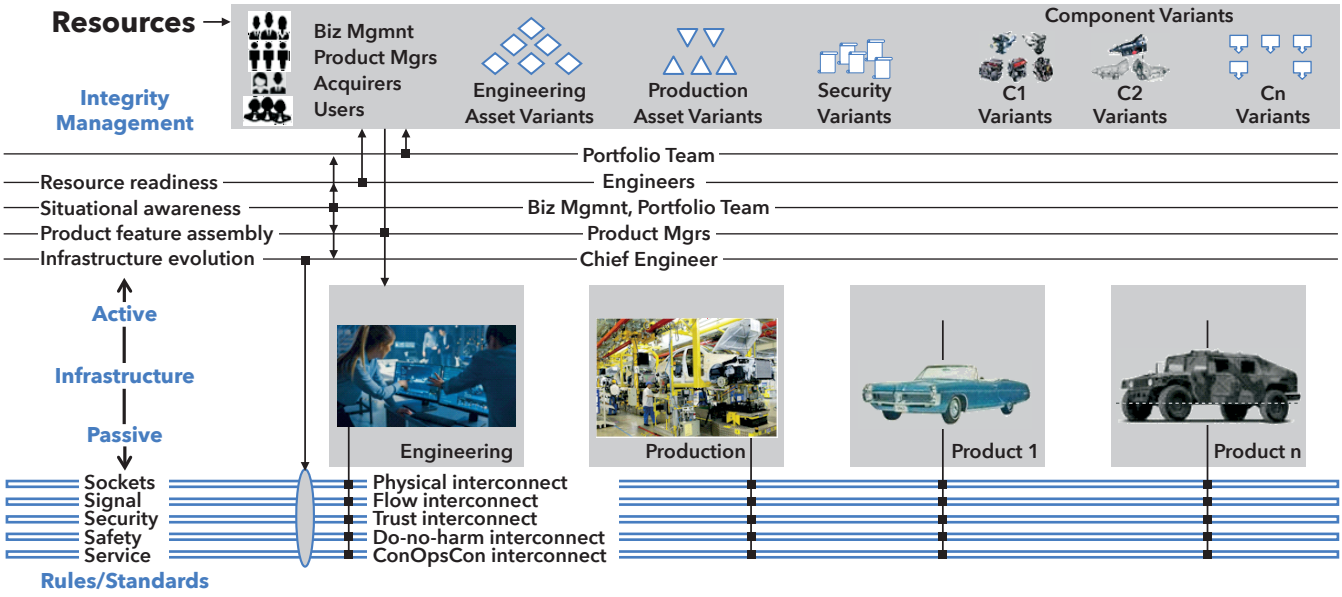



*Figure 1. Notional Concept: Security-Relevant PLE-Process Agile Architecture Pattern*

information, such as safety warnings and traffic information. They can effectively avoid accidents and traffic congestion. Both node types are dedicated, short-range communications devices. Vehicular communications usually develops as part of intelligent transportation systems (Wikipedia: "Vehicular communication systems").

### *Adaptable Attention Priorities*

*Maslow's [human] hierarchy of needs* (Wikipedia) contends fuel and security are the first two of six, sustained existence needs taking precedence over higher level purpose needs. This occurs in robotic mobile devices interrupting their tasks to seek an electrical outlet, and in devices and operating systems with various anti-tamper detection and prevention capabilities (short of self-destruction). For a notional technical hierarchy of needs see Dove and Willett (2020).

### *Diversity*

There is a socially attentive load on components attempting to cover a large awareness area, and inefficiency in duplicating their neighbors' same measures. All components do not have to participate, and all components should not look for the same things. One way this could implement might be to have a selection of (work intense) things to do randomly down selected by or for each component. Gal Kaminka makes this case in his doctoral thesis (Kaminka 2000) for distributed social behavior monitoring and detection, showing a centralized monitor does not do as well as multiple monitor/detectors among socially aware components. He also shows this can happen effectively without any one component monitoring all components, and without all the components having this monitoring capability.

### *Heterogeneous Awareness*

A recent study of grey squirrels (Lilly, Lucore, and Tarvin 2019) found they use signals from multiple bird species to indicate a present threat is in the area, as well as to indicate no imminent threat is present. Normal calm bird chatter finds the squirrels attending to foraging tasks, while alarm notes cause heightened agitation and evasive moves. Technical components receiving signals about the general state of alarm or calm in other components not in direct peer communication can ratchet the relative component attention level between self protective activity and functional purpose. Heterogeneity differs from diversity in that different social sub-groups have some cross communication, whereas diversity addresses a single social subgroup.

## ARCHITECTURAL VIEW

Engineered as systems, PLE components and component variations should apply good system security practices during their engineering activity, as with any system type. But a PLE product assembled from components and their variations provides an opportunity for security not readily available in a non-PLE product. This is true because of the PLE product architecture and the PLE process architecture. Both are classic agile architecture pattern forms (Dove and Schindel 2019), structured to facilitate reusable, reconfigurable, and scalable product and process configurations.

A PLE product has a standardized infrastructure facilitating interconnection among components and their variations to configure a product. A PLE process has a similar standardized infrastructure facilitating interconnection among engineering assets, production assets, and component assets, as depicted in Figure 1.

Techno-social security assets are a principle resource pool in the PLE process architecture. It is a pool of social contract variations because product intended for use by different customer types may need different capabilities. Figure 1 depicts a military vehicle and a drive-around-town vehicle as two possible product types in a product family. Military acquirers and users will likely want more security capability and features. The drive-around-town vehicle may have variations appropriate for evolving driverless operation or vehicle communication systems.

A techno-social contract for a PLE product family has three aspects for consideration: security assets associated with specific products, security assets associated with specific components, and system engineering assets associated with the PLE process. A product family employing a techno-social contract concept will likely have variants in all three asset classes.

Product security assets implement the techno-social contract at the product level of component-community interaction. Peyton Quinn's story referenced a community association providing security-related services to all community members. A product security asset may also phone home to the OEM with security information indicating issues needing attention in similar products.

Component security assets implement the techno-social contract at the component level. Some may also have direct communication capability with the OEM—to receive security updates and wipe-and-reloads, and to transmit component-specific security issues.

Systems engineering security assets enable assembling a techno-social contract for a product at two levels: resources include various techno-social contract assets which can draw upon various component and product security assets appropriate for a given techno-social contract. In human societies a social contract can be cultural, lawful, or both. A techno-social contract will generally rely upon a lawful approach governed by the contract's nature—short of artificial intelligence approaches beyond this article's scope. An OEM may decide contract governance includes contract enforcement, depending upon tolerance for 3rd party component inclusion.

## CONCLUDING REMARKS

We cannot guarantee security. There is nothing absolutely preventing the possibility a PLE variation introduces an exploitable security issue. Good and improved security practices in the PLE factory management processes will surely help; but it is insufficient to believe good security practice during PLE factory operation will ensure a secure product. This argues for security vigilance during operational product behavior.

PLE invites 3rd party suppliers. With a standardized component interface, the OEM cannot control replacing components in an operational product or adding components for additional capability. The automotive after market is a prime example. Defense acquisition's push to open systems architecture intended to enable componentry from other than the OEM. In any event, operational product augmentation or replacing OEM componentry can cause security issues. Measures countering security issues in the 3rd party operational environment can inform security practices in the OEM variant engineering activity.

A techno-social contract can provide emergent security behavior adapting to a varying threat. While instantiating techno-social contracts will not address all threat variations, it addresses more than the static safeguard predecessors. The result has the potential to act in a non-deterministic manner.

This article introduced notional concepts and patterns for a behavior-based techno-social contract among components in an operational PLE product. It also suggested three areas needing consideration for enabling, designing, and implementing a techno-social contract. Fundamentally the concepts and areas requiring work apply to agile security in general and warrants more attention for security in the Future of Systems Engineering. ■

## REFERENCES


- Dove, R. 2009a. "Paths for Peer Behavior Monitoring Among Unmanned Autonomous Systems." *ITEA Journal 2009* 30: 401–408. www.parshift.com/s/090901IteaJ-PathsForPeerBehaviorMonitoringAmongUAS.pdf.
- ——. 2009b. "Methods for Peer Behavior Monitoring Among Unmanned Autonomous Systems." *ITEA Journal 2009* 30: 504-512. www.parshift.com/s/091201IteaJ-MethodsForPeerBehaviorMonitoringAmongUas.pdf.
- Dove, R., and B. Schindel. 2019. "Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering." Paper presented at the 29th Annual International Symposium of INCOSE, Orlando, US-FL, 20-25 July. www.parshift.com/s/ASELCM-05Findings.pdf.
- Dove, R., and K.D. Willett. 2020. "Techno-Social Contracts for Security Orchestration in the Future of Systems Engineering." Proceedings 30th Annual International Symposium of INCOSE, virtual event, 18-23 July. www.parshift.com/s/200718IS20-FuSETechnoSocialContracts.pdf.
- Horowitz, B. (Principal Investigator). 2015. "Four part System Aware Cyber-Security Project report." Systems Engineering Research Center. Report No. SERC-2015-TR-036-4. https://apps.dtic.mil/dtic/tr/fulltext/u2/a626823.pdf.
- Kaminka, G.A. 2000. "Execution monitoring in multiagent environments." Ph.D. diss., University of Southern California (Los Angeles US-CA).
- Lilly, M.V., E.C. Lucore, and K.A. Tarvin. 2019. "Eavesdropping Grey Squirrels Infer Safety From Bird Chatter." PLOS ONE, 4 September. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221279.
- Rousseau, J-J. 1762. *On the Social Contract*. Translated by Maurice Cranston. New York, US-NY: Penguin Publishing Group.
- SparkNotes. 2005. "The Social Contract." www.sparknotes.com/philosophy/socialcontract/characters.


## ABOUT THE AUTHOR

**Rick Dove** is Paradigm Shift International's CEO, specializing in agile systems engineering and security research, engineering, and project management, and an adjunct professor at Stevens Institute of Technology teaching graduate courses in agile and self-organizing systems. He chairs the INCOSE working groups for Agile Systems and Systems Engineering, and for Systems Security Engineering. He is an INCOSE Fellow, and author of *Response Ability, the Language, Structure, and Culture of the Agile Enterprise*.

SEPTEMBER 2020
VOLUME 23/ ISSUE 3

# Effective Systems Security Requirements in Product Line Engineering

**Ademola Adejokun,** Ademola.Adejokun@lmco.com; and **Michael F. Siok,** Mike.Siok@uta.edu


■ **ABSTRACT**
Requirements engineering for complex software-intensive systems (and other systems) requires identifying, specifying, analyzing, and reviewing system requirements early in the system development process. However, many cases overlook system security requirements, treating them as an afterthought during this important initial process stage. Missing security requirements for these system types cannot guarantee system integrity. It is not cost efficient to retrofit requirements at later stages to include missing security capabilities specified earlier in-process. Detailed analysis and understanding of security requirements enable building confidentiality and integrity into our systems. Thus, early process activities must include security requirements engineering.

Product Line Engineering development must guarantee system integrity and assurance for a "family of systems" borne from a common design. Hence, detailed requirements elicitation and specification is important early in the product-line development and must include security requirements. Further, security requirements must revisit applicability, extension, and new security requirements specified to provide for security coverage of selected features contained within the product line's instances.

This paper describes an approach to security requirements engineering identification and includes introducing a security profile to facilitate developing and evolving a secure product line for software-intensive systems.

■ **KEYWORDS:** product line engineering, security, trustworthy system

## INTRODUCTION

Deficiencies in requirements engineering is a major risk amplifying and resulting in product defects over the entire product lifecycle. To develop a high quality and secure product, eliciting, identifying, analyzing, and managing system requirements is necessary (Insfran, Chastek, Donohoe, and Cesar 2014; Clements and Northrop 2001; and Kuloor and Eberlein 2002). Specifically, adequately specified security requirements. Concise and unambiguous requirements ensure a simple system design and implementation. This also reduces costly security-related defects found later in development and production stages due to requirements errors (Mellado, Fernandez-Medina, and Piattini 2010; Arciniegas, Duenas, Ruiz, Ceron, Bermejo, and Oltra 2006; and Baresi and Morasca 2007).

Product line engineering (PLE) includes a group or family of similar systems borne of a common design containing an allowed variations portfolio (features) generating instances of those systems for customers (Insfran et al. 2014). This underlying PLE nature enables continuous integration and a faster more cost-efficient product development and delivery by promoting secured systematic reuse of a large reusable Shared Asset Superset. The complexity and extensivity PLE nature requires adequate security requirements engineering for the common reusable shared assets superset.

The key enabling factor to advance PLE is generating a secure and trustworthy product instance drawn from a stable common, secure, and trustworthy base (the shared asset supersets) with allowed product features drawn from a features catalogue. Product instances generated from these trustworthy assets and features will inherit PLE's security capabilities and present an acceptable system security posture for the customer at delivery.

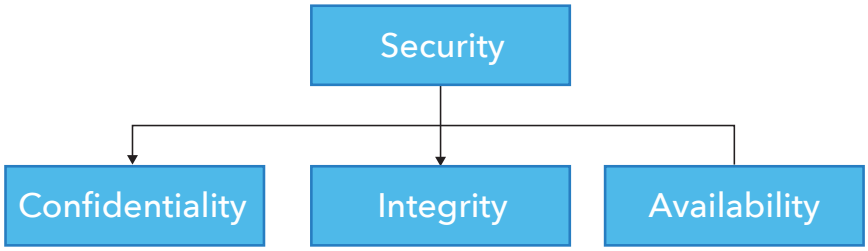



*Figure 1. Security tenets*

This paper explores a requirement engineering approach for PLE development emphasizing system trustworthiness of the common reusable product base (the shared asset supersets) and the generated product instances. The approach's key focus is reviewing, adopting, and tailoring relevant security standards for the PLE process lifecycle.

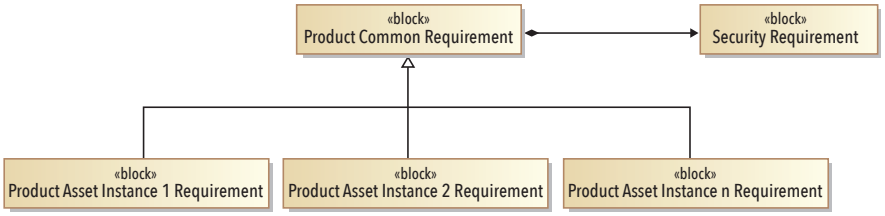



*Figure 2. Product variant inherits common requirements*

## SECURE PRODUCT LINE ENGINEERING

Confidentiality, integrity, and availability are the main security tenets to achieve system security and trustworthiness (Schramm and Grzemba 2011; Griglock and Kleidermacher 2001; Clements and Northrop 2001; Kuloor and Eberlein 2002; and NIST 2018). See Figure 1. Requirements supporting these security tenets establish a '*root of trust*' to facilitate a secure product line development and PLE product asset instnaces deployment.

An effective and systematic approach to define and manage product asset instance security requirements, product features, and feature portfolios ensures a secure product line with a consistent underlying and robust security policy and protection profile. This ensures secure critical data processing and continued PLE development process and generated product asset instance assurance.

The approach to achieve a secure product line is to promote product line commonality alignment with the main security tenets (Confidentiality, Integrity, and Availability) and to provide compliance with appropriate and relevant industry configuration management, development environment, and security requirements standards. Because PLE features are either infrastructure commonality parts or defined as part of the features portfolio, all product-line generated product asset instances can inherit common security requirements; specialized/ unique requirement(s) may be part of the relevant feature within the feature portfolio and that feature can manage it separately.

The following paragraphs provide this approach's key tenets descriptions.

***Confidentiality Requirements:***

Confidentiality requirements address protection against disclosing sensitive and critical data. The PLE should implement security controls to achieve confidentiality and ensure Data-at-Rest, Data-In-Transit, and Data-In-Use processing protection. Security engineering checklists should contain data protection verification features such as Encryption protocols.

***Integrity Requirements:***

Integrity requirements ensure PLE system reliability and data and processing accuracy. Verifying software functionality achieves reliability. Verifying data modifications occur in an authorized manner and data is complete and consistent before and after modification ensures accuracy. Specifying security control protocols, such as Hashing and Digital Signatures, ensures system integrity and are part of the Security Controls Checklist.

***Availability Requirements:***

Availability requirements ensure PLE system protection against service disruption. These requirements are part of product-line system infrastructure requirements and provide a means to conduct business impact analysis and define measurement of the system's maximum tolerable downtime and recovery time objective.

***Configuration Management Requirements:***

Configuration management requirements help define and direct the PLE infrastructure, application features, and system functionality development control and ensures the allowed product asset instance's integrity. Defining and enacting specific configuration items, including specifying effective management practices and measures, ensures PLE work product and process security and integrity.

***Deployment Environment Requirements:***

Deployment environment security requirements address the needs and methods to protect classified or sensitive data once the system transfers or deploys to the operational environment for systems integration and test and/or delivery.

