Model-Based Design for Affordability of a Netted Intelligence, Surveillance, and Reconnaissance Concept

K. Dewayne Brown, Mary Beth A. Chipkevich, Robert J. Bamberger, Tam-Thanh C. Huang, Mark A. Matties, James D. Reeves, and Christopher A. Rouff

ABSTRACT

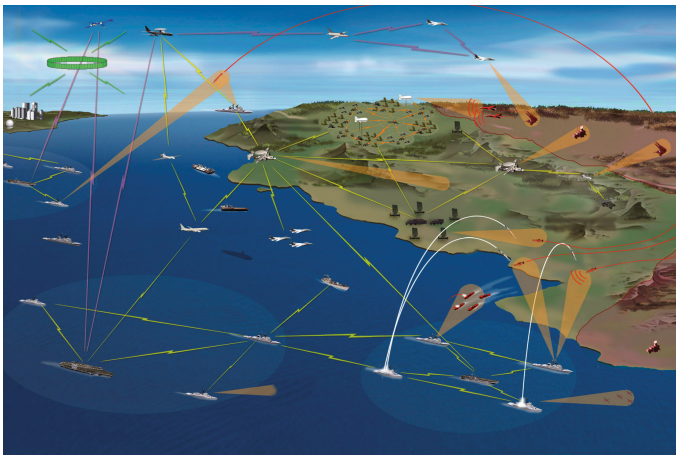
This article highlights the critical importance of systems engineering methodology and its influence on downstream outcomes in complex engineering concepts. It makes a development case for complex systems of systems that contrasts netted with traditional intelligence, surveillance, and reconnaissance. Model-based systems engineering methods supported by development infrastructure can significantly impact the life-cycle affordability of complex systems of systems. The long-standing systems engineering practices of the Johns Hopkins University Applied Physics Laboratory (APL), the International Council on Systems Engineering, and the Open Group Future Airborne Capability Environment Consortium, as well as the objectives for projects such as the Defense Advanced Research Projects Agency Air Dominance Initiative, provide context for an APL brand of model-based methods. This article introduces an APL model-based systems engineering methodology within an integrated development environment and discusses the methodology in the context of a netted intelligence, surveillance, and reconnaissance concept.

INTRODUCTION

More than 70% of program life-cycle costs are determined in the concept engineering phases. A model-based systems engineering (MBSE) methodology, integrated development environment (IDE), and systems engineering (SE) infrastructure provide engineers the capability to cost-effectively explore, model, prototype, and validate concepts. A case study for intelligence, surveillance, and reconnaissance (ISR) concept engineering piloted use of this capability, and it was determined that one concept was more affordable (in terms of life-cycle costs) and rapidly realizable (in terms of development time lines) than an alternative concept.

BACKGROUND

The overall complexity of national security and operational warfare is increasing.[1] Complexity of the systems of systems (SoS) needed to deliver mission capability is also increasing, particularly with regard to the amount of information exchanged, the degree of interaction among battle force units, and the battlespace environment.[2] Concurrently, DoD budgets are trending downward.[1] Affordability is the new number one threat to developing, delivering, integrating, and sustaining needed capabilities.[3] The DoD has recognized the need to reduce life-cycle costs early in the development of concepts and capabilities, as evidenced by various studies and initiatives (see Fig. 1).



The Reality…

"Our current security challenges are more formidable and complex than those we faced in downturns following Korea, Vietnam, and the Cold War. There is no foreseeable 'peace dividend' on our horizon."

—GEN DEMPSEY, CJCS
Testimony to SASC, 12 Feb 2013

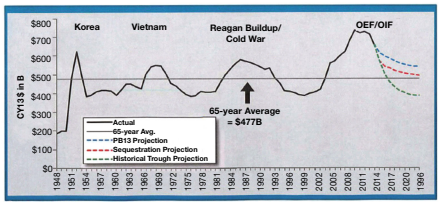


Figure 1. Battlespace complexity and affordability. [Right image from G. Boltz (APL) and left from Ref. 1.]

MOTIVATION

Applying SE practices provides necessary structure throughout the life cycle of a system. Supporting model-based infrastructure enables effective management of performance, schedule, and cost over significant portions of the concept-development life cycle, including exploration, verification, and validation. Models facilitate efficient expression of system requirements, structure, behavior, and function along with rapid communication, distribution, and functional decomposition of complex systems. Model-based methods enable effective, rapid, and low-cost decisions to be made among alternatives when engineering a complex SoS. In this case study, application of a model-based methodology demonstrates the ability of these methods to significantly impact the life-cycle affordability of ISR concepts.

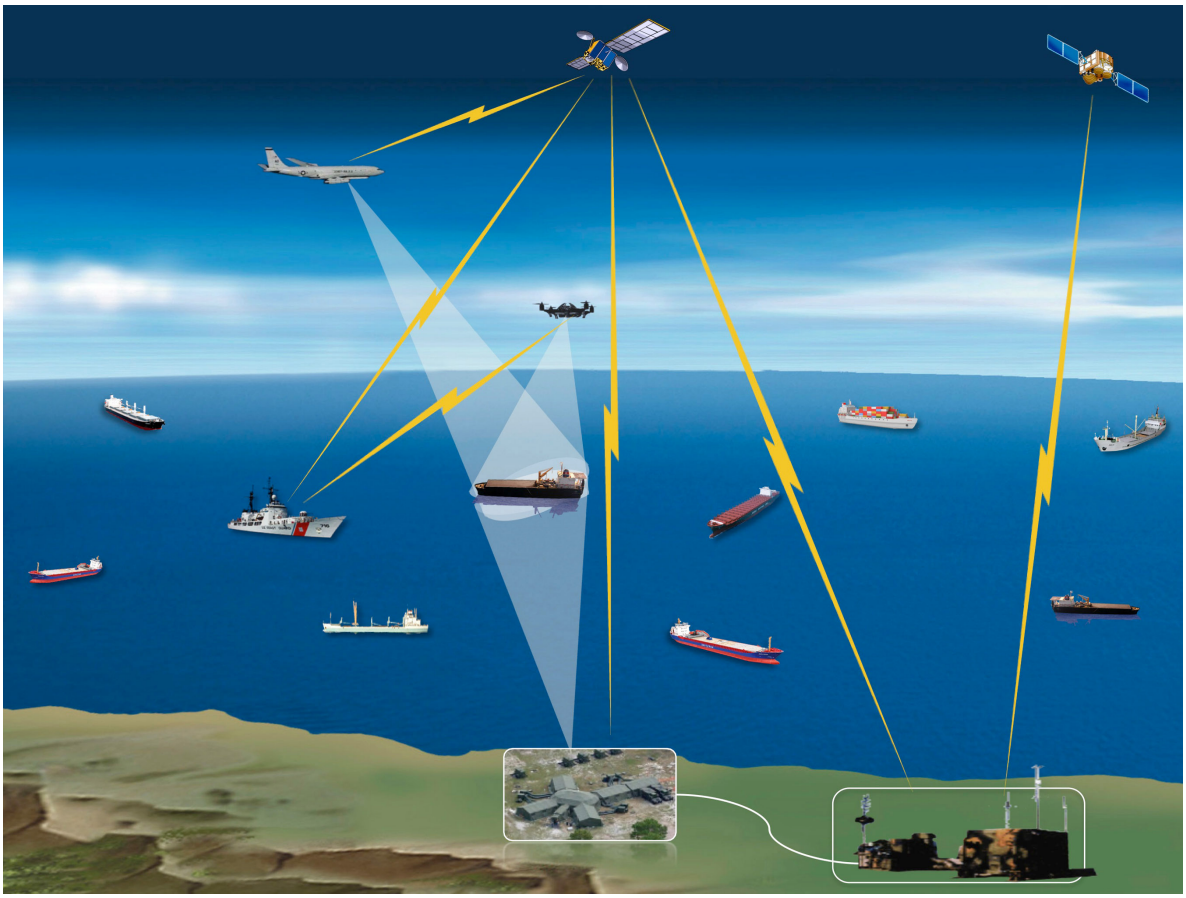


Figure 2. A maritime interdiction scenario.