## SECURITY STANDARDS APPROACH

Industry security standards and frameworks exist to provide a systematic approach to managing, securing, and processing sensitive and critical data. These standards provide specifications to achieve system resiliency, the system's ability to provide continued assurance and adaptable response before, during, and after a security event. Therefore, the approach here to achieve resiliency is to identify and tailor

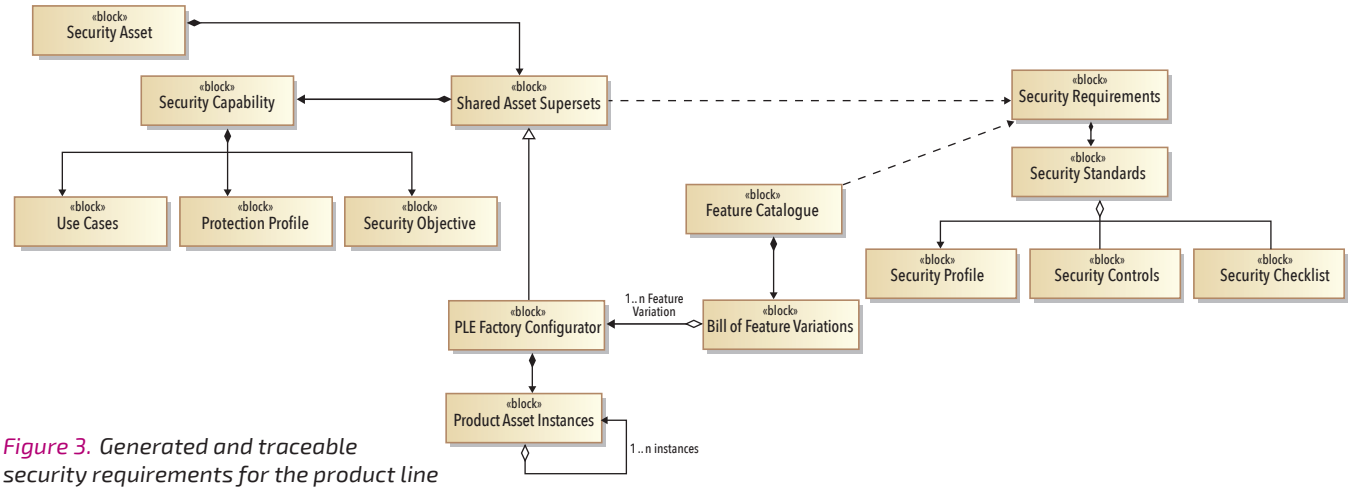



*Figure 3. Generated and traceable security requirements for the product line*

relevant and critical elements of the chosen security standards requirements and measures for the PLE.

Because of the PLE commonality and variability (Yu et al. 2008) specification, security requirements for critical system features of the shared asset supersets intend to align with a fundamental security profile. The product commonality top level, the PLE infrastructure, where generated product instances will inherit and reuse the common security features, must define and manage these requirements. Specific security requirements, needed and specified for the variable features, derive from and traced back to the overarching commonality or infrastructure specifications contained in the shared asset supersets. Figure 2 illustrates the these relationship's natures.The feature catalogue contains allowed PLE features that have been built with the appropriate security tenets; each feature in the catalogue complies with the security requirements of the PLE. The bill of features variations contain the definitions of each allowed variation of the PLE using the feature catalogue. Then, the PLE factory generator uses the reusable components from the shared assets repository and the bill of features variations (which draw components from the features catalogue) to construct a predefined allowed instance of the PLE. Figure 3 illustrates these relationships.

Standards such as the National Institute of Standards and Technology (NIST) Framework for Improving Critical Infrastructure Cybersecurity (Version 1.1), ISO/IEC 27001:2013 (the Information Security Management System), and ISO/IEC 15408:2005 (Common Criteria), identify and tailor relevant security requirements for the PLE Lifecycle application. For example:

- The NIST Framework for Improving Critical Infrastructure Cybersecurity is a risk-based approach to managing cybersecurity risk. The framework lists Functions and Categories/Controls describing specific cybersecurity activities common across critical infrastructure sectors. These categories can further align to the security tenets essential for the PLE requirement specifications. Table 1 illustrates this relationship (ISO/IEC 2013).

- The ISO/IEC 27001:2013 standard provides requirements for an information security management system (ISMS) necessary to keep identified information assets secure. The ISMS preserves the fundamental security tenets of confidentiality, integrity, and availability of the identified information assets by directing application of risk management techniques. These techniques provide confidence the information system management assures the security mechanisms. Table 2 illustrates this relationship (Arciniegas et al. 2006).

*Table 1. NIST framework for managing cybersecurity risk mapping to security tenets*

| Framework | Function | Category/Control | Security Tenet Alignment |
|---|---|---|---|
| 1 | IDENTIFY | Asset Management | Availability |
| | | Risk Management Strategy | Availability |
| | | Supply Chain Risk Management | Availability |
| 2 | PROTECT | Access Control | Confidentiality/ Integrity |
| | | Information Protection | Confidentiality/ Integrity |
| | | Protective Technology | Confidentiality/ Integrity |
| | | Data Security | Confidentiality/ Integrity |
| | | Maintenance | Availability |
| 3 | DETECT | Anomalies and Events | Availability |
| | | Anomalies and Events | Availability |
| | | Detection Processes | Availability |
| 4 | RESPOND | Response Planning | Availability |
| | | Communications | Availability |
| | | Analysis | Availability |
| | | Mitigation | Availability |
| | | Improvements | Availability |
| 5 | RECOVER | Recovery Planning | Availability |
| | | Improvements | Availability |
| | | Communications | Availability |

The Common Criteria process (https://www.commoncriteriaportal.org/pps/) focuses on the protection profiles for the security targets. The protection profiles specify security requirements facilitating the protections for the evaluated system. In this context, applicable Protection Profile standards tailor to the identified PLE security assets; the derived security requirements facilitate a secure PLE development again ensuring alignment with the security tenets. Table 3 specifies the Protection Profiles to extend the PLE security requirements.

## SUMMARY AND CONCLUSIONS

Security requirements are essential in the PLE development lifecycle. Lacking effective and adequate security requirements in early product line system development stages is a major risk manifesting in costly PLE infrastructure and generated system instances defects.

The infrastructure components of PLE provide the product-line commonality contained in the shared assets superset; the generated system instances is provided in the bill-of-features variations allowed variability that provides the delivered product instances with required and specialized security features. Thus, the PLE commonality and variability features promote a leveraged reuse approach enabling faster and more cost-effective product delivery over multiple system instance deliveries. Adequate security requirements specification and management across the product line is the key factor in developing and delivering

*Table 2. ISO/IES 27001 ISMS risk management mapping to security tenets*

| Category | Control | Security Tenet Alignment |
|---|---|---|
| Information Classification | Classification | Confidentiality/Integrity |
| | Labelling of information | Confidentiality/Integrity |
| | Handling of assets | Confidentiality/Integrity |
| Access control | Access control policy | Confidentiality/Integrity |
| | Access to networks and network services | Confidentiality/Integrity |
| Cryptography | Policy on using cryptographic controls | Confidentiality/Integrity |
| | Key management | Confidentiality/Integrity |
| Operations security | Change management | Integrity |
| | Capacity management Control | Integrity |
| | Separation of development, testing and operational environments | Integrity |
| Logging and monitoring | Event logging | Integrity |
| | Protection of log information | Integrity |
| Security in development and support processes | Secure development policy | Confidentiality/Integrity |
| | System change control procedures | Integrity |
| | Secure system engineering principles | Confidentiality/Integrity |
| | Secure development environment | Confidentiality/Integrity |
| | System security testing | Integrity |
| | System acceptance testing | Integrity |
| Information security continuity | Planning information security continuity | Integrity |
| | Implementing information security continuity | Integrity |
| Redundancies | Availability of information processing facilities | Availability |

*Table 3. Common criteria protection profiles mapped to security tenets*

| | Protection Profile | Security Tenet Alignment |
|---|---|---|
| 1 | **Access Control** | Confidentiality/Integrity |
| 2 | **Data Protection** | Confidentiality/Integrity |
| 3 | **Key Protection** | Confidentiality/Integrity |
| 4 | **Operating System** | Confidentiality/Integrity |
| 5 | **Product for Digital Signature** | Confidentiality/Integrity |
| 6 | **Trusted Computing** | Confidentiality/Integrity |

a product instance with specific security assurance properties efficiently.

Paradigms for specifying and managing security requirements across PLE lifecycles are a necessity. Systemically aligning and categorizing the PLE requirements with fundamental security tenets and compliance with the relevant industry security standards promotes a leveraged reuse development approach and infuses a working "plug and play concept" for generated system instances within the product line "family of systems." Bringing security into the open standard PLE lifecycle compliance will help advance the PLE lifecycles and the current state-of-the-practice for engineering future complex software-intensive systems. ■

## REFERENCES


- Arciniegas, J. L., J. C. Duenas, J. L. Ruiz, R. Ceron, J. Bermejo, and M. A. Oltra. 2006. "Architecture Reasoning for Supporting Product Line Evolution: An Example on Security." In *Software Product Lines: Research Issues in Engineering and Management*, edited by T. Kakola, and J.C. Duenas, 327-372. Berlin, DE: Springer-Verlag Berlin Heidelberg.
- Baresi, L., and S. Morasca. 2007. "Three Empirical Studies on Estimating the Design Effort of Web Applications." *ACM Transactions on Software Engineering and Methodology* 16 (4):15-55. DOI:https://doi.org/10.1145/1276933.1276936.
- Clements, P. and L. M, Northrop. 2001. *Software Product Lines: Practices and Patterns*. Boston, US-MA: Addison-Wesley Professional.
- Common Criteria Portal. 2020. "Common Criteria Protection Profiles." https://www.commoncriteriaportal.org/pps/.
- Griglock, M., and D. Kleidermacher. 2001. "Safety-Critical Operating Systems." Embedded. https://www.embedded.com/design/prototyping-and-development/4023830/Safety-Critical-Operating-Systems.
- Insfran, E., G. Chastek, P. Donohoe, and J.C.S.P Leite. 2014. "Requirements Engineering in Software Product Line Engineering." *Requirements Engineering* 19, 331-332. DOI 10.1007/s00766-013-0189-0.
- ISO (International Organization for Standardization). 2013. Framework for Improving Critical Infrastructure Cybersecurity. ISO/IEC 27001:2013(E). Information Security Management. Geneva, CH.
- Kuloor, C., and A. Eberlein. 2002. "Requirements Engineering for Software Product Lines." Paper presented at the 15th International Conference on Software and Systems Engineering and their Applications, Paris, FR, 3-5 December.
- Mellado, D., E. Fernandez-Medina, and M. Piattini. 2010. "Security Requirements Engineering Framework for Software Product Lines." *Information and Software Technology*, 52(10): 1094-1117. DOI:10.1016/j.infsof.2010.05.007.
- NIST. 2018. "Framework for Improving Critical Infrastructure Cybersecurity", version 1.1, 16 April. https://www.nist.gov/cyberframework/framework.
- Schramm, M. and A. Grzemba. 2011. "Trustworthy Building Blocks for a More Secure Embedded Computing Environment." Paper presented at the 2011 International Conference on Applied Electronics. Pilsen, CZ. 7-8 Sept.
- Yu, Y., A. Lapouchnian, S. Liaskos, J. Mylopoulos, and J. C. S. P. Leite. 2008. "From Goals to High-Variability Software Design." Foundations of Intelligent Systems, 17th International Symposium Proceedings. Springer Lecture Notes in Computer Science, 4994: 1–16.About the Authors


## ABOUT THE AUTHORS

**Ademola (Peter) Adejokun** has over 20 years' experience in systems and software engineering; he currently works as a cyber security systems engineer at Lockheed Martin Aeronautics in Fort Worth, Texas. Ademola is a licensed professional engineer in Texas, an INCOSE ESEP, Six Sigma Black Belt, a certified PMP, and holds the Security+ certification. Ademola is a senior member of the IEEE and ACM. He serves on the Object Management Group's UML Testing Profile and UAF subgroups; he also serves on the National Council of Examiners for Engineering and Surveyors (NCCEES) Software and Electrical/Computer Engineering PE Licensure Exam Committees. Ademola holds a BS in computer science & engineering and a BS in physics from the University of Texas at Arlington. In his spare time, Ademola is pursuing a Master's degree in security engineering from Syracuse University.

**Michael F. Siok** is currently an adjunct with the Computer Science and Engineering Department at the University of Texas in Arlington, Texas. Mike's previous experience includes 34 years as systems and software engineer at Lockheed Martin Aeronautics Company working in avionics systems, software engineering, and avionics test equipment systems development areas. Mike is an INCOSE ESEP, a senior member of the IEEE, and a registered professional engineer in the great state of Texas. He holds a Bachelor's of Engineering Technology in electronics from Southwest State University in Marshall, MN and MS and Doctorate degrees in engineering management from Southern Methodist University in Dallas, Texas.

SEPTEMBER 2020
VOLUME 23/ ISSUE 3

# Rule-based Verification of System Security using Feature-Based Product Line Engineering

**James K. Teaff,** James.K.Teaff@rtx.com


■ **ABSTRACT**
Systems security engineering is a discipline to engineer a system of interest ensuring system functionality under disruptive conditions associated with misuse and malicious behavior. Today's cyber-physical systems must survive in a constantly changing threat environment. Cybersecurity controls and cyber resiliency capabilities require continuous delivery to meet this challenge. This article describes creating and using a rule base as an integral part of a systems and software product line engineering (PLE) factory enabling continuous resilient and secure cyber-physical systems delivery ensuring the system can function under disruptive conditions. A hypothetical intelligence gathering network comprising unmanned vehicles, a ground control segment, and an intelligence analytics segment is the system of interest. Systems engineering activities include continuously analyzing the cyber-attack surface for each system variant in the product portfolio; creating or amending cyber resiliency capabilities and cybersecurity controls for each system variant addressing each attack vector; and a living rule base maintained within a PLE factory ensuring each system variant automatically generated by the factory contains the requisite system capabilities. Applying this approach results in rule-based verification of continuously delivered cyber resiliency capabilities and cybersecurity controls for each fielded system variant. Moving at the speed of mission need is essential, and automated rule-based verification of system deliveries meets that need.

## INTRODUCTION

"Software is eating the world," Marc Andreesen famously declared in his August 2011 Wall Street Journal article. This trend ultimately resulted in modern cyber-physical systems comprising people, hardware, and software fielded supporting diverse missions. These cyber-physical systems must survive in a constantly changing threat environment, which mandates continuous cyber resiliency capabilities and cybersecurity controls delivery. The software malleability lends itself to implementing the majority of the requisite system capabilities, and software, as noted by the United States Defense Innovation Board, is immortal—ever evolving long after its initial delivery to the field (McQuade et al. 2019) (McQuade, Medin, and Murray 2018). Recognizing speed and cycle time as the most important metrics for managing software, the Defense Innovation Board states system developers need to deploy and update software working for its users at the speed of relevance. Clearly, whatever can automate needs automation, removing human-in-the-loop processes in order to move at the speed of mission need; improve system quality, cyber resiliency, and cybersecurity; and enable users to trust the tools they use in the field.

To meet the continuously fielding secure systems challenge, organizations leverage feature-based systems and software product line engineering (FB-PLE). The key to FB-PLE is automation through a product line engineering factory. At the factory's center is a rule base used to verify each system variant created by the factory. A rule base is a specialized knowledge base or a knowledge repository obtained from subject matter experts and digitized as a rule set. A product rule is a system constraint represented in a rule base as an assertion. Each assertion is a Boolean expression—a true or false expression. Creating and using a rule base begins with analyzing the cyber-attack surface for each system variant in the product portfolio. Systems engineers then design invariant and variant system capabilities for each fielded system addressing the identified attack vectors. The factory digitizes and subsequently uses product rules to automatically ensure each output product configuration is secure, delivery after delivery to the field.

For the systems engineer, rule-based system security verification enables knowledge capturing from subject matter experts and then each field delivery's tireless, error-free verification by the factory using the rule base as the single

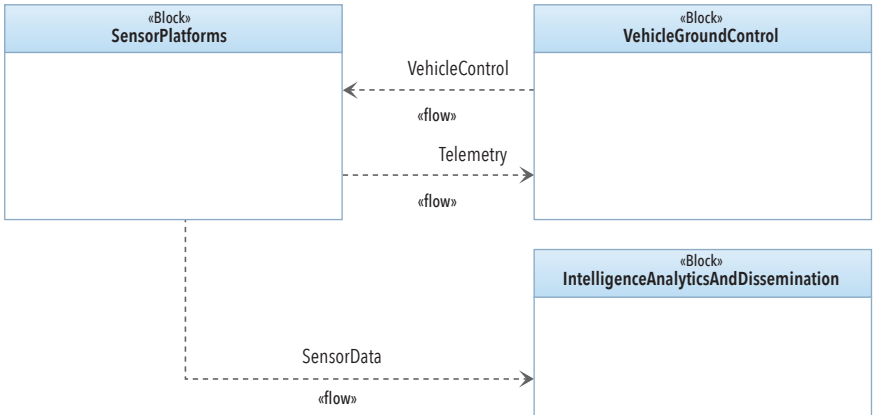



*Figure 1. Hypothetical intelligence gathering network (H-IGN)*

truth source. Contrast this with the typical systems engineer's writing requirements in a requirements management tool, developers interpreting the natural language requirements when creating a module, and finally testers independently interpreting the requirements when creating and executing a test. Each process step has the potential for introducing verification faults through inconsistencies inherent in multiple truth sources.

With the rule-based verification context established, the remainder of this article details the hands-on tasks a systems engineer performs creating and using a product rule base.

## HYPOTHETICAL INTELLIGENCE GATHERING NETWORK

This article uses a hypothetical intelligence gathering network (H-IGN), as depicted in Figure 1, as a rule-based system variant verification example.

The H-IGN system mission is to support the intelligence analyst via capturing and processing sensor data collected by sensor platforms. The H-IGN system comprises one or more unmanned vehicles (UxV), a ground control segment, and an intelligence analytics & dissemination segment. The H-IGN product portfolio supports three primary configurations:

- Small area of interest (AOI): line of sight UxV control and one analyst such as covering a city block
- Medium AOI: less than 200 square miles and up to five analysts such as covering a city
- Large AOI: less than 125,000 square miles and up to twenty analysts

Additionally, the H-IGN system supports the following variations in sensor platforms:

- One or more unmanned aerial vehicles
- One or more unmanned ground vehicles
- One or more maritime unmanned surface vehicles

The H-IGN system supports the following sensor platform control options:

- Continuous active ground control
- Autonomous vehicle control

The system capabilities and total lifecycle cost for each product variant in the portfolio differs based on its product specification: its factory bill-of-features.

## CYBER RESILIENCY CAPABILITIES AND CYBERSECURITY CONTROLS RULE BASE

The systems engineering activities for creating the factory product rule base include:

- Continuously analyzing the cyber-attack surface for each system variant in the product portfolio
- Creating or amending cyber resiliency capabilities and cybersecurity controls for each system variant addressing each identified attack vector
- Digitizing cyber resiliency and cybersecurity verification rules forming a living rule base
- Using the factory's living rule base ensuring each system variant automatically generated by the factory contains the requisite cyber resiliency capabilities and cybersecurity controls

The cyber-attack surface analysis for the H-IGN system identifies the ground control to UxV communications system (COMMS) as an attack vector. Potential attacks include:

- Passive data interception
- Covert data modification
- COMMS disruption

COMMS system feature's engineering ensures ongoing human and/or autonomous sensor platform control for each system variant, each differing in capabilities and cost. Features include:

- Wireless COMMS—radio frequency (RF); microwave radio
  - Encryption algorithms
  - Frequency hopping algorithms
- Line of sight COMMS
- Wired COMMS—essentially the UxV tethered to a ground operator

Modeling the sensor platform control modes results in a variant feature model, depicted in Figure 2, comprising the "Ground Control" feature and the "Autonomous" control feature.

Modeling the configurable COMMS capability set results in a variant feature model, depicted in Figure 3, comprising the "Wireless COMMS" feature and the "Line Of Sight COMMS" feature.

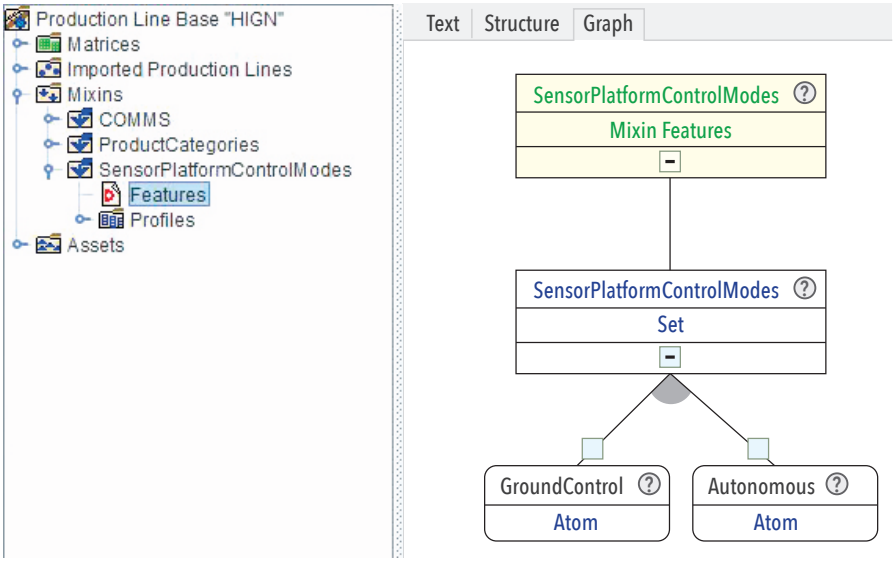



*Figure 2. H-IGN sensor platform control modes*

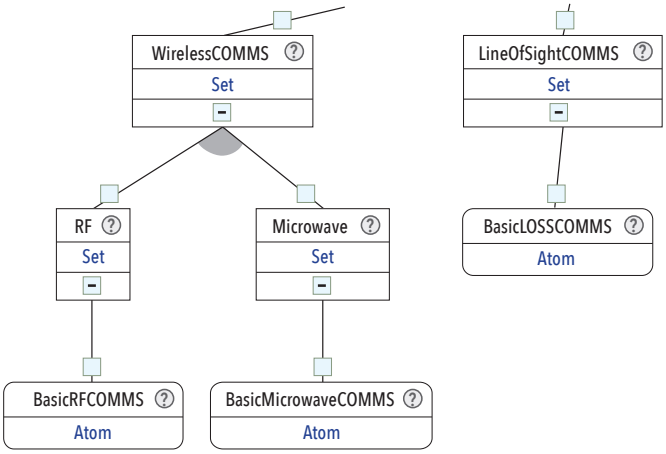



*Figure 3. Subset of COMMS capabilities*

The H-IGN subject matter experts provide the following specification:

- All product variants shall have redundant COMMS capabilities for greater cyber resiliency
- All product variants shall encrypt data to ensure cybersecurity
- RF COMMS shall use a frequency hopping algorithm to ensure cybersecurity

Additionally, for each AOI variant the subject matter experts provide the following specification:

- The small AOI variant:
  - Shall have both wireless radio frequency and line of sight COMMS for greater cyber resiliency
  - Customers may additionally choose to add "Basic" or "Enhanced" wired COMMS, but not both (this drives capability and cost variation)
  - Shall not support autonomous vehicle control (this drives capability and cost variation)
- The medium AOI variant:
  - Shall have both wireless radio frequency and microwave radio COMMS for greater cyber resiliency
  - Customers may additionally choose to add line of sight COMMS for greater cyber resiliency
  - Shall not support wired COMMS
  - Shall support autonomous vehicle control for greater cyber resiliency
- The large AOI variant:
  - Shall have both wireless radio frequency and microwave radio COMMS for greater cyber resiliency
  - Shall not support line of sight COMMS
  - Shall not support wired COMMS
  - Shall support autonomous vehicle control for greater cyber resiliency

Each subject matter expert specification digitizes into a product rule. The product rules exist in the factory rule base as assertions. Assertions use a Boolean expression language including Boolean operators and comparison operators as described in Appendix A—Product Rule Boolean Expression Language. The set operators frequently used are: ">" and ">=". ({TheSet} > A) is true when TheSet contains A and at least one other member. ({TheSet} >= A) is true when TheSet contains A.

For example, the subject matter expert specification "All product variants shall have redundant COMMS capabilities for greater resiliency" results in the product rule:

```
Assert ((COMMS.COMMSTypes > {WirelessCOMMS}) OR
((COMMS.COMMSTypes > {LineOfSightCOMMS}) OR
(COMMS.COMMSTypes > {WiredCOMMS})));
```

While the specifications "All product variants shall encrypt data", and "RF COMMS shall use a frequency hopping algorithm", results in the product rule set:

```
Assert ((COMMS.COMMSTypes.WirelessCOMMS.RF >=
{BasicRFCOMMS}) REQUIRES (COMMS.COMMSCybersecurity.EncryptionAlgorithm == {EncryptionAlgo_Alpha}));

Assert ((COMMS.COMMSTypes.WirelessCOMMS.RF >= {BasicRFCOMMS}) REQUIRES (COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm == {FreqHopAlgo_Bravo}));
```

Adding these assertions to the factory's product rule base results in the initial rule base for COMMS as presented in Figure 4.

```
Features of Mixin "COMMS"
    guideline:
    Feature Declarations
    Assertions
        Assert ((COMMS.COMMSTypes > {WirelessCOMMS}) OR ((COMMSTypes>
        {LineOfSightCOMMS}) OR (COMMS.COMMSTypes > {WiredCOMMS})))
        Assert ((COMMS.COMMSTypes.WirelessCOMMS.RF >= {BasicRFCOMMS})
        REQUIRES (COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm=
        = {FreqHopAlgo_Bravo}))
```

*Figure 4. Initial COMMS Rule Base*

## RULE-BASED SYSTEM VARIANT VERIFICATION

Once the digitized product rules are in place, the factory checks the assertions in the rule base and issues an error to the systems engineer for every rule violation as they work. For example, if a systems engineer attempts to design a system variant with basic RF COMMS without a frequency hopping algorithm, the factory throws an "assertion failed" when evaluating the product rule base:

```
Assertion failed: Assert ((COMMS.COMMSTypes.WirelessCOMMS.RF >= {BasicRFCOMMS}) REQUIRES (COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm == {FreqHopAlgo_Bravo}))
```

Figure 5 (next page) depicts an example product rules evaluation report. Once the systems engineer adds a frequency hopping algorithm to the system variant, and re-evaluates the product rule base, the assertion passes.

## RESPONDING TO THREAT ENVIRONMENT CHANGES

After fielding various hypothetical H-IGN systems, a compromise in the frequency hopping algorithm appears. A new algorithm installs, and the factory COMMS feature model updates adding frequency hopping algorithm "Charlie," while retaining frequency hopping algorithm "Bravo" for audits and traceability. The product rule base updates with an assertion deprecating the old algorithm using the "EXCLUDES" operator; and an assertion requiring the new algorithm added as follows:

```
Assert ((COMMS.COMMSTypes.WirelessCOMMS.RF >=
{BasicRFCOMMS}) EXCLUDES (COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm == {FreqHopAlgo_Bravo}));
```

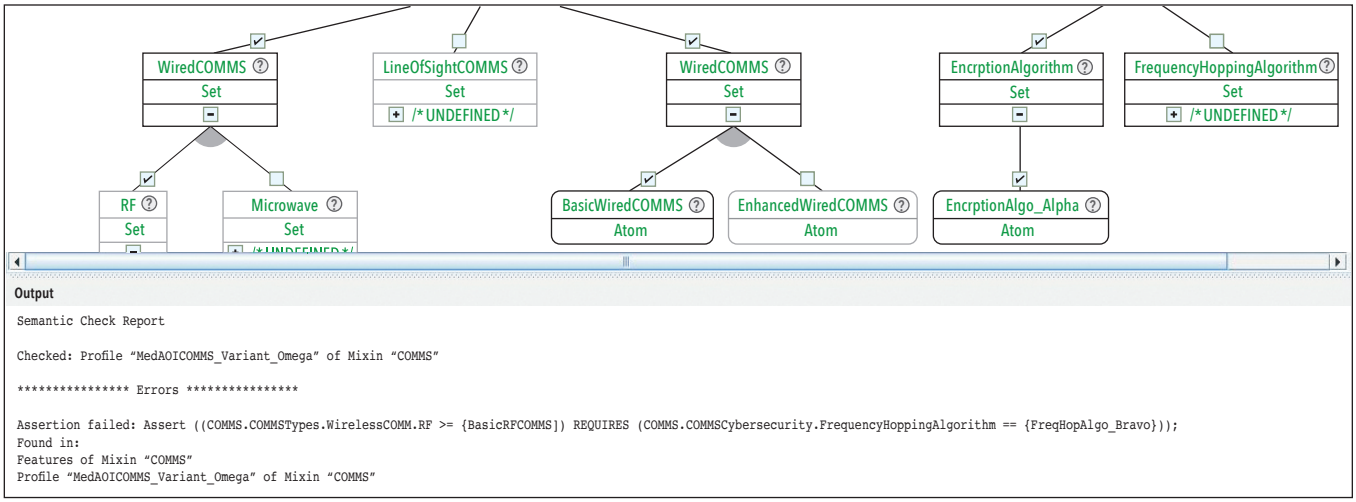



*Figure 5. Example product rules evaluation report*

Assert ((COMMS.COMMSTypes.WirelessCOMMS.RF >= {BasicRFCOMMS}) REQUIRES (COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm == {FreqHopAlgo_Charlie}));

The systems engineer performs an impact assessment to determine the product variant specifications requiring refactoring and redelivery, depicted in Figure 6. The systems engineer updates each impacted product variant. Subsequently the PLE factory automatically verifies the updated cyber resiliency capabilities and cybersecurity controls deliver to the fielded systems during each production run.

**Output**

```
Semantic Check Report

Checked: Profile "MedAOICOMMS_Variant_Omega" of Mixin "COMMS"

********** Errors **********

Assertion failed: Assert ((COMMS.COMMSTypes.WirelessCOMM.RF >= {BasicRFCOMMS}) EXCLUDES
(COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm == {FreqHopAlgo_Bravo}));
Found in:
Features of Mixin "COMMS"
Profile "MedAOICOMMS_Variant_Omega" of Mixin "COMMS"

Assertion failed: Assert ((COMMS.COMMSTypes.WirelessCOMM.RF >= {BasicRFCOMMS}) REQUIRES
(COMMS.COMMSCybersecurity.FrequencyHoppingAlgorithm == {FreqHopAlgo_Charlie}));
Found in:
Features of Mixin "COMMS"
Profile "MedAOICOMMS_Variant_Omega" of Mixin "COMMS"
```

*Figure 6. Example impact analysis report*

## SUMMARY

Moving at the speed of relevance is today's organizational mandate. Modern cyber-physical systems must survive in a constantly changing cybersecurity threat environment. This article described how to meet the continuous cybersecurity controls and cyber resiliency capabilities delivery challenge using a rule base embedded within a systems and software product line engineering factory to automatically create, verify, and deliver system variants. Systems engineering tasks included continuously analyzing the cyber-attack surface for each product line system variant, creating or amending cyber resiliency capabilities and cybersecurity controls for each system variant addressing each attack vector, and a living rule base maintained within a factory ensuring each system variant automatically generated by the factory contains the requisite system capabilities. By creating a single truth source and removing human-in-the-loop faults from verification processes, system developers improve system quality, cyber resiliency, and cybersecurity, enabling users to trust the tools they use in the field. ■

## APPENDIX A—PRODUCT RULE BOOLEAN EXPRESSION LANGUAGE

| ***Boolean operators*** | |
|---|---|
| And | Boolean AND Boolean |
| Or | Boolean OR Boolean |
| Not | NOT Boolean |
| Requires | Boolean REQUIRES Boolean |
| Excludes | Boolean EXCLUDES Boolean |
| ***Comparison operators*** | |
| Equals | Expression == Expression |
| Not equals | Expression != Expression |
| Less than | Expression < Expression |
| Greater than | Expression > Expression |
| Less than or equal | Expression <= Expression |
| Greater than or equal | Expression >= Expression |
| ***Comparison operators used for expressions involving Sets*** | |
| ({TheSet} == A) is true when TheSet contains A and only A | |
| ({TheSet} >= A) is true when TheSet contains A | |
| ({TheSet} > A) is true when TheSet contains A and at least one other member | |

> continued on page 38

# Leveraging a System Model to Initiate Security Architecture Development for Product Lines

**Angel Agrawal,** agrawal.angel95@gmail.com


■ **ABSTRACT**
This paper presents a method for leveraging Model Based Systems Engineering to facilitate systems security architecture development, with product line engineering considerations. The sensitivity of systems security information constrains the development of the system security model. Creating a tailored subset of the system of interest (SOI) model can generate value as a system security decision driver, and serve as the cornerstone for system security model development. This establishes clear correlation points between the SOI model and system security model. Conducting this collaboration early in the SOI lifecycle enables engineers to implement security considerations in the system design, reducing the cost and schedule impact of delaying system security development. For product line systems, common features are quickly identified with their associated security considerations. Variants retain these common features, preventing rework.

## INTRODUCTION

Model-Based Systems Engineering (MBSE) principles, when combined with a modeling tool, enable creating numerous system of interest (SOI) views. The Systems Modeling Language (SysML®) provides a standardized syntax and view set for defining the SOI. These views focus on the SOI from various stakeholders' perspectives. Systems Security Engineering's (SSE) primary concern, as defined by the United States Department of Defense, is "… to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities (SEBoK 2018)." The product's (or product line's) systems model, whether it be physical, logical, or functional in nature, can adapt to address security considerations. *Security Aware, Model Based Systems Engineering with SysML* by Herries et al. (2013) provides a fundamental groundwork for tools necessary to initiate modeling security architecture for a SOI. This paper will expand on their work, proposing additional views and efforts in SysML to leverage a systems model to support the security perspective. Additionally, this paper incorporates considerations for modeling product line engineering (PLE) as part of the security views. The following views leverage a SOI systems model, combined with PLE and security modeling techniques to create a fundamental SysML views set for capturing a security aware system architecture.

## FOREWORD ON MBSE FOR PRODUCT LINE ENGINEERING

Feature-based product line engineering has been described as a factory (Clements and Young 2017). This factory uses shared asset supersets, a feature model, and bills of features to deliver product instances. The factory analogy can extend to developing a SysML systems model for a product, derived from a model capturing the product line's asset superset. As a model captures the SOI design elements, the assets in a model are the various model elements within it. Variation points apply to these model elements as a tool for the configurator to understand what model elements to keep and which to remove when generating a model instance. Variation points coincide with the various features defined in the feature model, not with the anticipated model output. The complete feature model can then instantiate as a bill of features (analogously as a product specification). The configurator tool assembles an instance of the systems model, resembling a product satisfying the bill of features.