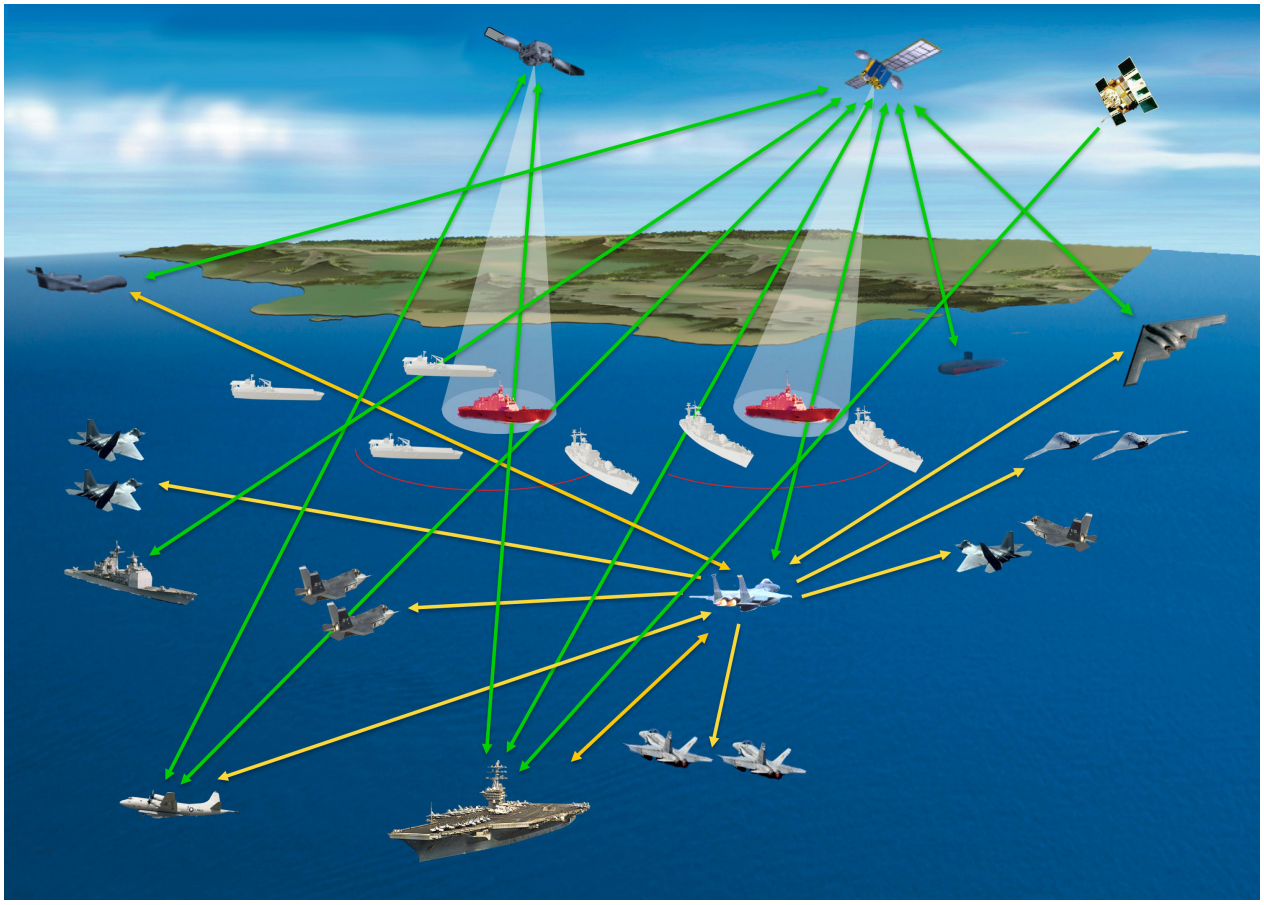


Figure 3. Anti-access/area-denial maritime dynamic targeting scenario.

THE ISR CASE: THE CHALLENGE

Warfighters need a capability to command and operate an ISR SoS more effectively and affordably than they can today. Tasking–collection–processing–exploitation–dissemination (TCPED) is a process that uses an SoS during military operations to achieve ISR mission objectives determined by commander's intent. The tasking–collection (TC) portion of TCPED encompasses actions and decisions related to planning for, tasking of, and operation of systems that contribute to ISR mission objectives. The increasing number and diversity of ISR assets as well as competing ISR mission requirements challenge the effectiveness of existing doctrine, processes, and systems to command and manage the use of ISR platforms and sensors.[4,5] The volume and multiple modalities of sensor data can overwhelm the processing–exploitation–dissemination (PED) portion of TCPED.[5]

"As-is" traditional ISR SoS TCPED processes have time-consuming and manpower-intensive TC and PED activities and require high-bandwidth reachback communications. As-is TC requires mission control operators at ground control stations or on manned airborne, surface, or subsurface platforms to perform sensor control and platform control of assigned "stovepiped" ISR assets in a reactive rather than a predictive manner. This reduces time available to properly respond to a situation and suboptimizes overall ISR SoS performance. As-is PED involves transfer of raw sensor data via reachback communications to intelligence community facilities. Analysts use available systems at these locations to process, exploit, and disseminate fused intelligence products for use by command-and-control decision makers.

In dense battlespace environments and with increasingly complex SoSs, synchronization among nodes in the SoSs operated by naval, joint, and coalition forces requires timely and tight coupling with operations, targeting, and intelligence operational doctrine to deliver the ISR targeting information necessary to support weapons planning.[5,6] This is particularly true for dynamic targeting scenarios. As-is TCPED with traditional ISR SoSs does not provide the required timely coordination and reporting between operational processes and nodes/capabilities.[2,4,5]

Figures 2 and 3 show notional scenarios for maritime interdiction and anti-access/area-denial maritime dynamic targeting, which provide operational context with communications, threat detection tracking, classification, and other challenges for the netted ISR concept.

THE ISR CASE: NETTED ISR CONCEPT

The concept of a "to-be" TCPED-centric netted ISR SoS is an alternative to the present as-is ISR architecture. A TCPED-centric netted ISR SoS enables the warfighter to command a collaborative sensor fusion capability and optimally, reliably, and locally control actions and decisions for each node; it also increases the ability of the SoS accomplish missions and tasks. Data fusion (e.g., PED), previously completed in central locations, now occurs on sensor platforms. It uses organic and remote sensor data, increasing resiliency in degraded communications environments. Data fusion also occurs at data fusion centers specified by the commander. Constellation-wide sensor resource allocation (e.g., TC) occurs on both platforms and among operational commands designated by the commander to optimize data collection. The output of data fusion is fed back nearly instantaneously as input to sensor fusion. The netted ISR concept is enabled by end-to-end connectivity with quality of service that allows in-degree nodes of the SoS to assuredly share relevant information.