## FUNDAMENTAL SSE MODEL VIEWS

The following is a collection of views generated in a SOI systems model, with the primary objective of supporting security architecture development. These views bridge the gap between the systems model and security model by focusing on the system architecture aspects which will drive the security architecture's development. They are developed utilizing pre-existing model elements and relationships, focusing on the sensitive system elements. Creating these views requires access to information regarding these sensitive items and their usage. We assume the model (and consequently SOI) elements characterized as 'sensitive' are known. Modifying the feature model

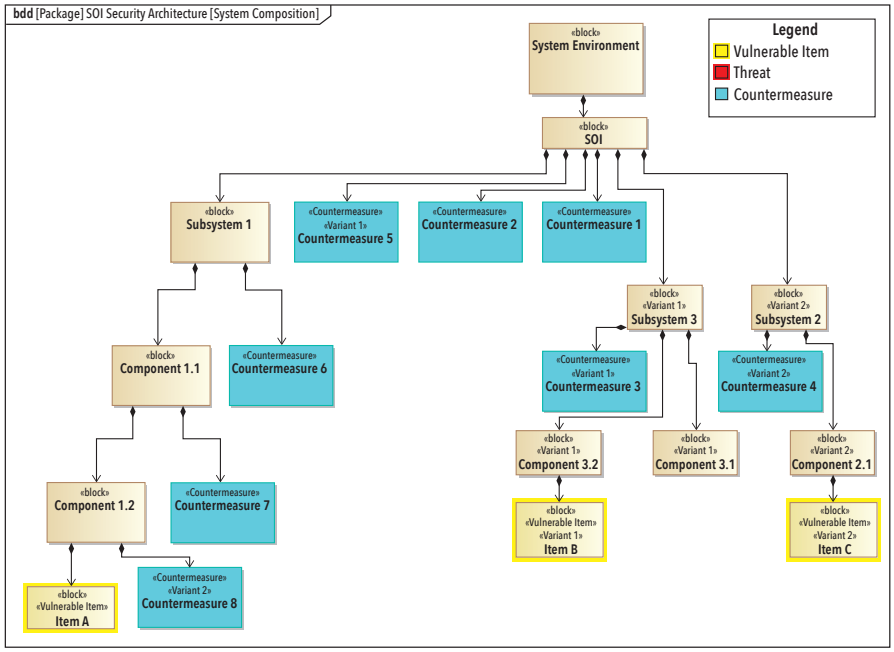



*Figure 1: System Composition*

for this product line can capture security related model elements which are intrinsic features. This may not be necessary for security related model elements which are a direct composition of pre-existing model elements already associated with a feature from the feature model.

### *Stereotypes*

Stereotypes can easily be used to identify security related model elements. Herries et al. suggests stereotypes for defining model elements from a security perspective (Herries, Oates, and Thom 2013). The profile constructed in their paper suggests using the stereotype "Vulnerable Item" to indicate the resources which may be the target of attacks. The stereotype "Threat" captures the effort(s) conducted by a malicious user (misuse cases). Elements or activities stereotyped by "Countermeasure" mitigate "Threats." The profile created in Herries, Oates, and Thom (2013) suggests a "Countermeasure" stereotyped block's function is to protect vulnerable items. In this paper, activities stereotyped as "Countermeasure" capture efforts to address different misuse cases directly. From the PLE perspective, this paper will use stereotypes to serve as variation points, indicating the variations corresponding to the different features. "Variant 1" and "Variant 2" are two stereotypes used in this paper. They denote the potential for including marked elements in the SOI's different configurations.

### *Level 0 (Program Level) State Diagram*

Every SOI state and configuration must take into account system security. System State Diagrams may describe the SOI states throughout its operation and usage (operational level). An abstraction level above the operational level is the lifecycle level. Lifecycle states captured here include development, testing, production, retirement, and other similar states surrounding the SOI's life from conception through retirement. The included vulnerable items, countermeasures, systems, interfaces, and activities may all be different between those states, so understanding and capturing this is important. Creating State Do, Entry, and Exit Activities occurs as necessary from the security perspective.

To a certain degree these states may be trivial. However, for the system security perspective, defining two things is essential: A state indicating a compromised system, and all the transitions between the other states and the "compromised" state. The compromised state captures the SOI state when an unintended or malicious user has accessed the SOI and now intends on gaining access to the vulnerable items. Defining a Do Activity here should serve as the groundwork for understanding the activities associated with exploiting and defending SOI vulnerabilities. From a PLE perspective, placing variation points on states and their transitions captures the differences across this program level features based view. Subsequent diagrams account for deeper abstraction levels tied to features.

### *Block Definition Diagram (BDD) for Item Containment*

Employing block definition diagrams can create views capturing subsystem and component architecture. From the security perspective, it can identify the relationship between vulnerable items, countermeasures, and the system's general components. Defining all vulnerable items and relationships to the SOI is essential to understanding their context within the system. From a functional modeling perspective, decomposing down to the abstraction level(s) using vulnerable items provides insight to their role in the system's behavior. From a physical and logical perspective, thorough decomposition provides insight as to the system elements which will comprise the vulnerable item(s) context. Figure 1 provides an example system composition for a SOI as a part of the SOI's environment, modeling the relationships as directed composition or aggregation. Personnel with need to know access may need to model the definition of the countermeasure model elements. However, modeling relationships to "Threat" activities will generate value as it will enforce countermeasure inclusion throughout the system architecture. Product line engineering considerations in this view should already exist. As mentioned previously, this example uses "Variant 1" and "Variant 2" stereotypes. Primary considerations for PLE when creating these views involve ensuring vulnerable items and countermeasures tied to components receive variation points consistent with the variation points applied to their owning element. For features which apply to vulnerable items and countermeasures separately than their owning components, variation points must apply separately.

### *BDD's for Item Exposure During Transit.*

Expanding the concept of identifying the model elements storing or using a program's defined vulnerable items, creating another Block Definition Diagram (BDD) can describe vulnerable items transitioning between different locations. Conventionally, using an Internal Block Diagram (IBD) would be used to show the connections amongst internal components to the SOI (a white box perspective). However, including the "Threats" and "Countermeasures" as activities as opposed to properties forces the use of a BDD. Fortunately, BDDs also permit creating ports and connectors similar to IBDs, and pre-existing IBDs can define the ports and connectors before pulling model elements into this particular view. Figure 2 shows this concept. Item flow on the connectors between ports represents the vulnerable items when exposed during transit from one domain to another. Directed allocation relationships can come from "Threat" activities to the various connectors representing an attempt to exploit a system

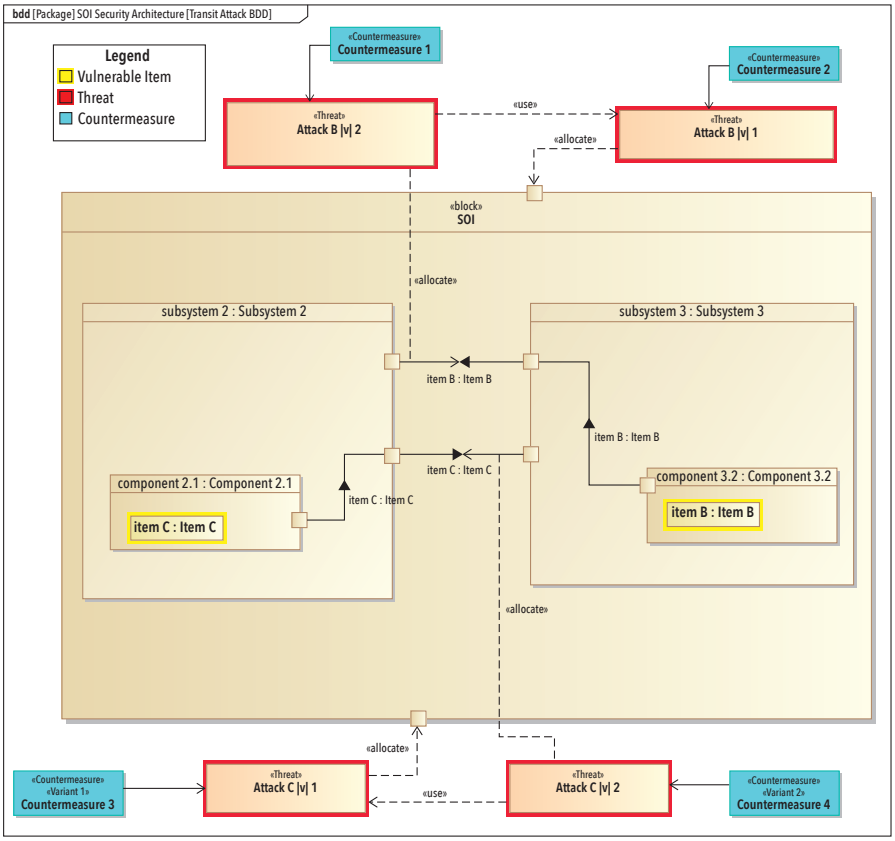



*Figure 2: Transit Attack BDD*

vulnerability. Relationships between "Threat" activities, modeled as directed use relationships, define a dependency on one activity's success for the other's application. "Countermeasure" activities integrate at this level and draw a directed association to "Threat" activities based on the 1 to 1 correspondence between a system's countermeasures and the misuse cases they mitigate. Placing the variation points on the various interfaces involved, easily accounts for feature variation as well as the blocks associated with them (if not already defined). The value of having this view is understanding the transit path as a potential exposure point for these vulnerable items. Capturing such information here facilitates developing adequate countermeasures to prevent exploitation during those processes. Furthermore, this view, as well as the following view, capture the fundamental relationships between "Threat" and "Countermeasure" activities, serving as the basis for the countermeasure coverage matrix and potentially systems security attack trees (not discussed in this paper).

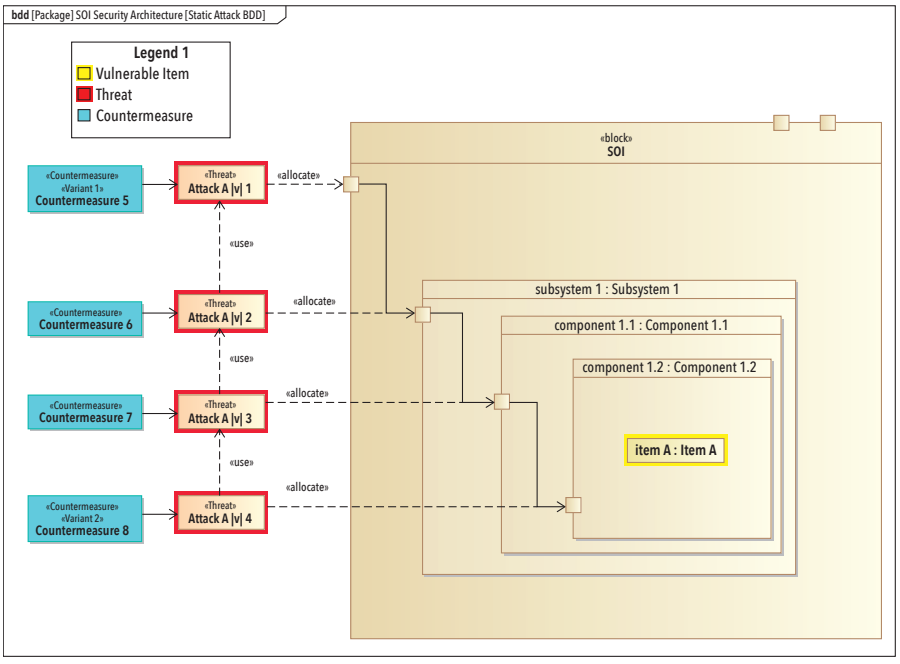



*Figure 3: Static Attack BDD*

### *BDD's for Static Attack Allocation*

In similar regard to using BBDs for identifying when sensitive items are vulnerable, a different misuse case allocation BDD can capture the SOI access points, SOI subsystems, and eventually the vulnerable information itself. Ports define and represent vulnerabilities in the system for which a malicious user may gain access to the vulnerable items. Much like the previously described view; allocations, usage relationships, and associations occur amongst the various model elements. Figure 3 provides a simple example of this, similar to Figure 2. The previously applied variation points on the blocks throughout the model account for product line considerations. Variation points can thus apply to the "Threat" and "Countermeasure" activities accordingly based on their relationship with the BDD view elements.

This view and the previous view's advantage is not requiring fully defined "Threat" or "Countermeasure" activities. Their existence and relationships are the important subject here. The scope of this work does not extend to a level infringing need to know restrictions, beyond identifying the vulnerable items. A modeling engineer's responsibility without need to know access ends at creating the "Threat" activities as undefined placeholders as well as creating the coverage matrix without creating any actual "Countermeasure" activities or defining them. Naming these "Threat" and "Countermeasure" activities can be simple enough to understand their role in the system life cycle. Fully defining these activities is a subsequent activity in the development of the security architecture.

### *Attack Coverage Matrix*

The quintessential value of creating directed association relationships between "Threat" activities and "countermeasure" activities is the ability to create a Countermeasure Coverage Matrix, shown in Figure 4. One axis represents the misuse cases captured by "Threat" activities. The other axis represents the "countermeasure" activities.

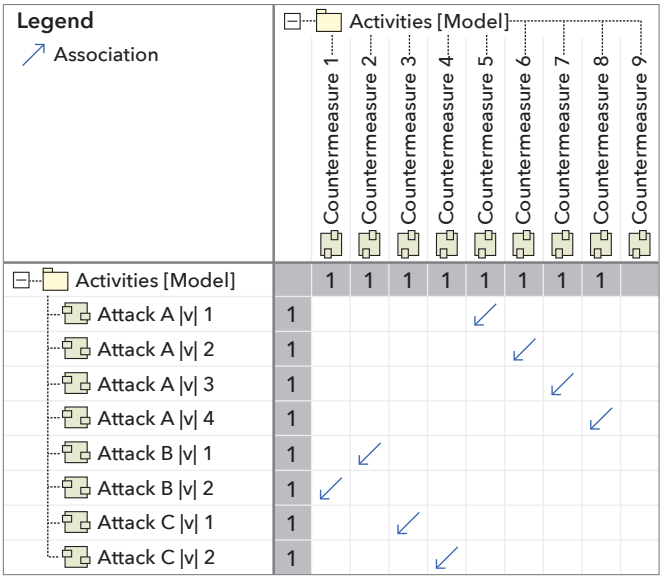



*Figure 4: Countermeasure Coverage Matrix*

The interior indicates the existence of the association relationship from the "countermeasure" to the "Threat." The applicable domain is the largest model containment span, accounting for all countermeasure and attack activities. Creating this matrix facilitates the assessment to determine if any anticipated misuse cases do not have an associated countermeasure prescribed and captured as part of the SOI. The same can occur for blocks and ports to understand what system components are the most vulnerable: what subsystems contain vulnerabilities which are left unguarded. When configuring the system model (as an asset superset) to meet a bill of features, and removing feature specific variants, any existing gaps in countermeasure coverage will be easy to recognize. It is imperative to generate this view for a countermeasure coverage assessment after the configuration step in the PLE process, to ensure coverage after removing non-associated superset assets from the final product. Then the countermeasure matrix will provide accurate information as to the correspondence between all misuse cases and countermeasures.

## CONCLUSION

This paper proposes a standardized diagram set, using the SysML modeling language and PLE modeling tools, for capturing a high level system security views. By creating security focused views, systems modelers can create a high-level security architecture while requiring little need to know access. Effective MBSE and PLE execution can ensure any resulting asset configuration has an adequate system security architecture defined for it. This method's potential value is twofold. First, it leverages MBSE techniques to facilitate collaboration and communication between SOI architects and SOI Security engineers. Secondly, it extends PLE capabilities to another systems engineering specialization. ■

## REFERENCES


- Clements, P., and B. Young. 2017. "Model Based Engineering and Product Line Engineering: Combining Two Powerful Approaches at Raytheon." Paper presented at the 27th Annual International Symposium of INCOSE, Adelaide, AU-SA, 15-20 Jul.
- Herries, H., R. Oates, and F. Thom. 2013. "Security-Aware, Model-Based Systems Engineering with SysML". Paper presented at the First International Symposium for ICS & SCADA Cyber Security Research, Leicester, UK, 16-17 September.
- SEBoK Editorial Board. 2018. *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 1.9.1, R.J. Cloutier (Editor in Chief). Hoboken, US-NJ: The Trustees of the Stevens Institute of Technology.


## ABOUT THE AUTHOR

**Angel Agrawal** is a systems engineer for Raytheon Technologies, and operates on the MBSE team for his program. He is a graduate from the University of Illinois at Urbana – Champaign where he earned his MS in aerospace engineering along with a certificate in aerospace systems engineering.

---

**Teaff** continued from page 36

## REFERENCES


- Clements, P., and L. Northrop. 2002. Software Product Lines: Practices and Patterns. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, US-PA.
- Krueger, C., and P. Clements. 2013. "Systems and Software Product Line Engineering." In Encyclopedia of Software Engineering, edited by Philip A. LaPlante. New York, US-NY: Taylor and Francis.
- McQuade, J., M. Medin, and R. Murray. 2018. "Defense Innovation Board Do's and Don'ts for Software." White paper, US Department of Defense Software Acquisition and Practices Study. https://innovation.defense.gov/software.
- McQuade, J., R. Murray, G. Louie, M. Medin, J. Pahlka, and T. Stephens. 2019. "Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage." White paper, US Department of Defense Software Acquisition and Practices Study. https://innovation.defense.gov/software.


## ABOUT THE AUTHOR

Jim Teaff wrote his first line of code as a professional in the early 1980's while working for a tech startup. Subsequently over the past several decades he has worked for numerous commercial and aerospace and defense companies across the full system development lifecycle–from principal investigator at a research lab, to a proposal writer, IPT lead, chief architect, SEIT lead, requirements analyst, scrum master, programmer, tester, 2nd tier O&M support, engineering manager… and more. Jim holds a Master of Engineering in engineering management degree from the University of Colorado; a Bachelor of Science degree in computer science from Colorado State University; and a graduate certificate in information systems security from Regis University. He is an INCOSE Certified Systems Engineering Professional (CSEP), and is an active member of the INCOSE Product Line Engineering International Working Group. Jim joined Raytheon Technologies in 2016 as a product lines champion and is assisting the organization with the continued rollout of product line engineering and management methods and tools.

# Towards a Model-Based approach to Systems and Cybersecurity

## Co-engineering in a Product Line context

**Juan Navas,** juan.navas@thalesgroup.com; **Jean-Luc Voirin**; **Stephane Paul**; and **Stephane Bonnet**


**ABSTRACT**
As cybersecurity threats multiply and global public opinion becomes aware of cybersecurity attack's potential consequences, customers become more demanding regarding cybersecurity concerns in the products they acquire. Consequently, product providers should consider such concerns early in their solution's development life cycle. This paper presents how a model-based approach can contribute to an effective co-engineering effort between cybersecurity and product engineering during product architecture definition.

### SYSTEMS AND CYBERSECURITY CO-ENGINEERING CHALLENGES

Society is crossing the threshold into the fourth industrial revolution where dependency on cyber-physical systems will dramatically increase. As these services' complexity will rise due to the new and unexpected system combinations, the cybersecurity vulnerabilities and potential cybersecurity attack targets will increase as well. Not surprisingly, the INCOSE *Systems Engineering Vision 2025* (Beihoff et al. 2014) included security, and particularly cybersecurity, as one of the eight key system characteristics desired by stakeholders. It encourages systems engineers to treat cybersecurity as a key system attribute they shall understand and incorporate in their designs.

The way a system shall be protected against cybersecurity threats is determined by the context on which the system operates, its interactions with the external actors, the components of the system and their properties and interactions. In a Product Line context, in which the system is a product portfolio part, analyzing the commonalities and variabilities between products and between the elements composing them is a key input for designing effective cybersecurity controls.

Hence, the development process' very beginning, and each subsequent development stage, should address and consider cyber security concerns. Such a *security-by-design* co-engineering approach not only diminishes the project's technical costs and schedule risks, but also permits trade-offs between cybersecurity concerns and other functional and non-functional system concerns.

Implementing this co-engineering approach in a Product Line context encounters the following barriers:

- Cybersecurity engineering requires specialized skills and has its own vocabulary, which usually varies following regulatory frameworks. Acquiring these skills requires a substantial investment, and human resources with both systems and cybersecurity engineering skills are difficult to find.
- Experience shows cybersecurity engineering activities have their own lifecycle potentially uncorrelated with the systems engineering activities and to the product roadmaps. This is mainly due to constraints from certification authorities and need-to-know constraints.
- The needs and constraints relative to the cybersecurity concerns, as well as the elements constituting the product architecture, may strongly differ according to the market, the product addresses, and the available technologies, among other factors. Furthermore, cybersecurity concerns' priorities may change in time. These factors make implementing reuse strategies difficult.

The authors present Model-Based Systems Engineering (MBSE) practices contributing to incorporating cybersecurity concerns into the systems engineering activities, and particularly in product's architectural definition, in Product Line Engineering contexts.

### FINDING A COMMON GROUND

To enable effective and efficient collabo-

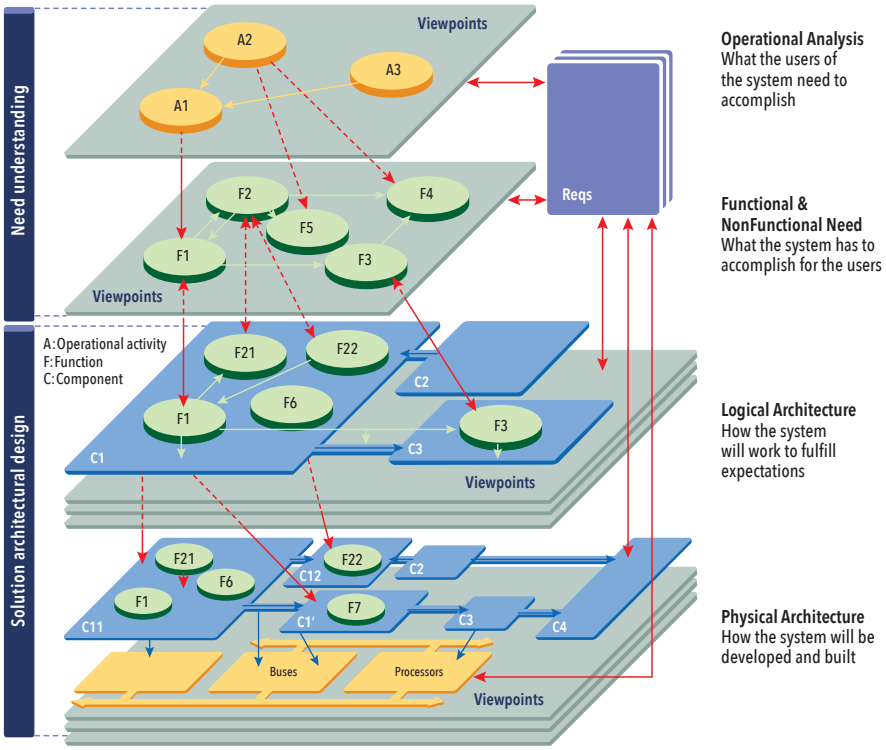



*Figure 1. Arcadia engineering perspectives*

ration between cybersecurity and systems engineering, both worlds need to agree on a common vocabulary. The concepts each one needs to understand from the others' will be a body of knowledge subset: this binds the co-engineering efforts to a limited discussion scope. It is also the basis for defining the cybersecurity properties assigned to the architecture elements.

Regarding product architecture engineering, we rely on the Arcadia model-based engineering method. Arcadia has been implemented in many real-life contexts for several years. Arcadia has experimented in and validated many real-life contexts for several years. Its large adoption in many different engineering contexts demonstrates an industry-proven comprehensive method for systems engineering, capable of adapting to each context in a dedicated manner. Our Arcadia implementation uses the open-source workbench Capella.

Arcadia intensively relies on functional analysis. It introduces four engineering perspectives with specific intents (Figure 1): Operational Analysis, System needs Analysis, Logical Architecture and Physical Architecture. By doing so, it promotes a clear distinction between the need's expression (the first two perspectives) and the solution's expression (the last two perspectives). Each perspective provides a concept set, relations, architectural design tasks, and diagrams, guiding the architect into reaching a good quality design. For details on Arcadia perspectives please refer to Voirin (2017).

Regarding cybersecurity concerns, in this article we avoid referring to a specific standard or methodology. We use a well-known or generic concept set to map domain-specific cybersecurity concepts.

A *Threat Source* is the intent and method targeted at the intentional exploitation of a vulnerability or a situation and method that may accidentally exploit a vulnerability. Threat Sources map to system stakeholders in Arcadia, *Entities* and *Actors*, which are external to the system and to the *Activities and Functions*.

A *Threat* is a situation to avoid that is unwanted by the stakeholders. Different attack kinds, affecting the system Confidentiality, Integrity and Availability (CIA) properties, concretize threats. Arcadia does not include any concept directly mapped to Threats, so it adds the concept. Arcadia's *Scenarios* or (Mis) *Functional Chains* can represent the attacks.

What the stakeholders aim to protect from cyberattacks is what they value the most. This may include tangible items such as system components (software, hardware, devices, networks) and intangible ones such as the services provided by the system, sensitive information stored or manipulated by the system, and the organization's reputation. Cybersecurity standards, such as IEC 62443; Ross, M. McEvilley, and Oren, 2016; and NIST SP 800-30, commonly call these items *Assets*. However, the Product Line Engineering community also uses the term *shared assets* for a different meaning (please refer to the Feature-based Product Line Engineering overview in this edition). To avoid misunderstandings here, we will call *Resources* the tangible and intangible items with potential or actual value to an organisation.

A *Primary Resource* is information or services deemed important by the organisation. Primary Resources relate to Arcadia concepts encapsulating information manipulated and services provided by the system, including *Capabilities, Functions* and *Exchange Items*. A *Supporting Resource* is an item supporting primary resources or security controls. They include information systems, organisations, and premises. In Arcadia, *Components* (including *Actors*) represent such elements. *Security Controls* are the management, operational, and technical controls (safeguards or countermeasures) prescribed to protect the system CIA and its information. In Arcadia, *Functions* and *Components* represent them.

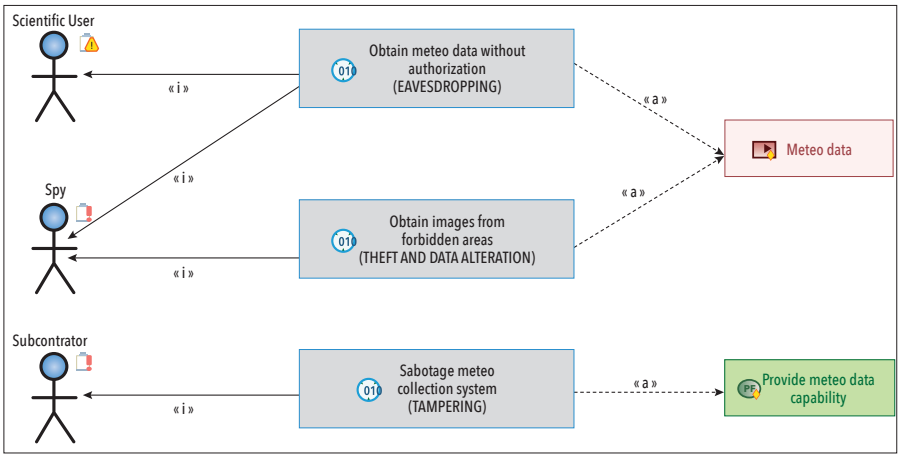



*Figure 2. Threats diagram describing how system actors affect a meteorological balloon system's main mission (provide meteorological data) and sensitive information*

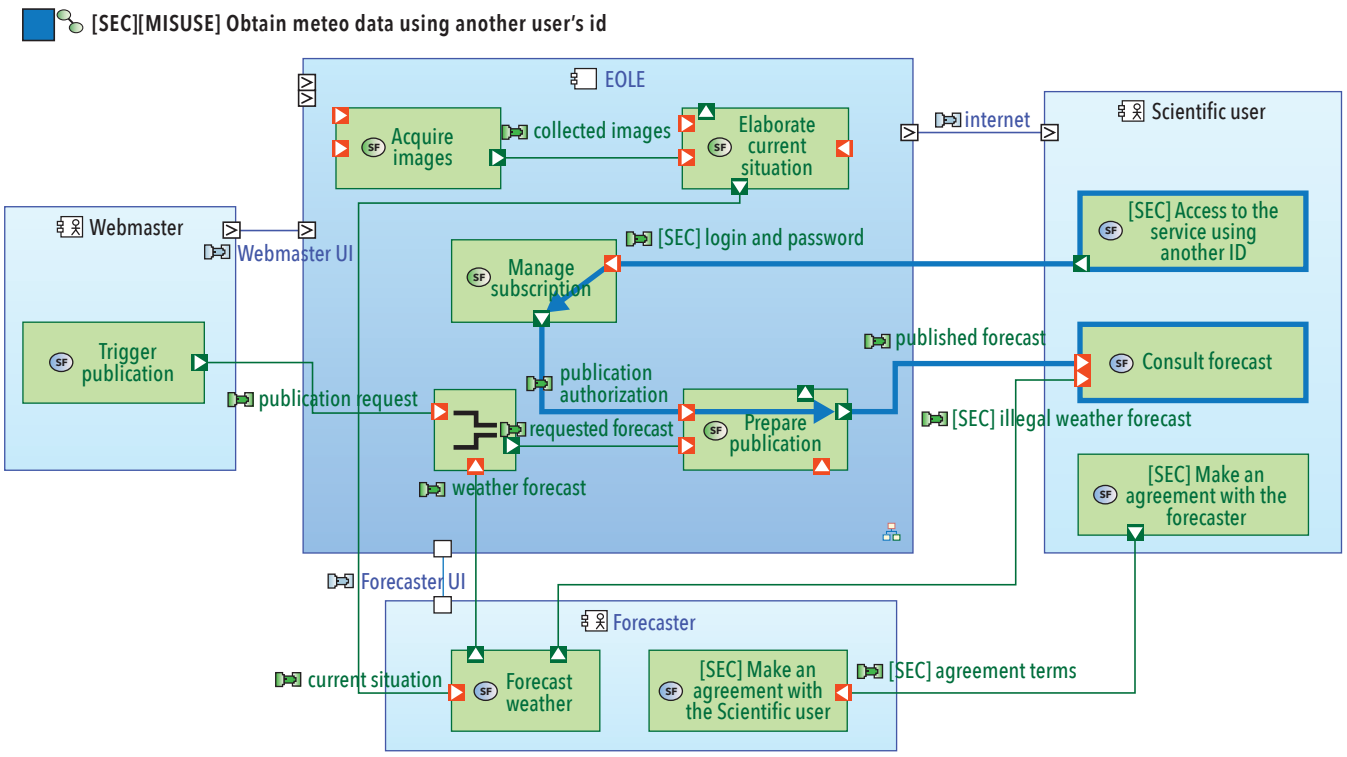



*Figure 3. A Functional Chain (bold blue functional exchanges) describing an attack and the impacted functions: other user IDs can supply meteorological data*

## ANALYZING THE PRODUCT'S CYBERSECURITY CONTEXT AND NEEDS

Systems engineering emphasizes analyzing the problem before jumping straight to the solution, as a means to develop systems effectively contributing to achieving stakeholders' missions. In Arcadia, this analysis occurs in the Operational Analysis and System Analysis perspectives and comprises all the problem space elements, including cybersecurity ones.

Analyzing the product context and the stakeholder expectations leads to: i) identifying threat sources and other malicious agents, ii) defining threat sources goals and intents, iii) identifying the valuable primary resources these agents may attack, and iv) defining the mechanisms threat sources may use to attack the product. Formalizing the threats relevant to the system and its context can use a dedicated Threats diagram such as Figure 2. Figure 3 describes how a specific attack may take place.

This practice, when performed in collaboration, supports the technical dialogue between systems and cybersecurity teams and produces the following results:

- A common and shared comprehension of the operational context in which the product will evolve, the applicable requirements, and the constraints
- Characterizing cybersecurity needs and defining requirements on CIA the product's cyber-protection capabilities shall address
- The product's first cybersecurity-specific features set, related to business (targeted markets, applicable standards, industrial configuration) and/or operational (users and interfaces with other systems) aspects
- Identifying the commonalities and variabilities induced by these features, and how they impact the product architecture definition
- Multi-criteria evaluations including cyber-security aspects, allowing identifying necessary trade-offs and cybersecurity-related requirement prioritizations in the Product Requirements Specification

These results may feed a formal cybersecurity risks analysis beyond this article's scope.

## DESIGNING CYBERSECURITY-AWARE PRODUCT ARCHITECTURES

The solution's design occurs in Arcadia's Logical Architecture and Physical Architecture perspectives. The former aims at defining a preliminary, technology-agnostic product architecture focusing on the expected behavior to fulfill stakeholders' needs. The latter will be the main reference for subsystems and/or components' development teams. It aims at defining the final architecture addressing specific technologies and geographical considerations, and at specifying the interfaces between the subsystems and/or components and with the external actors.

Performing a functional analysis elicits the security controls to implement and secures the product against the identified threats. Integrating cybersecurity functions to the product architecture leads to defining protected services and implementing patterns for managing sensitive data.

Identifying the supporting resources occurs at this point. In a Product Line context it is important to identify how the solution's building blocks (regardless of their nature, previous product version reuse, vendor integration, existing library part) implement the primary resources and how security controls impact them. This leads to defining cybersecurity-related properties applied to the product components and the interfaces between them, as shown in Figure 4.