With netted ISR, command, sensor fusion, and data fusion capability is allocated to functional elements on segments that distribute across physical nodes in the SoS. Segments are a collection of functional elements. The expected role of each node determines the extent to which each segment is instantiated on the node. The designated role of each node for a specific operational mission determines the extent to which the instantiated segments are used on each node. As such, the netted ISR concept is extensible to platforms with varying space, weight, and power constraints; scalable to a very large number of connected nodes; and adaptable to a range of collaborative behaviors (e.g., nodes behaving as individual contributors, as neighbors, as a community, or with a swarm identity).

Functional elements of command include the ability to collaborate, translate the commander's intent into measures allocated to tasks and objectives, assess the performance of the SoS and infer achievement of the commander's intent, decide the priority of mission requirements to achieve the commander's intent, and assign mission requirements to ISR assets.

Functional elements of sensor fusion include the ability to infer sensor collection needs, decide and task sensor data collections and platform movement, skew sensors, move sensor platforms, collect sensor data, and share sensor data.

Functional elements of data fusion include the ability to receive and prepare sensor data for processing; screen sensor data; share relevant sensor data; perform processing to detect objects, track hostiles, precisely track hostiles, track all objects, classify targets, and determine intent of targets; develop object/target reports; share object/target reports; and maintain an ISR data repository.

Implementation of data fusion includes data conditioners, screening components specialized to multi-intelligence and multi-modality sensors, fusion components that perform data association, kinematics state estimation, class estimation, and data association. The data fusion approach exploits complementary attributes of diverse sensor phenomenology/geometries and data collected over time to maintain a low system-level, post-correlation false-alarm rate. Output conditioning prepares and disseminates fusion output (e.g., actionable information, target reports, and target nomination reports) at achievable data transmission rates.[7]

Implementing sensor fusion will include automated components that coordinate and synchronize ISR sensor platforms as an integrated unit to continuously maximize aggregate net fused information gain and adjudicate tasking among competing priorities across the entire search volume and all targets, as well as over a configurable finite planning time horizon.[8]

Implementing command of a netted ISR SoS will involve automation that shifts much knowledge and inference functionality performed by operators in the cognitive domain to processors in the logical domain. Collaborative cognition will occur between warfighters and the command capability of the netted ISR SoS. The netted ISR command functional element will provide the capability to reliably optimize responses to emergent behaviors associated with SoS complexity;[9] relate those responses to commander's intent, battlespace complexity, cognitive parameters, and decision policies established in doctrine; and synchronize the information, social, cognitive, and physical domains of battlespace management command and control.[10]

The concept of netted ISR and its collaborative command, sensor fusion, and data fusion capability aligns well with the *Navy Information Dominance Roadmap*, which states, "Battlespace Awareness will require enhanced information content, advanced means to rapidly sense, collect, process, analyze, evaluate and exploit intelligence regarding our adversaries and the operating environment."[11]

Netted ISR Reference Architecture

To facilitate SE of the netted ISR concept, a reference architecture was developed, as depicted in Fig. 4. The reference architecture is intended to guide and frame architecture and solution development. There are sensor, command, and fusion segments, which are instantiated on nodes that distribute across an SoS and interconnect by a heterogeneous network.

All node types have a multiprocessing hardware architecture. The processors can be subdivided into non-real-time and near-real-time types. Software that is event driven, such as user/mission applications and higher-level networking services (application, presentation, session,

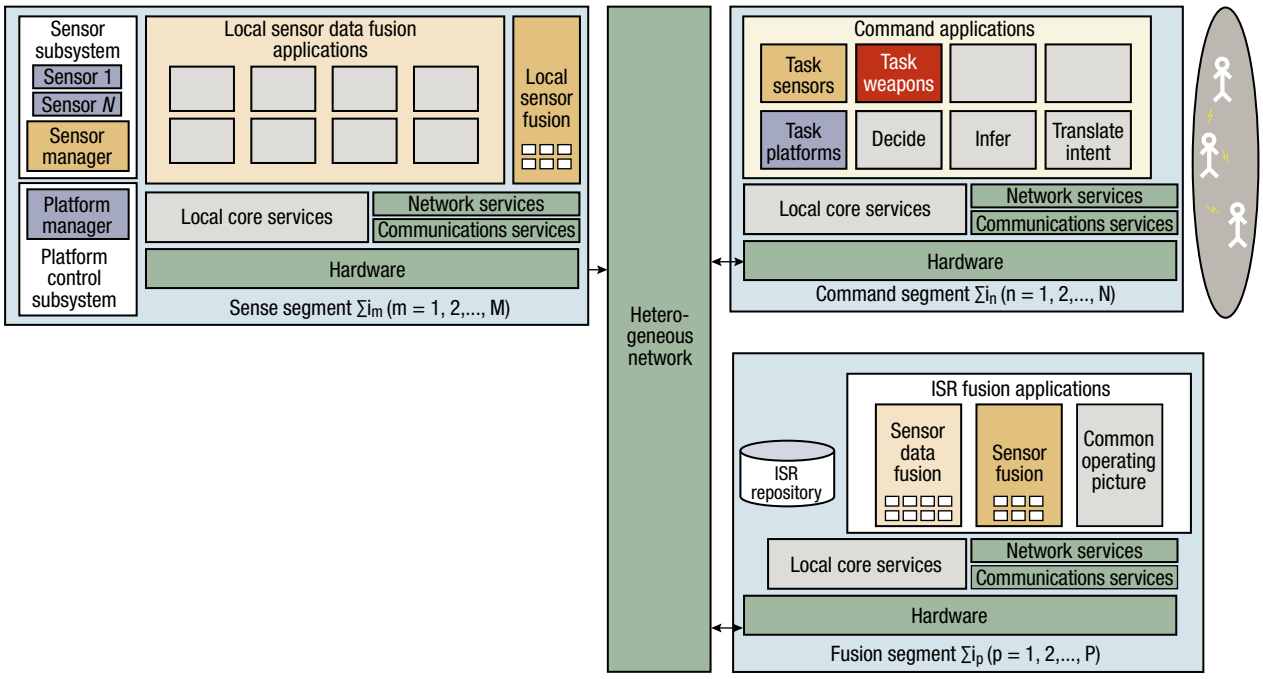


Figure 4. Netted ISR reference architecture.

transport, and network OSI [Open System Interconnection] layers), are hosted on the non-real-time processors, such as general-purpose processors and/or graphic processor units. The software architecture can be partitioned into two broad groups. Software that is near real time, such as lower-layer networking services (data and physical OSI layers), is hosted on near-real-time processors, such as digital signal processors or field-programmable gate arrays. The software architecture can be partitioned into user/mission applications and services such as local core, networking, and communication. The local core, networking, and communication services enable applications to execute the mission and interact with any node distributed in the SoS and across all segments of the SoS. User/mission applications are specific to each type of segment. For instance, sensor segments host sensor interfaces, whereas the command segment hosts user interfaces and fusion segments host data-repository interfaces. This reference architecture is modular and scalable, which facilitates rapid integration of innovative technology, legacy design reuse, and interoperability between diverse functional platforms and heterogeneous networks.

Case Study Approach: Netted ISR Model-Based Design for Affordability

With the ISR challenge defined, operational context provided, and initial netted ISR concept and reference architecture developed, the SE challenge is to validate that the netted ISR concept can realize the needed ISR capability more affordably and rapidly than the traditional ISR of today.

Resource constraints on the MBSE methodology development were a 4-month schedule and eight part-time subject-matter experts. The study team focused on design for affordability of the netted ISR concept and performed one innovation cycle using the MBSE methodology and SE infrastructure. The approach included modeling and analysis of ISR performance for an operationally relevant scenario as well as rapid development of an ISR network model, which enabled efficient exploration of both the traditional and netted ISR concepts. The network model was central to rapidly establishing ISR prototypes of both the traditional and netted concepts. Preliminary verification tests demonstrated performance of the ISR concepts. A cost model was developed and used to assess the affordability of both concepts.

Results of Netted ISR Case Study

Results from one innovation cycle of the concept engineering phase showed that the netted ISR concept directly improved the affordability and effectiveness of ISR. Preliminary verification tests for one scenario demonstrated that the netted ISR concept required 30% fewer sensor platforms and at least 25% fewer operators than the traditional ISR model. Netted ISR can reduce operational costs and reduce the number of assets needed to perform ISR. Cost modeling and affordability analysis showed significant, compounded life-cycle savings with the netted ISR concept compared to the traditional concept, especially as the battlespace becomes more complex.

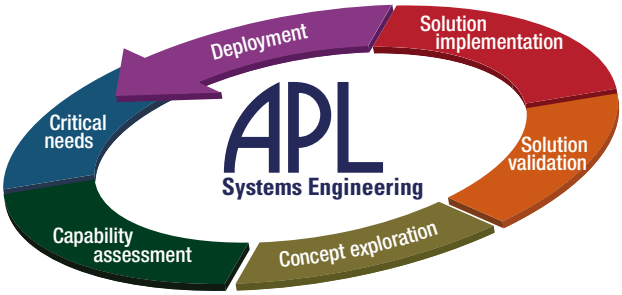


Figure 5. APL SE process. (Reprinted from Ref. 12.)

This case study made use of an MBSE methodology, IDE, and SE infrastructure to design for affordability as part of a 4-month concept engineering phase of a complex SoS. The remainder of this article describes the Johns Hopkins University Applied Physics Laboratory (APL) MBSE methodology, IDE, and SE infrastructure used for this case study and its relevance to supporting and influencing SE over the life cycle of a system.

APL SE Process

Implementation of the netted ISR architecture requires an effective balance of scientific and engineering principles, performance, requirements, and sponsor program constraints. This is a classical SE challenge. The APL SE loop,[12] which has matured over decades, is shown in Fig. 5, and the major phases used to solve national and international critical SE challenges are described in Table 1. Every step produces knowledge and experience, which is fed back into subsequent spirals. The APL SE process is mature and proven and has informed the development of the APL MBSE IDE.

Table 1. APL SE Process

Process Step	Description
Critical needs	Operational/mission data analysis is conducted, focused on establishing concept feasibility and exposing critical needs.
Capability assessment	Existing systems are evaluated to determine whether they can meet the need or whether a new capability development is required.
Concept exploration	Alternative candidate concept designs, models, and analyses are completed. Trade analysis enables effective down-selection to a best concept based on any number of metrics, such as performance, efficiency, economy, risk, utility, or a combination of these characteristics.
Validation	Proof-of-concept (PoC) prototypes are developed to verify functional performance in a representative environment.
Implementation	The concept is realized in an operational prototype, and verification tests are completed.
Deployment	The concept is deployed for field test validation.

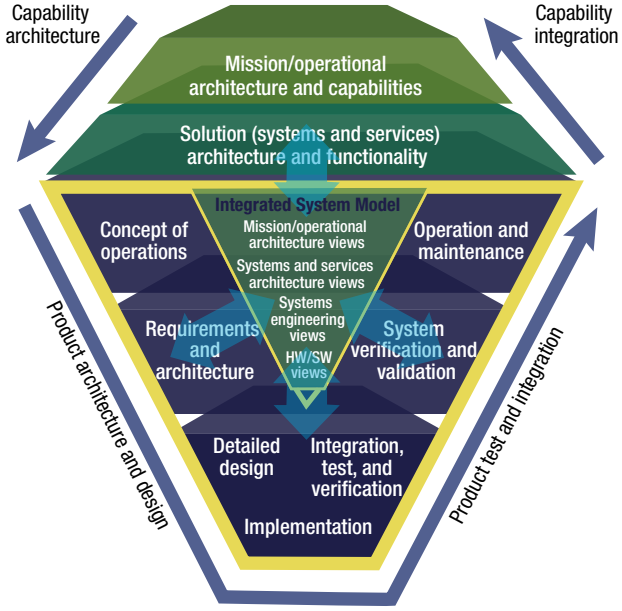


Figure 6. INCOSE MBSE methodology. HW/SW, hardware/software. (Reprinted from Ref. 13.)

INCOSE MBSE

MBSE provides a methodology to more effectively manage the SE challenge at lower cost and shorter development cycles. The International Council on Systems Engineering (INCOSE) has developed an MBSE methodology,[13] which is shown in Fig. 6. This diagram highlights how models are central to this methodology. The Integrated System Model is a repository for all knowledge about the system (aiding in communication and collaboration). Capability and product architectures, along with design, verification, and validation testing, are derived from the Integrated System Model. This results in consistency, which improves maintainability and reduces ambiguity. Requirements traceability

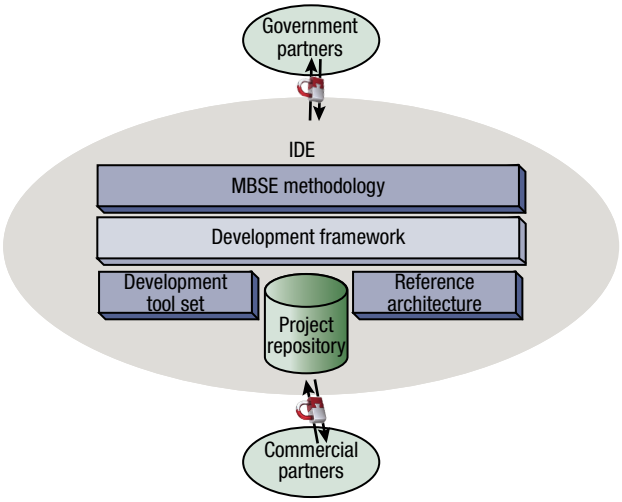


Figure 7. Integrated development environment.

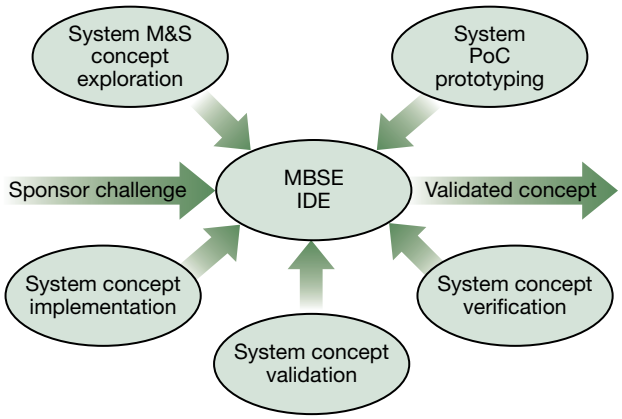


Figure 8. APL MBSE methodology. M&S, modeling and simulation.

to architecture and associated design elements enables change impact analysis. Documentation is generated from the Integrated System Model, ensuring accuracy. Automatic code-generation processes ensure efficient implementation and quality management. Because the INCOSE MBSE developments are embraced by a broad range of SE organizations, the INCOSE model has informed the APL MBSE IDE development.