This practice leads to an architectural design considering and providing proper cybersecurity concern consideration evidence, through the following results:

- A common and shared product architecture comprehension between systems and cybersecurity teams
- Incrementally identifying and characterizing the supporting resources, beginning with the Logical Components and ending with the finer-grained Physical Components
- The product's second cybersecurity-specific feature set, related to

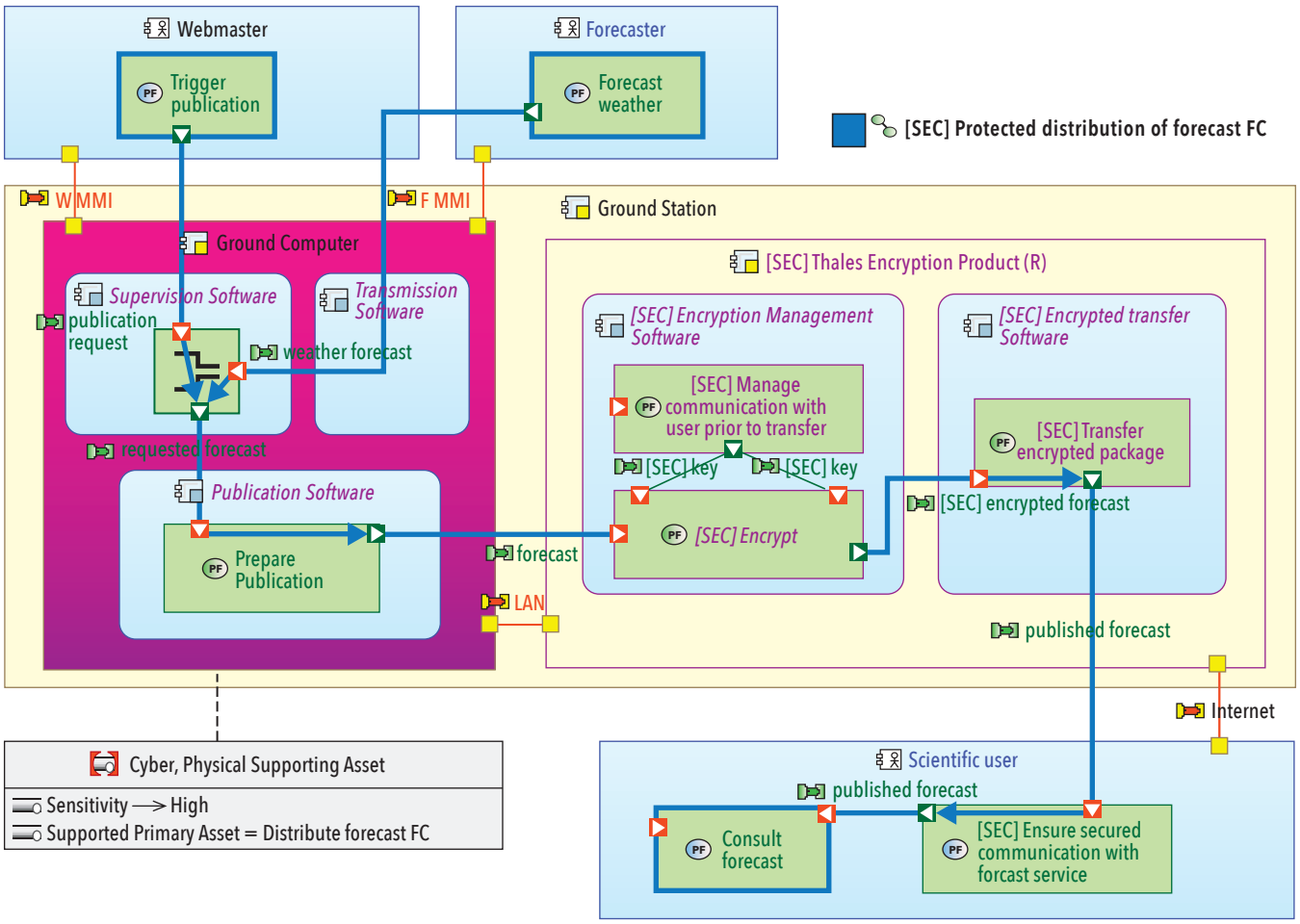



*Figure 4. Cybersecurity aware system's physical architecture. Ground Computer component is a supporting resource, implementing a primary resource (provide meteo data capability), colored (in pink) according to its cybersecurity properties. Thales Encryption Product and its subcomponents are also supporting resources (purple borders) as they implement the security controls represented by the [SEC]-prefixed functions in the diagram*

architectural choices (technologies, reusable building block capabilities)
- Identifying the commonalities and variabilities induced by the whole product feature set, and how they impact defining the final architecture
- Defining product configurations integrating cybersecurity concerns, ready to derive into project-specific architectures
- Multi-criteria evaluations including cyber security aspects, allowing considering cybersecurity constraints while defining the best possible architecture.

These results can feed a formal cybersecurity risks assessment and risk treatment decisions, which may induce architectural modifications, and are beyond this article's scope.

## HANDLING VARIATION MANAGEMENT

The MBSE practices presented here lead to progressively defining the feature catalog, which is the feature superset any given product configuration may include, including cybersecurity-related ones, and the dependencies and/or exclusions between the product features.

The architectural design model will also be the model element superset included in all product configurations. A given configuration's feature choices, the dependencies between the features, the mapping between the features and the model elements, and the dependencies between the model elements either remove from the model or tailor cybersecurity-relevant elements. At this point, it is important to verify the consistency between cybersecurity features and other system-wide features.

In Figure 5 below, the features catalog includes two mandatory features. The Market feature determines the product's selling region (Region 1, Region 2, or both). The Cyber Performance feature relates to using a proprietary encryption component providing high performance, or a third-party component providing low performance (one option between High and Low performance). Choosing Region 2 requires selecting the third-party component (reason may include costs or national production policies).

The architectural model ensures proposing the required cybersecurity-related services in any product configuration. In this particular example, it secures a future encryption component integration with the Ground Computer and the Scientific User by ensuring the physical and functional interfaces are compatible between them. In Figure 5, the forecast service distribution, identified as a primary resource, can happen through either the high or low performance encryption components, as their interfaces are compatible even if the component's internal behavior may differ or even be unknown to the product architect.

## SUMMARY AND PERSPECTIVES

This article presented a model-based engineering practice and technique set enabling an effective co-engineering effort between cybersecurity and systems

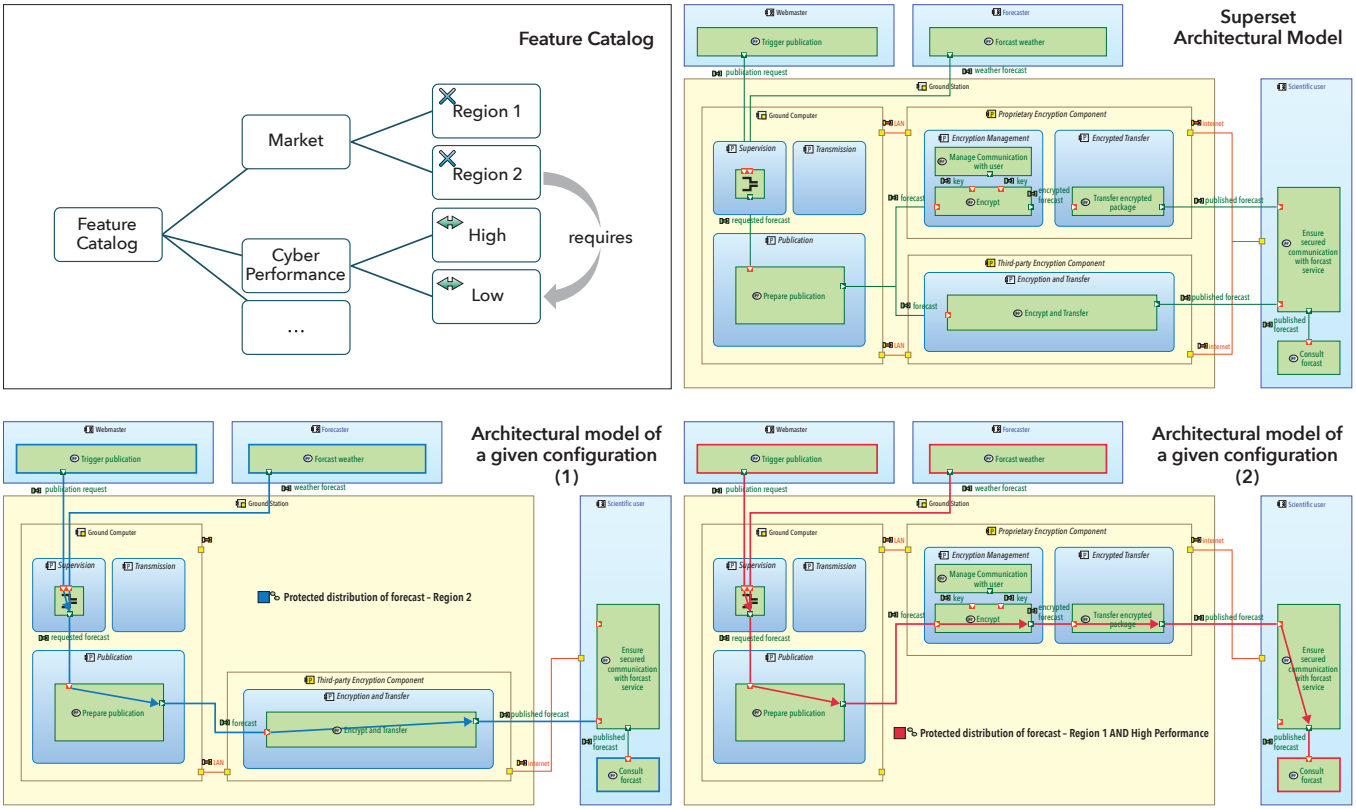



*Figure 5. Top-left: the Feature Catalog. Top-right: the superset architecture model containing all the architectural variants, here the two encryption component types Bottom left and right: two possible product configurations, the first only suitable for Region 2 market, the second is suitable for Region 1 market.*

engineering in a Product Line context. These practices are based on a common vocabulary, allowing collaboration between engineering domains, and on the Arcadia and Capella systems engineering methodology and tool.

One of this work's perspectives is to implement practices to integrate cybersecurity-dedicated models into the product architecture model. In many cases the cybersecurity effort must occur separately from the main systems engineering effort, due either to the high cybersecurity architectural analysis complexity itself, or to confidentiality constraints. To perform system-wide tradeoffs analysis including cybersecurity concerns but also safety, human factors and others, we need an integration model providing the relevant information (and no more) for each concern. ■

## REFERENCES


- AFNOR (Association Française de Normalisation). 2018. AFNOR-XP Z67-140. Information Technology—ARCADIA—Method for Systems Engineering Supported by its Conceptual Modelling Language—General Description: Specification of the Engineering Definition Method and the Modelling Language. Paris, FR: AFNOR.
- Beihoff, B., C. Oster, S. Friedenthal, C. Paredis, D. Kemp, H. Stoewer, D. Nichols, and J. Wade. 2014. *A World in Motion—Systems Engineering Vision 2025*. San Diego, US-CA: INCOSE.
- Capella. 2020. "Capella Homepage." https://www.eclipse.org/capella/
- Voirin, J-L. 2017. Model-Based System and Architecture Engineering with the Arcadia Method. Oxford, UK: Elsevier.
- IEC (International Electrotechnical Commission). 2009. IEC 62443-1-1:2009. Industrial Communication Networks—Network and System Security—Part 1-1: Terminology, Concepts, and Models. Geneva, CH: IEC.
- Joint Task Force Transformation Initiative. 2012. "Guide for Conducting Risk Assessments." NIST Special Publication 800-30.
- Ross, R., M. McEvilley, and J.C. Oren. 2016. "Systems Security Engineering—Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems." NIST Special Publication 800-160 1: 1-260.


## ABOUT THE AUTHORS

**Juan Navas** is in charge of the modelling & simulation coaching team at Thales Corporate. He accompanies systems engineering managers and systems architects to implement model-based systems engineering (MBSE) and product-line engineering (PLE) approaches on operational projects and product roadmaps. He has over 12 years of experience applying systems engineering practices in oil & gas, telecommunications, nuclear, defense and aerospace industries. He holds a PhD in computer science, a Master degree in software-intensive and automation systems, and Engineering degrees in electronics and electrical engineering.

> continued on page 50

# Integrating Security into Enterprise Architecture with UAF and PLE

**Matthew Hause,** mhause@designxi.com


■ **ABSTRACT**
At the systems of systems and enterprise levels, systems engineers and architects must plan for system security from system concept inception ensuring security embeds into every process, procedure, system, and component as well as the enterprise's mindset. While the various United States Department of Defense Architecture Framework (DoDAF) views contain security attributes, there is no integrated view set defining system security goals, threats, risks, mitigating elements, and demonstrating how these integrate and implement into the operational, system, standards, and services views. The Unified Architecture Framework (UAF) implements DoDAF using the Systems Modeling Language (SysML) as well as the British Ministry of Defence Architecture Framework (MODAF) and NATO Architecture Framework. In addition, UAF has integrated a security view set facilitating engineers defining security goals and requirements and demonstrating how these implement throughout the architecture. By using these integrated security views, engineers can design system protection as well as system protection options and variants.

■ **KEYWORDS:** Security, UAF, MBSE, SoS, Modeling, Architecture Frameworks

## INTRODUCTION

The INCOSE *Systems Engineering Vision 2025* defines model-based systems engineering (MBSE) as "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (INCOSE 2007). The Systems Modeling Language (SysML) is the most widely used standardized systems modeling language and notation. It models systems in both the abstract and concrete (logical and physical) views including behavioral, structural, parametric, and requirements views (Object Management Group 2013). For enterprise modeling, understanding systems of systems and how they change over time requires an architecture framework. DoDAF is the United States Department of Defense Architecture Framework and MODAF is the Ministry of Defence Architecture Framework. The Unified Architecture Framework (UAF) builds on SysML and defines the overall goals, strategies, capabilities, interactions, standards, operational and systems architecture, and systems patterns (UAF 2019). Security and human factors (personnel) views added to DoDAF and MODAF improve these frameworks' coverage. The Object Management Group (OMG) ratified the UAF, previously called the Unified Profile for DoDAF and MODAF (UPDM). Several papers cover the UPDM/UAF and its system of systems (SoS) modeling support including Hause, and Dandashi 2015 and Hause 2014. This paper does not include full SysML and UAF details for space reasons. Please see the references for more information. The purpose here is to describe the UAF security views and how they can describe system features and variants for applying security throughout the enterprise and over time.

## MOTIVATION FOR THE UPDM/UAF

UPDM provided a standardized DoDAF and MODAF frameworks expression means using a common metamodel, and provided interoperability between the frameworks. UPDM also leveraged SysML's extensibility to include new concepts including complete views.

### *UAF Views*

Before modeling a system or system of systems, one must understand both the system and model's purposes. UAF has a view set defining a capability set over its life-cycle phases. These define the goals, vision, enterprise phases, evolution over time, the capabilities, and how systems and subsystems realize these. The UAF provides traceability from these elements to the other UAF views including the operational architecture which defines the system's abstract, logical, and solution independent expression. This defines what must happen

and traces directly to the systems views defining how to realize these capabilities and operational architectures. To use an analogy, the operational view could define a need to generate power, and the systems views define fossil fuel, solar, wind, tidal, and other power providing methods. Standards views define system standards and systems conforming to them, services views define services systems will implement, and the project views define when to deploy and retire the systems. In addition, the UAF's latest version also defines security and human factors views. The systems assurance group at the OMG is also working to integrate threat and risk analysis as a cross cutting concern set.

## ARCHITECTURE CROSS CUTTING CONCERNS

Cross cutting concerns are an architecture's non-modular characteristics which cut across other aspects. A simple example would be vehicular safety. When designing a car, there is no specific safety module car component. Safety must be inherent and intrinsic to the car design and implementation or the car will not be safe. Furthermore, overall safety performance, as well as the vehicle's operating environment, is the vehicle operator's responsibility. In the same way, a system of systems contains various cross cutting concerns we must address. These include security, safety, resilience, flexibility, robustness, and others. Defining the vulnerability points for security and resilience allows engineers to perform trade-off and threat and risk analysis on the entire architecture. Integrating the analysis tools with the UAF architecture provides a means of defining the problem, designing possible solutions, and then performing trade-off analysis to determine the best fit.

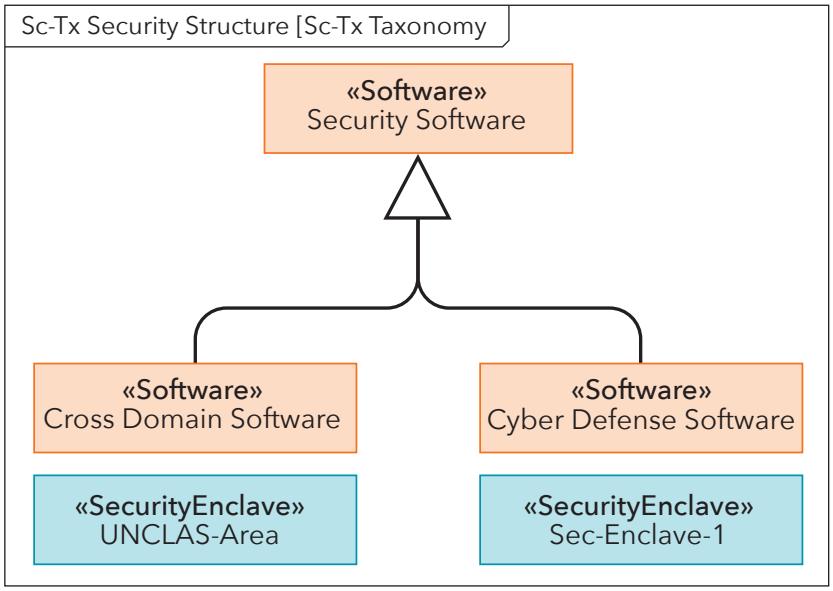



*Figure 1. Security taxonomy for the search and rescue architecture*

These possible solutions can narrow to a single solution implemented in the final system. They can also provide an alternative set linked to system features providing a product line or product line family.