MBSE IDE

APL is maturing an MBSE IDE that enables distributed stakeholders to collaboratively contribute solutions to the SE challenge. The APL SE IDE is initially provisioned and targeted for the concept exploration, implementation, validation, and deployment SE processes. It is relevant for application by projects such as the Defense Advanced Research Projects Agency-sponsored Air Dominance Initiative project,[14] whose objective was to integrate its Strategic Technologies Office capabilities for communications in a contested environment. The goals of the Air Dominance Initiative project included integration of legacy and future airborne communication waveforms through heterogeneous networking protocols across increasing numbers of pervasive airborne nodes while achieving drastically reduced innovation cycles. The APL IDE as shown in Fig. 7 facilitates distributed collaboration on shared centralized resources among government sponsors, users, and technical leadership with commercial hardware, software, and service providers. It is based on an integration of MBSE methodology, a distributed integration framework, hardware and software reference architectures, development toolsets, and a repository.

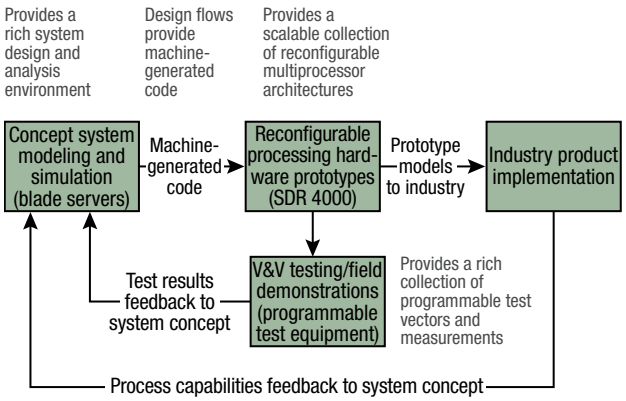


Figure 9. APL MBSE methodology flow.

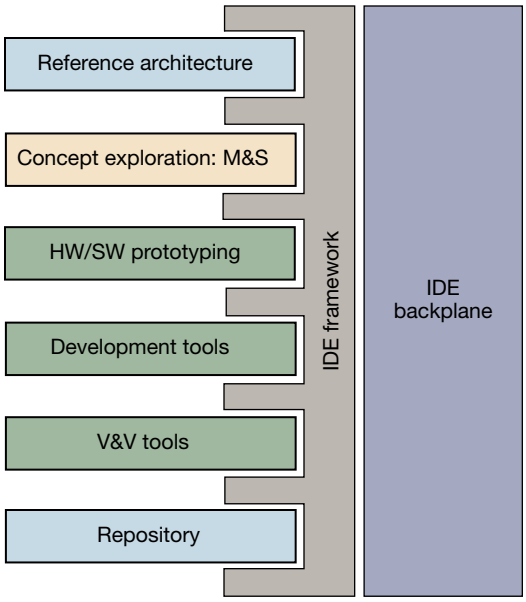


Figure 10. APL IDE framework. HW/SW, hardware/software.

AN APL-Branded MBSE Methodology

An APL-branded MBSE methodology (APL has developed a unique collection of processes, methods, and tools) is shown in Figs. 8 and 9. The major iterative and collaborative processes include (*i*) concept exploration through modeling and simulation, (*ii*) rapid prototyping through code generation, (*iii*) verification and validation (V&V) testing, and (*iv*) submission to the project repository. An inner closed loop enables rapid verification of concepts developed in the low-cost-of-change (relative

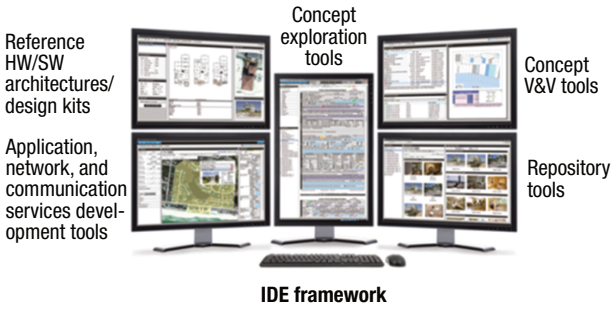


Figure 11. APL IDE notional user interface.

to cost of change in physical hardware and software) modeling and simulation environment and influenced by lessons learned from V&V testing. The outer closed loop enables stakeholders to influence the concept development through disciplined design for X (where X is a variable for affordability, six-sigma, testing, manufacturing, and so forth).

IDE Framework

The SE IDE framework shown in Fig. 10 illustrates how hardware and software reference architectures, development, V&V tools, and a repository can be integrated. This integration facilitates transfer of information, availability of resources, and maturation of concepts. Shown in Fig. 11 is a notional framework interface that enables distributed developers to navigate, access, and process using shared IDE resources for concept exploration, implementation, verification, and validation. Within the IDE, integrated tools are able to interoperate across a common backplane. Future tool versions will be provisioned with plug-and-play interfaces, common reference architecture templates, and repository libraries.

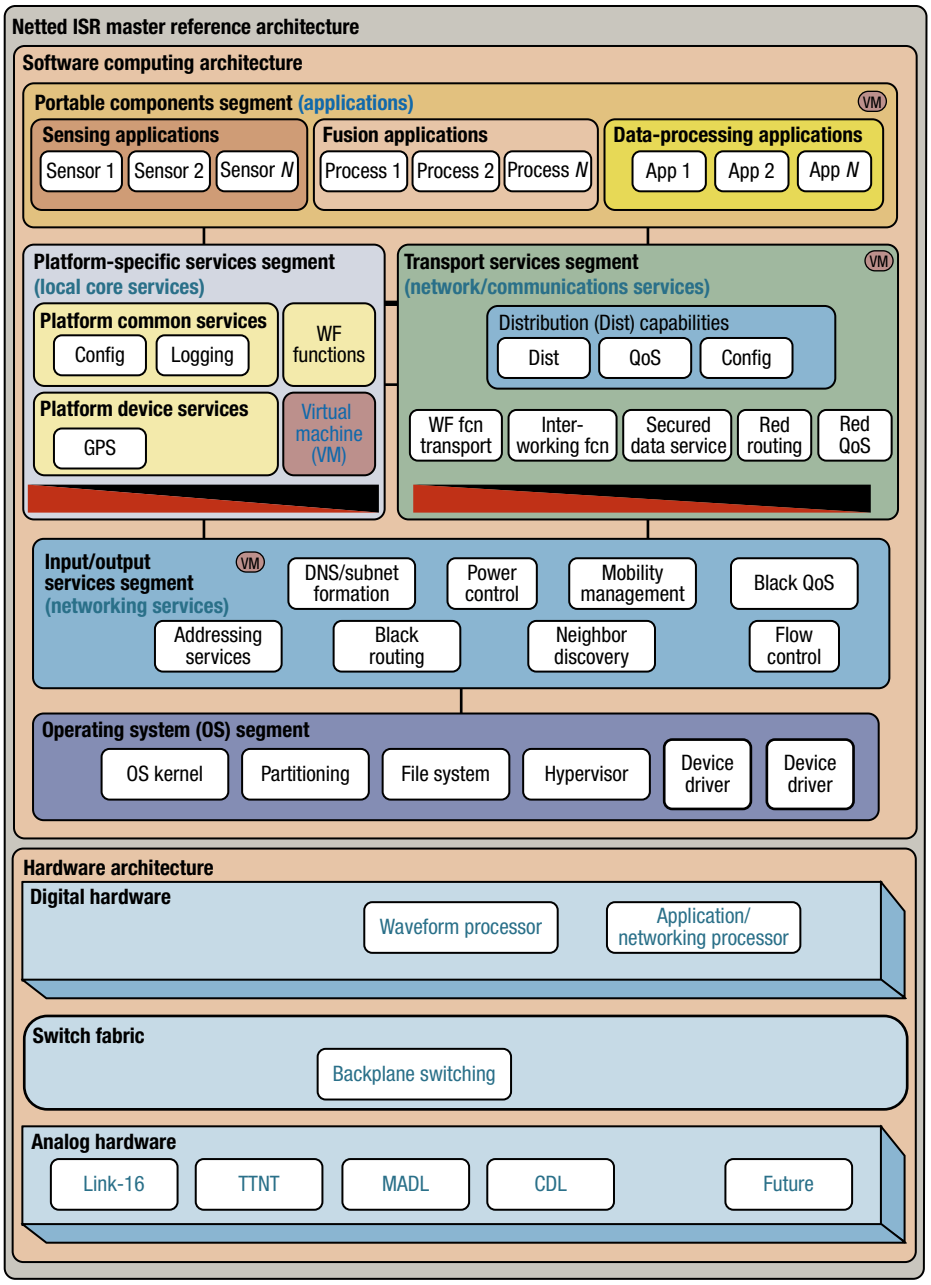


Figure 12. APL master reference architecture. CDL, common data link; Config, configuration; DNS, domain name service; fcn, function; MADL, multifunction advanced data link; QoS, quality of service; WF, web file.

Hardware and Software Reference Architecture

Within the SE IDE is a hardware and software master reference architecture, as shown in Fig. 12; it is based on the Technical Standard for Future Airborne Capability Environment (FACE)[15] and provides functional and interface definitions for the hardware and software to be implemented. It illustrates the major software application, networking, and communication services as well as the major non-real-time and near-real-time hardware processor types. Incremental implementation of this reference architecture is planned for several APL projects over the next few years.

Concept Exploration

The SE IDE includes modeling and simulation toolsets that enable concept exploration through PoC modeling and simulation. These concept virtual prototypes (VPs) enable exploration of a range of alternative approaches, analyzing the performance, costs, and benefits of each approach in an agile and low-cost-of-change environment.

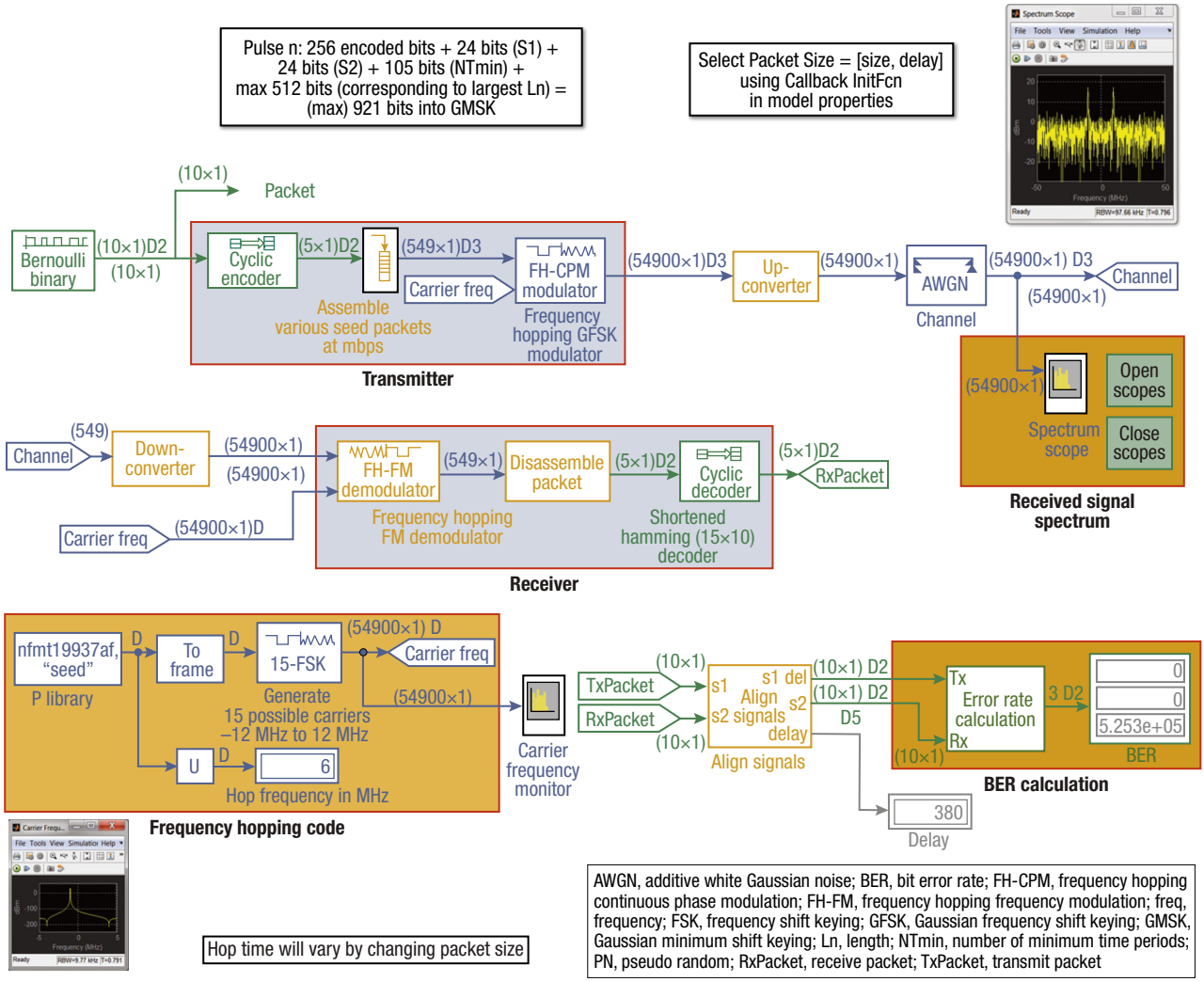


Figure 13. TTNT Simulink physical layer modulation virtual prototype.