## THE UAF SECURITY VIEWS

The DoDAF and MODAF lacked security views. Adding these to the UAF helped define requirements, strategies, implementations, and solutions for all security forms throughout the enterprise. The UAF security views illustrate the security assets, security constraints, security controls, families, and measures required to address specific security concerns. Their purpose is to address the security constraints and information assurance attributes existing on exchanges between systems and operational elements. The stakeholders for these views include security architects, security engineers. systems engineers, and operational architects.

### *The Example Model*

The example model shown below applies the UAF to a common civilian maritime Search and Rescue (SAR) operations scenario—a Yacht in distress. A Monitor Unit picks up the yacht's distress signal and passes it on to the Command and Control (C2) Center. The C2 Center coordinates the search and rescue operation among the Rescue Helicopter, a Naval Ship, and a Rescue Boat. A UK Ministry of Defence example model is this model's base. The system contains a systems' set with different stakeholders, owners, command hierar-

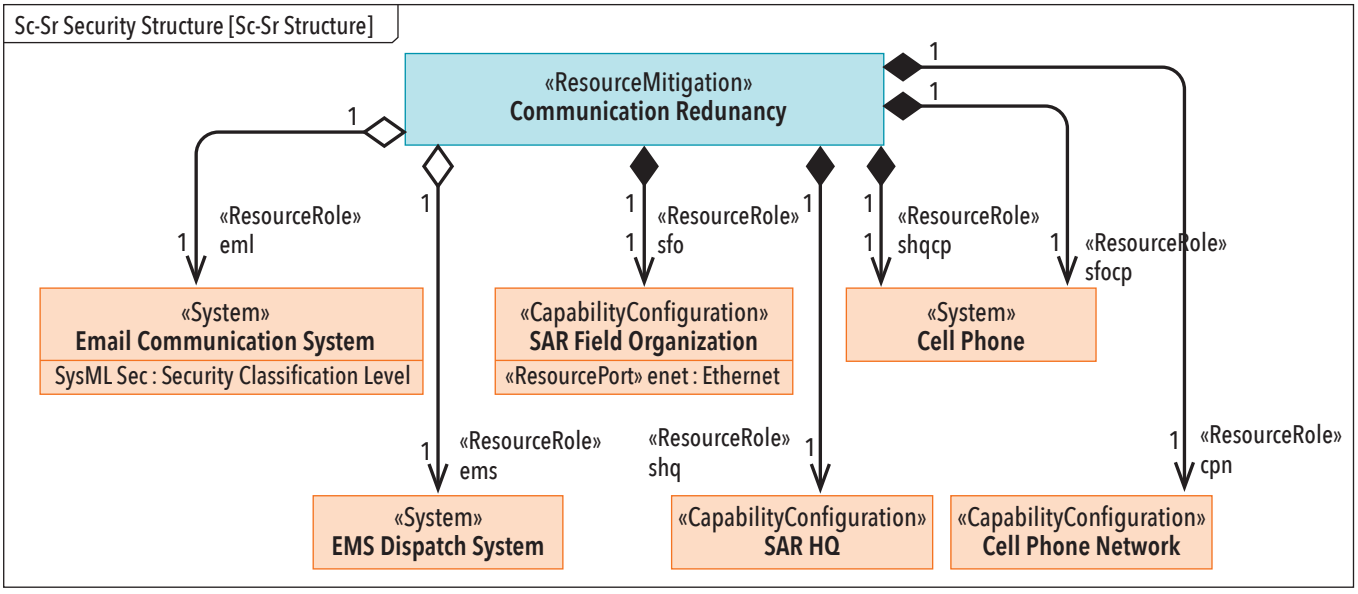



*Figure 2. Security structure for communication redundancy*

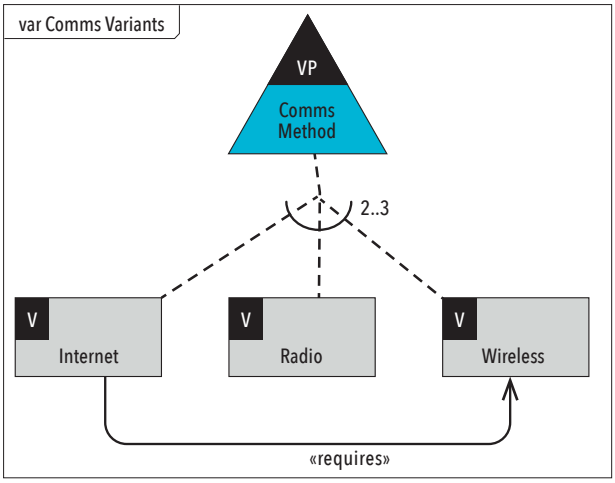



*Figure 3. System communication variability*

chies, purposes, security and safety levels, and constraints. In short it is a complex system of systems. Communications and interactions involve naval vessels and helicopters, first responders, civilian and federal government vessels and vehicles. There is a need to communicate and cooperate, but also a need to ensure system, personnel, and communication security. The example will demonstrate how the UAF security views can define secure architecture. For the complete model, refer to the example model in the UAF specification (Object Management Group 2019).

The Security Taxonomy (Sc-Tx) domain shows the security assets and security enclaves. The diagram defines the security asset and asset owner hierarchy available to implement located (security enclaves) security, security constraints (policy, guidance, laws and regulations) and details as Figure 1 shows.

In addition to the security elements, the Security Taxonomy diagram can also define system resources. In this way the different security elements can group in the same diagram and package hierarchy. Since the UAF does not constrain where to define and store elements, this contributes to more modular architectures. Figure 1 shows the security software resources and security enclaves.

The security structure captures asset allocation (operational and resource, information and data) across the security enclaves, showing applicable security controls necessary to protect organizations, systems, and information during processing, while in storage, and during transmission. It also captures Asset Aggregation and allocates aggregated information usage at a location as shown in Figure 2.

A Resource Mitigation is a security measure set intended to address specific cyber risks. It comprises a tailored security control subset protecting the asset at resource (Resource Role). In this case, Communication Redundancy comprises the SAR Field Organization, SAR HQ, and communication technology. The communication technology choices are the email communication system, EMS dispatch system, and the cell phone network. Not all these systems will take part in the final configuration but this stage does include them as trade-off analysis will compare them. Along with performance and cost, security controls, levels, and methods can, in the evaluation, compare the communication method efficacy. Product line engineering can define the available trade-off analysis choices.

### *Variability*

The communication method is an architecture's feature with communication elements to allow communication as shown in Figure 3.

Figure 3 shows the usable comms method variations. These include the internet,

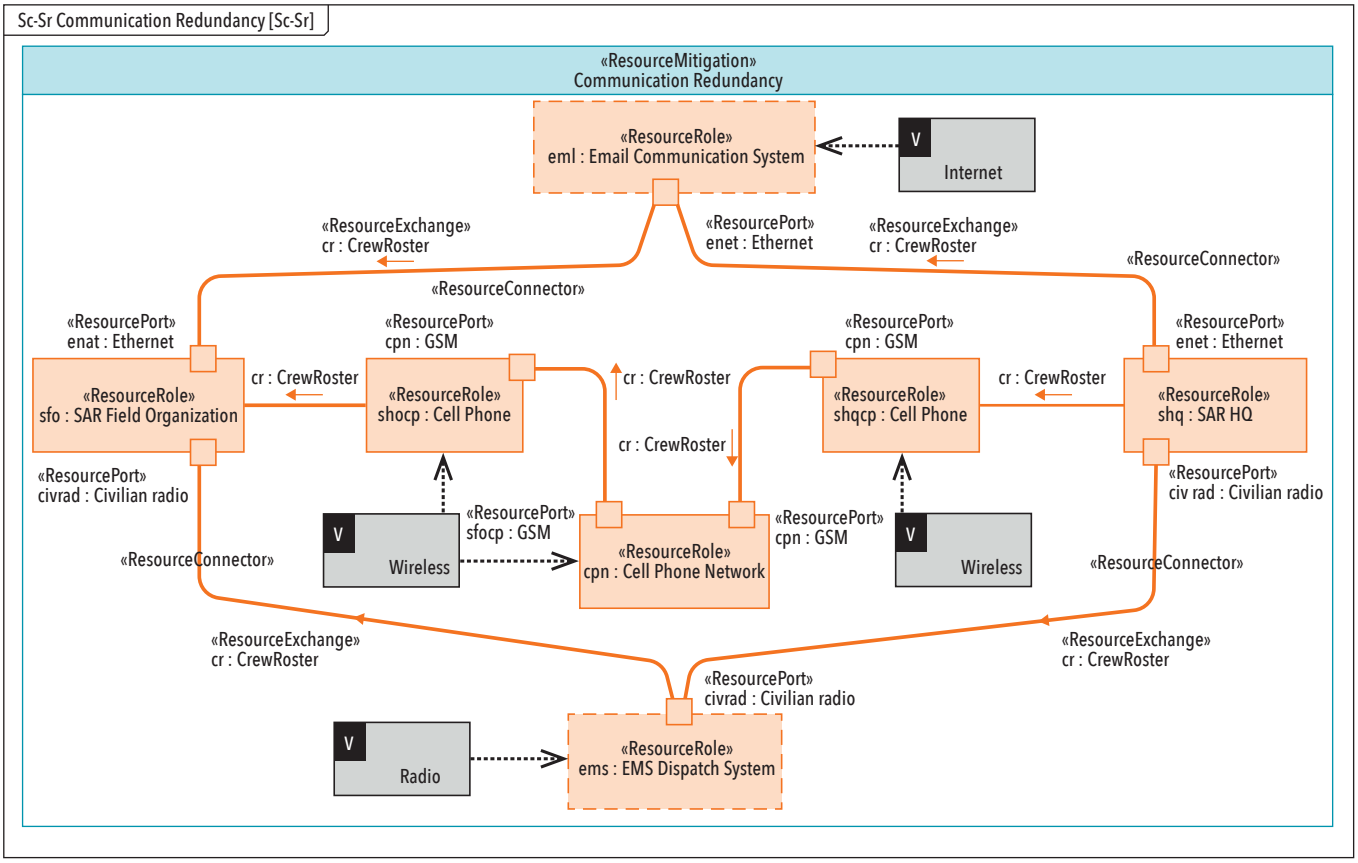



*Figure 4. Internal communication redundancy view*

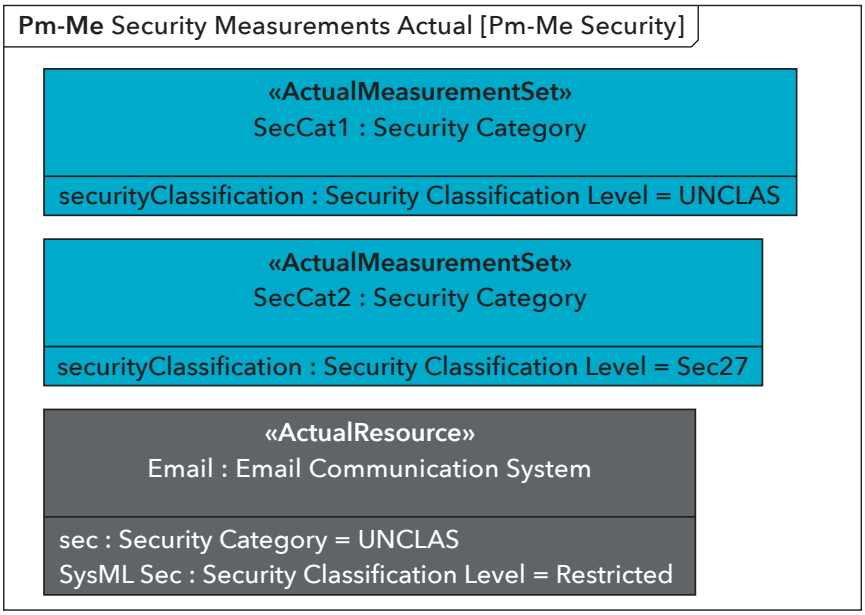



*Figure 5. Actual security measurements and actual resource with measurements*

radio, and wireless. The multiplicity, shown as 2..3, requires using at least two and not more than three. This ensures a backup method in case the main one fails. This enables the communication redundancy resource mitigation defined in Figure 2. In addition, Figure 3 shows the Internet requires Wireless as a solution part. Because some locations are remote, wired internet connections are not always available.

The security connectivity view lists security exchanges across security assets, the applicable security controls, and the security enclaves housing the exchange's producers and consumers. Figure 4 shows the internal structure for the communication redundancy resource mitigation.

Figure 2 described the structural communication redundancy breakdown. Figure 4 shows how the parts communicate. In this case the crew roster must distribute from the SAR HQ to the SAR Field Organization. Figure 2 shows the communication paths at a very high level. For example, the crew roster travels from the SAR HQ to the cell phone to the cell phone network to the Field Organization through the other cell phone. The diagram can automatically and directly generate interface control. In addition, the variants defined in Figure 3, can link to the various elements. The engineer selects the required variants and generates a product model removing the other elements. This applies to all model parts including functional elements such as activities, operations, and parameters.

The architecture can define and reuse measurement definitions and actual measurements. They can link to systems, activities, and interactions as well as directly integrate into systems as shown in Figure 5.

Figure 5 shows an actual measurement set defining the security categories of unclassified and Sec27. It also shows the email communication system's actual resource with its security category and the security classification level.

The security processes view provides a security control set and any possible enhancements as applicable to assets. Figure 6 shows a security control activity set, the software performing them, and the assets they protect.

In addition, defined security processes can execute behaviors associated with security as shown in Figure 7. The activity diagram describes operational or resource level processes applying (operational level) or implementing (resource level) security controls/enhancements to assets located in and across enclaves. This demonstrates interactions crossing security levels and in and out of systems. The security processes can demonstrate how to protect the data as well as the assets themselves.

Figure 7 shows an activity set taking place to access the SAR system. These can, once added to operational activity diagrams, demonstrate logical security measure requirements as well as system function activity diagrams to describe