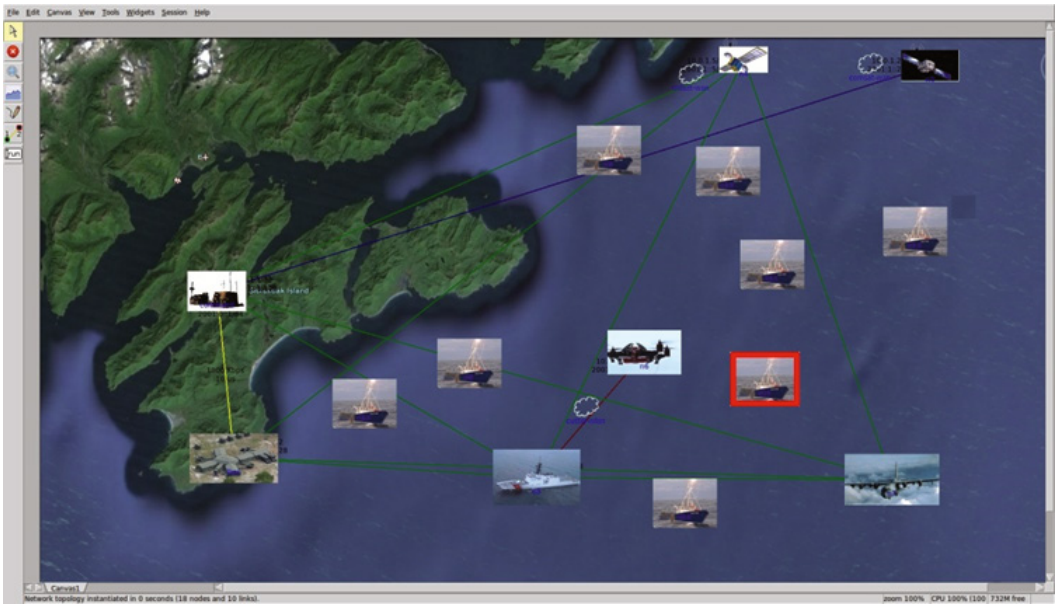


Figure 14. Netted ISR maritime interdiction scenario virtual prototype.

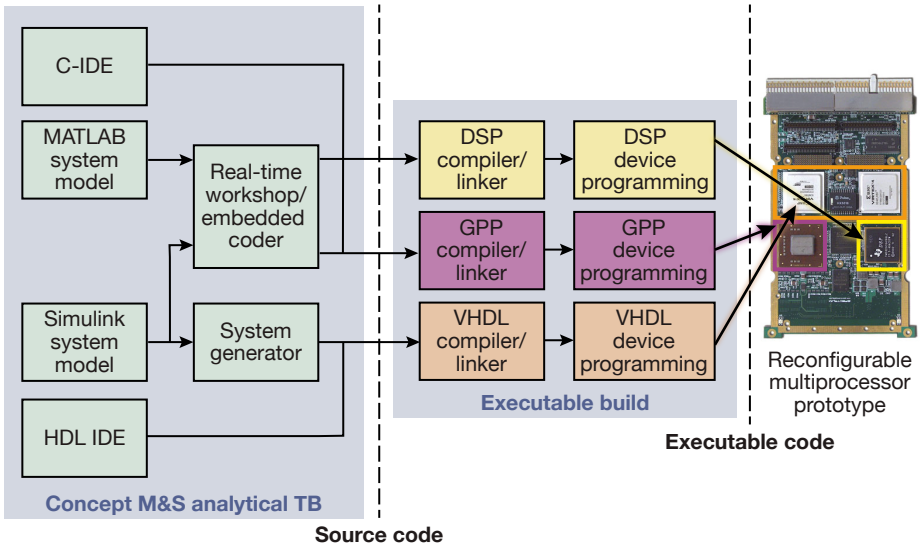


Figure 15. Rapid prototype auto-generation processing. DSP, digital signal processor; GPP, general-purpose processor; HDL, Hardware Description Language; M&S, modeling and simulation; TB, testbed; VHDL, Very High-Speed Integrated Circuit HDL.

Alternative down-selection and optimization highlight how rapid concept engineering is achieved.

As an example, consider a Tactical Targeting Network Technology (TTNT)[16] surrogate modulator model, as shown in Fig. 13. This simulation model (SM) is only a portion of the communication physical layer processing but highlights the capability to test system performance sensitivity under a range of modulation parametric variations.

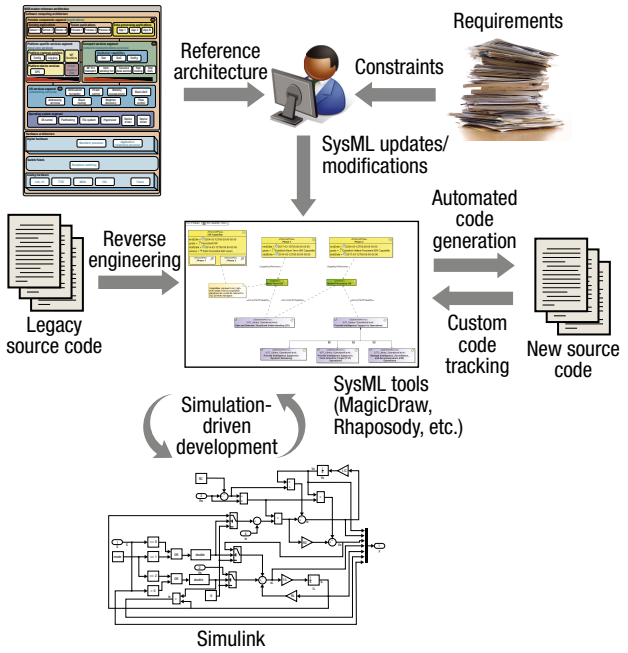


Figure 16. PhPs by automatic code generation.

As a second example, consider the netted ISR communication network SM for the maritime interdiction scenario shown in Fig. 14. This VP enables exploration of operational performance sensitivity to communication link parameters such as quality of service, throughput, range, power, and bandwidth. Each of these exemplar virtual prototypes enabled rapid exploration of critical performance parameters in a low-cost-of-change environment.

Concept Validation

The SE IDE also includes source-code-generation toolsets for rapid prototyping (i.e., converting system SMs developed during concept exploration into a PoC physical prototype [PhP] whose response to physical environmental processes can be measured with calibrated equipment). The calibration enables quantification of SM confidence intervals when measurements are compared to simulated responses. To enable rapid, low-cost-of-change prototypes, rapid software prototype tools are used, as shown in Fig. 15. These tools convert SM into source code, which can be compiled into executable code for target processors.

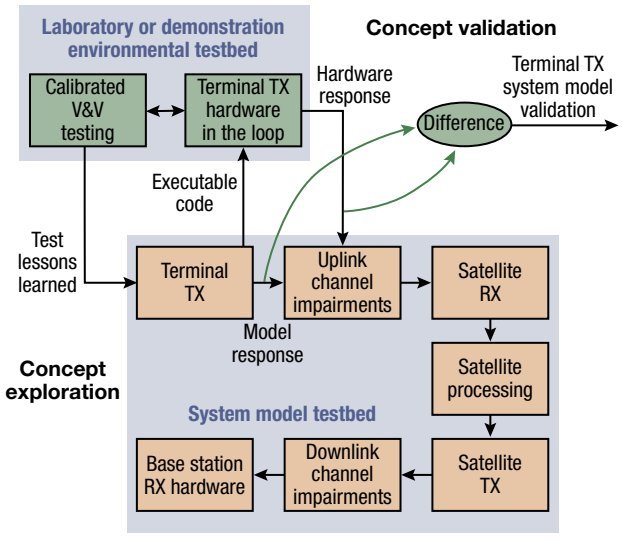


Figure 17. Hardware/software-in-the-loop (SiL) V&V TB. RX, receiver; TX, transmitter.