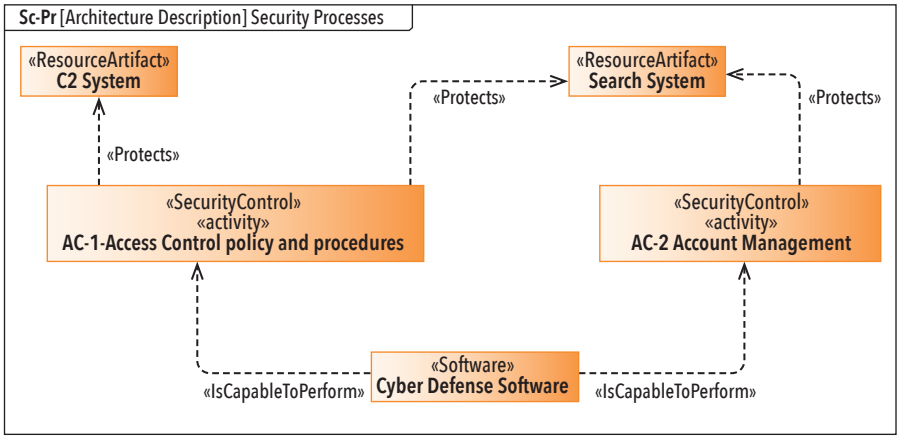



*Figure 6. Processes, the elements performing them, and the things they protect*

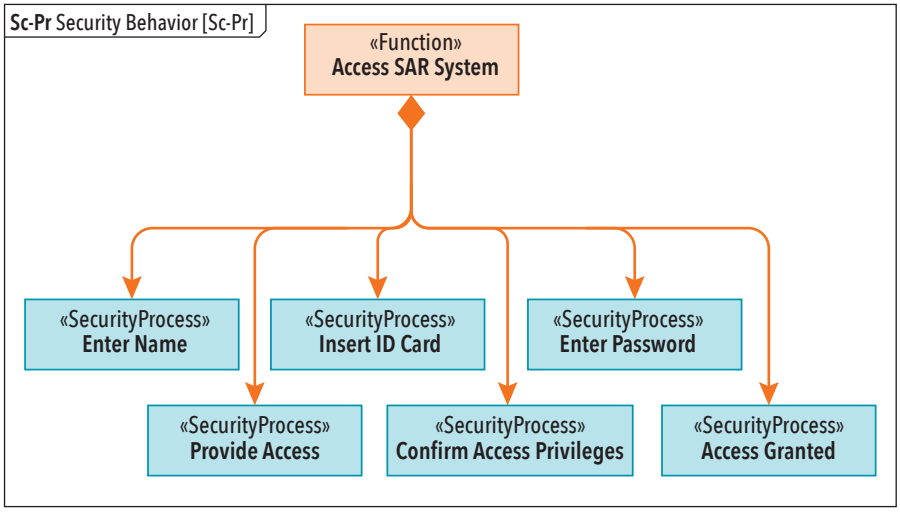



*Figure 7. Security behavior as part of system functions*

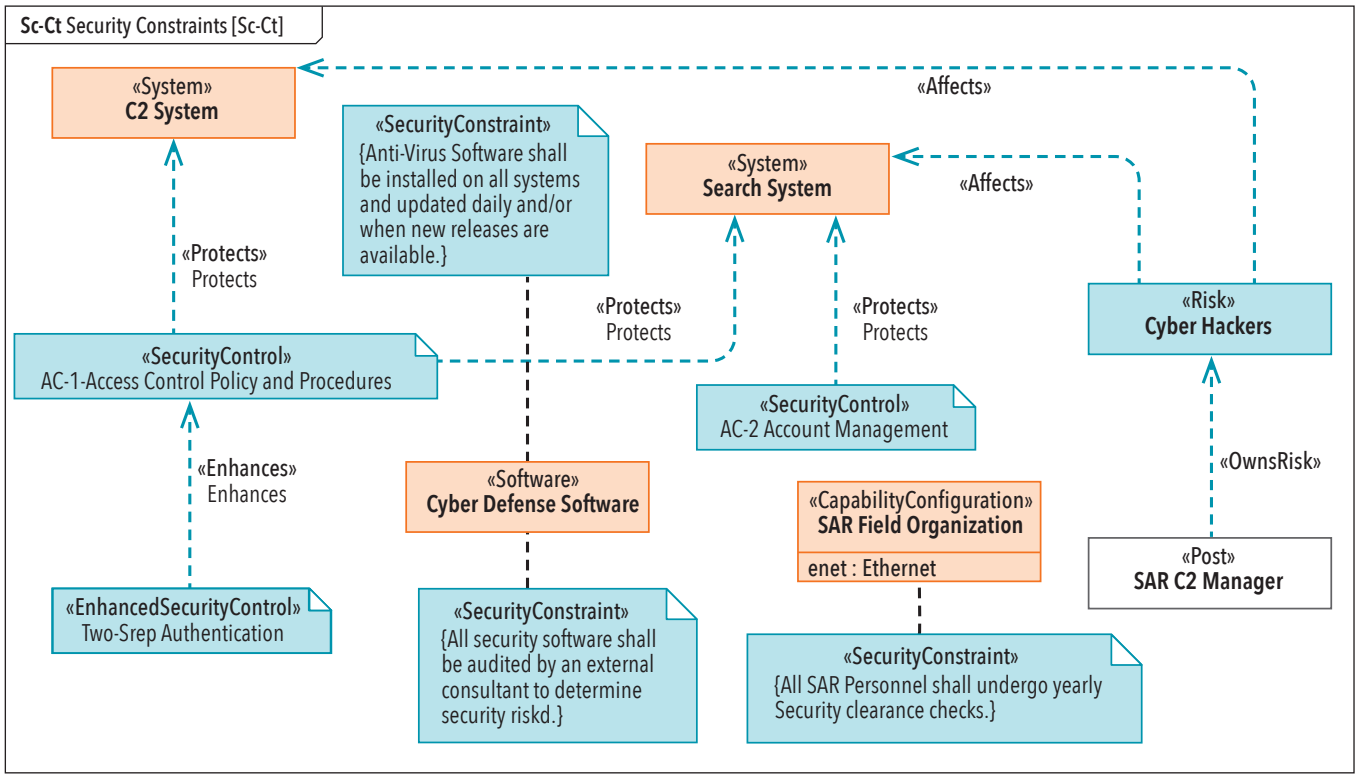



*Figure 8. Security constraints for the search and rescue*

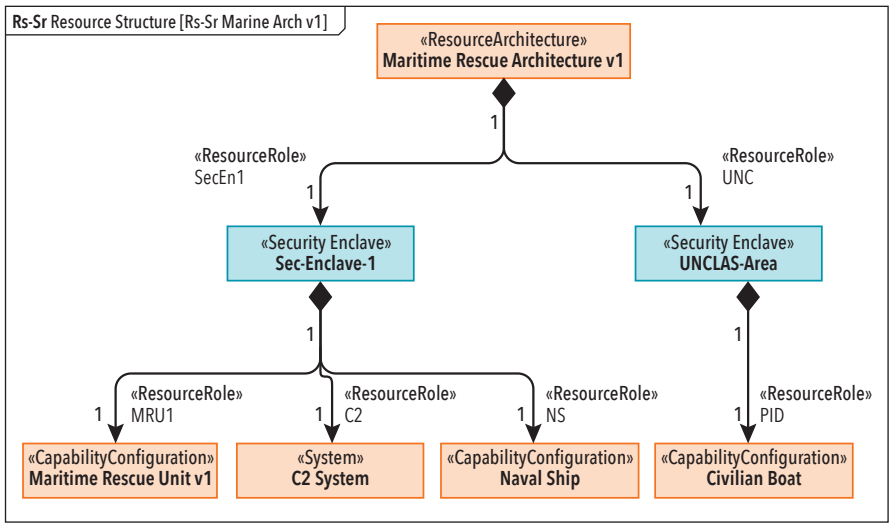



*Figure 9. Maritime rescue architecture with security enclaves and systems*

specific security measures and technologies. State diagrams can also show them to describe state-based security behavior.

The security constraints view specifies textual rules/non-functional requirements as security constraints on resources, information, and data (security-related in the form of rules, access control policy). Identifying risks and specifying risk likelihood, impact, asset criticality, and other measurements enables risk assessment. Figure 8 shows the Search and Rescue architecture security constraints.

Figure 8 shows the systems, software, and their relationships to the security constraints and controls. It also shows the risks and the systems they affect as well as the person who owns the risks. An enhanced security control demonstrates how to add additional security, in this case, adding two-step authentication to the access control policy and procedures.

The Security Traceability domain depicts mapping a risk to the: risk owner, risk mitigations, and affected asset roles. Security Controls to Risks Mapping matrix shows which operational or resource asset roles mitigate risks and represents the Security Traceability. Risks to Assets Mapping matrix shows which risks apply to Asset Roles. The model automatically generates these and is beyond the scope of this article.

### *Security Integrated into Other views*

As stated earlier, security is a cross cutting construct. As a result, it must integrate into the architecture. The previous security figures demonstrate this. UAF also defines security elements in other UAF views. Figure 9 shows a Maritime Rescue Architecture resource/systems diagram.

The figure shows the Maritime Rescue Architecture's structural breakdown with two different level security enclaves. The rescue and control systems are in security enclave 1 and the civilian boat is in the unclassified area. Figures 3 and 4 shows how to include other security elements such as security mitigations.

Figure 10 (on the next page) shows the Maritime Rescue Architecture's internal structure.

Figure 10 shows the security enclaves; the enclosed systems, colored blue to emphasize their location and extent; and their classification levels. The right shows the civilian boat, personnel, and communication systems. The left shows the rescue systems, personnel, and communication systems.

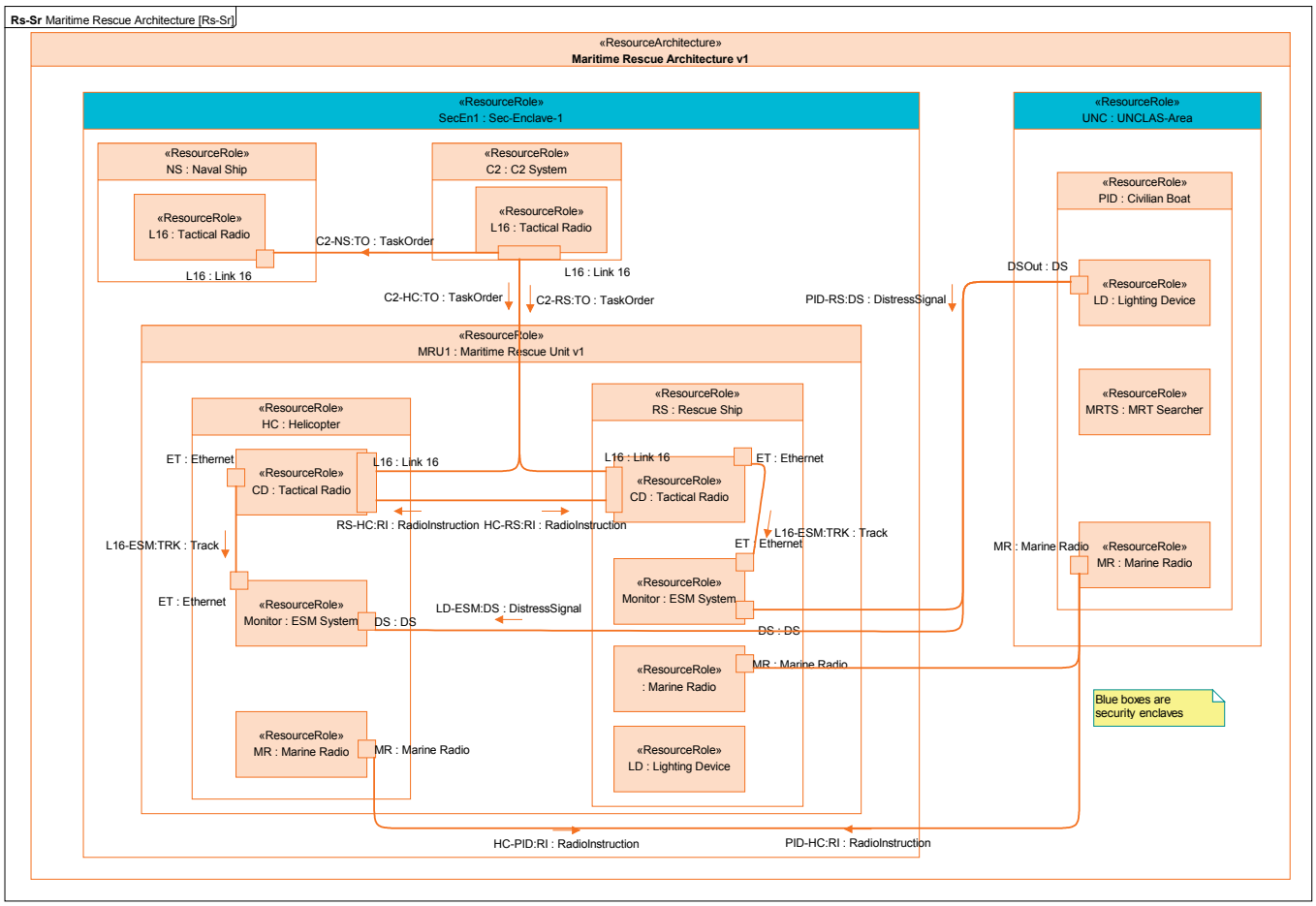



*Figure 10. Maritime rescue architecture internal view*

This emphasizes the data and interactions across the security boundaries. PLE can demonstrate various configurations and security options.

## SUMMARY AND CONCLUSIONS

The UAF describes enterprise and system architectures and an integrated security view set. The security views define the security requirements and issues at the project's start in a separate view set. They also integrate security into the different views to highlight security vulnerabilities and demonstrate how to mitigate them. These security views provide the architecture options expressed to assist in trade off analysis and alternative evaluation. The UAF views promote a proactive cyber security and cyber resilience treatment in the architecture during development. Resource mitigation defines the alternatives for mitigating security risks in the architecture. The measures shown in the UAF sample problem show the benefit of addressing vulnerabilities while developing the architecture. These provide a quantitative and qualitative security alternative analysis. ■

## REFERENCES


- Dahmann, J., G. Rebovich, J. Lane, and R. Lowry. 2010. "System Engineering Artifacts for SoS." Paper presented at the Fourth Annual Institute of Electrical and Electronics Engineers Systems Conference, San Diego, US-CA, 5-8 April.
- Department of Defense (DoD). 2013. "Defense Acquisition Guidebook." http://at.dod.mil/docs/DefenseAcquisitionGuidebook.pdf
- Hause, M. 2014. "SOS for SoS: A New Paradigm for System of Systems Modeling." Paper presented at the Institute of Electrical and Electronics Engineers and American Institute of Aeronautics and Astronautics Aerospace Conference, Big Sky, US-MT, 1-8 March.
- Hause, M., and F. Dandashi, 2015. "UAF for System of Systems Modeling, Systems Conference (SysCon)." Paper Presented at the Ninth Annual Institute of Electrical and Electronics Engineers Systems Conference, Vancouver, CA-BC, 13-16 April.
- INCOSE. 2007. "Systems Engineering Vision 2020." http://www.ccose.org/media/upload/SEVision2020_20071003_v2_03.pdf.
- Object Management Group. 2012. OMG2012-06-01. OMG Systems Modeling Language (OMG SysML™), V1.3, http://www.omg.org/spec/SysML/1.3/PDF/.
- ——. 2013. OMG2013-08-04:2013. Unified Profile for DoDAF/MODAF (UPDM) V2.1, http://www.omg.org/spec/UPDM/2.1/PDF
- ——. 2019. "The Unified Architecture Framework (UAF)." https://www.omg.org/spec/UAF

## ABOUT THE AUTHOR

**Matthew Hause** is a principal at Systems Security Innovation (SSI), the INCOSE liaison to the OMG UAF group, a member of the OMG architecture board for 12 years, and a member of the OMG SysML specification team. He has been developing multi-national complex systems for over 40 years. He started out working in the power systems industry then transitioned to command and control systems, process control, communications, SCADA, military systems, energy management systems, distributed control, and many other areas of technical and real-time systems. He is an expert in SysML, UML, UAF, systems engineering and architecture frameworks. His role at SSI includes consulting, mentoring, presentations at conferences, and developing and presenting training courses.

---

**Navas et al.** continued from page 43

**Jean-Luc Voirin** is Director, Engineering and Modeling, in Thales Defense Missions Systems business unit and Technical Directorate. He holds a MSc & Engineering degree from ENST Bretagne, France. His fields of interests include architecting, system engineering, modeling and simulation. He has been an architect of real-time and near real-time computing and mission systems on civil and mission aircraft and fighters. He is the principal author of the Arcadia method and an active contributor to the definition of methods and tools. He is involved in coaching activities across all Thales business units, and in charge of research on future engineering tooled processes.

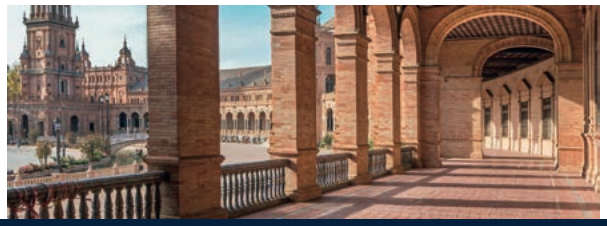
**Stéphane Paul** has a Masters in computer science (1988) and PhD in microelectronics (1991). In 1993, he joined Alcatel to work on object-oriented analysis. Starting from 1996, he was involved in European R&D projects, dealing with advanced surface movement guidance and control systems (A-SMGCS). From 2000 to 2008, he acted as technical authority on airport systems at Thales ATM's technical directorate. From 2008 to mid-2010, he was head of the Collaborative Technologies Laboratory at Thales Research & Technology. From July 2010, he started research on model-driven security engineering, with a particular focus on security risk assessment, methods and tools. He was also involved in safety & security co-engineering, system of systems engineering (SoSE with UPDM), security for service-oriented architecture, and security solutions for critical embedded systems. He delivers a 3-day cybersecurity engineering course within Thales since mid-2016. He developed an agile approach to risk assessment. In parallel, he provides risk assessment support to different Thales entities. Outside of Thales, during 2014-17, he also gave UML courses to Master 2 students, and since 2019 teaches risk management to Master 1 students at ESME Sudria (Paris). He is a certified adult trainer since Jan 2020.

**Stéphane Bonnet** is systems design authority for the avionics business unit of Thales. He oversees system architectures for all business lines and leads the system-level R&T. Over the last 15 years, he has played a key role in the development of the Capella and Arcadia MBSE open solution, and has spent a considerable amount of time helping engineering managers and systems architects implement the MBSE cultural change, with a range of activities spanning from strategic engineering transformation planning to project-dedicated assistance to modeling objectives definition and monitoring.

# 2021 Annual INCOSE international workshop








join us for the first virtual

## Annual INCOSE International Workshop

29 - 31 January 2021

www.incose.org/iw2021

## 2020 KEY NUMBERS

**170**
Meetings

**618h**
of Productive Workshop

**13h**
of Social Events

# Next regional events

Don't miss an opportunity to join one of the regional conferences

SEPT

- **INCOSE Japan Symposium 2020 Virtual Event - hosted by JCOSE**
  *02 - 03 September 2020*
  Japan
  www.js2020.online/

- **2020 Annual INCOSE Western States Regional Conference (WSRC) - Now Virtual!**
  *17 - 19 September 2020*
  United States
  www.incose.org/wsrc2020

- **INCOSE Brasil Virtual Conference 2020**
  *28 - 29 September 2020*
  Brasil
  incose.com.br/conference/

OCT

- **CSER 2020 - Conference on Systems Engineering Research**
  *08 - 11 October 2020*
  United States
  cser2020.org

- **2nd Annual INCOSE New England Fall Workshop**
  *15 - 17 October 2020*
  United States
  www.neincose.org/incose-ne-fall-workshop-2020

- **HSI 2020 Workshop**
  *27 - 29 October 2020*
  Israel

APR

- **14th Annual INCOSE Great Lakes Regional Conference (GLRC14)**
  *11 - 14 April 2021*
  Detroit, United States
  www.incose.org/glrc