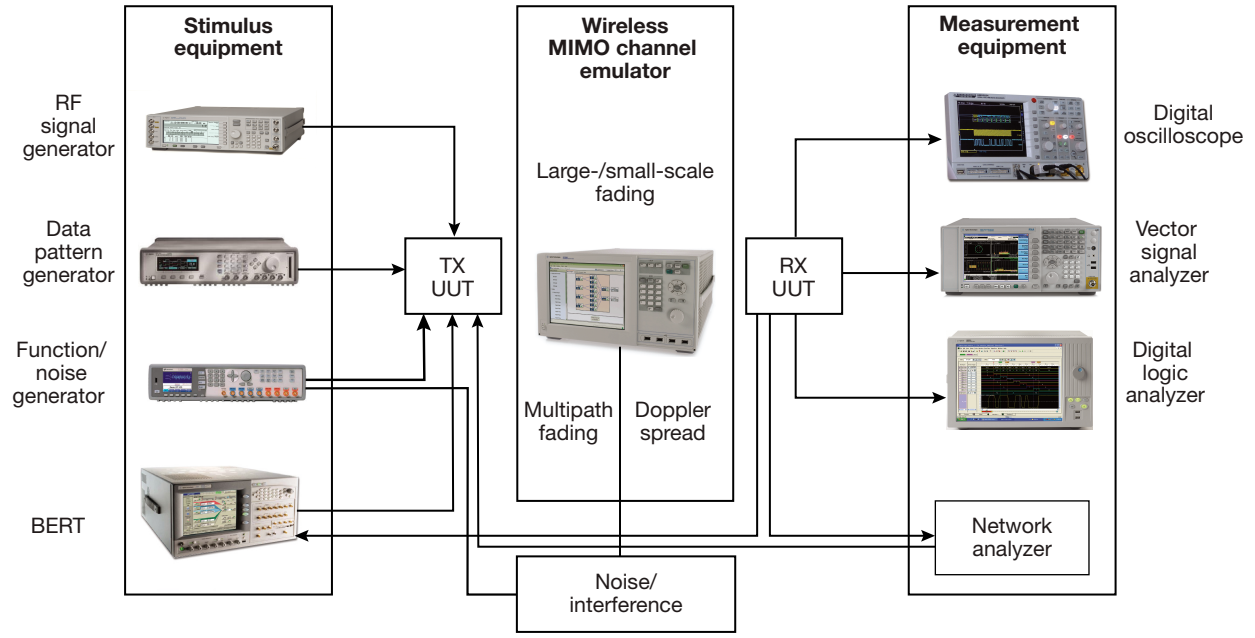


Figure 18. Communication V&V TB. BERT, bit error rate tester; MIMO, multiple-input and multiple-output; RX UUT, receive unit under test; TX UUT, transmit unit under test.

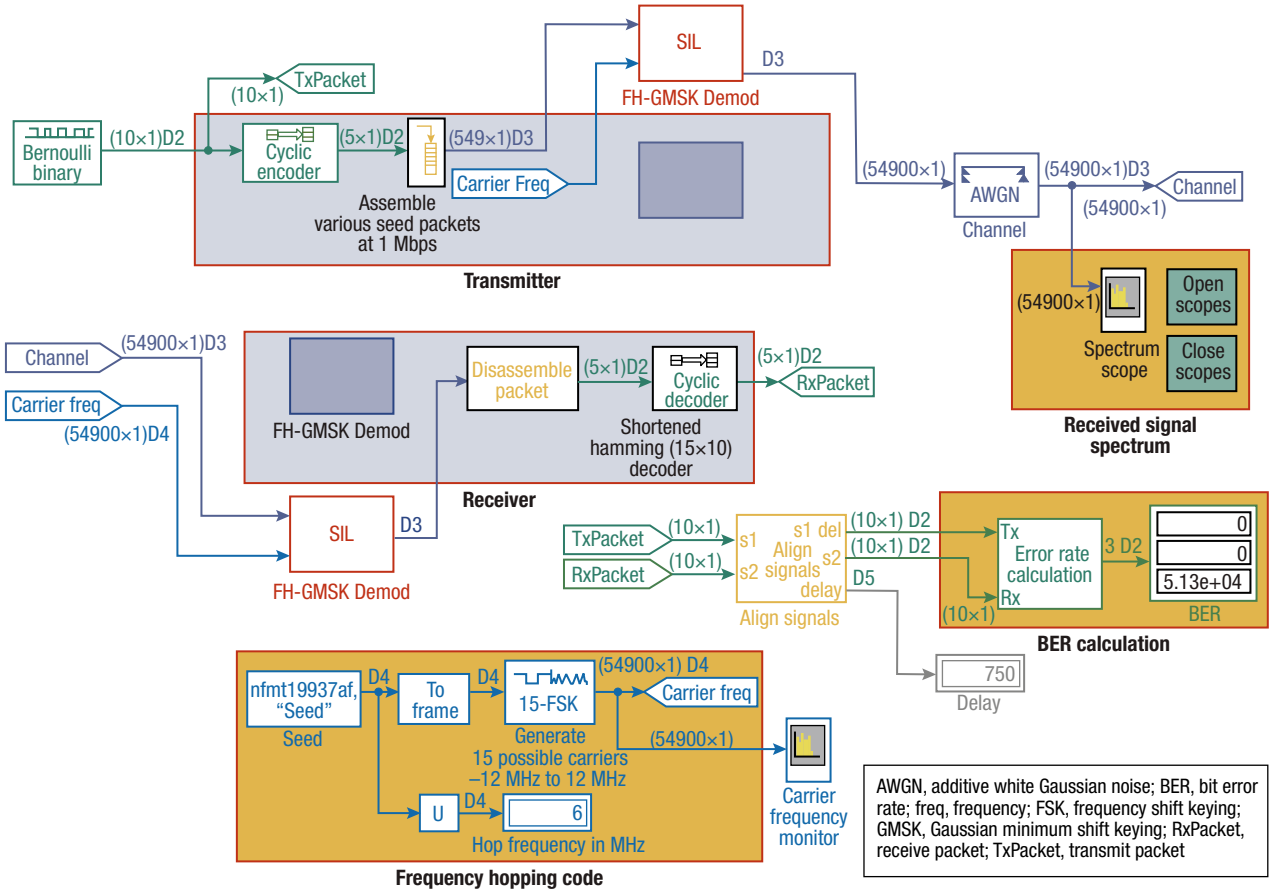


Figure 19. TTNT physical layer modulation SiL.

Reference Architecture SysML Rapid Prototyping

An MBSE rapid software prototyping example includes a processing hardware and software reference architecture such as FACE, performance requirements, and resource constraints. System designers can leverage legacy code to prototype new source code. As shown in Fig. 16, the SE IDE provisioned with SysML (Systems Modeling Language) and UML (Unified Modeling Language) toolsets (reference templates and libraries) is capable of leveraging legacy design, integrating new requirements within a reference architecture, and generating new source code. This example demonstrates how legacy code is reverse-engineered by a SysML[17] tool, such as MagicDraw (http://www.nomagic.com/products/magicdraw.htm) or Rhapsody (http://www-142.ibm.com/software/products/us/en/ratirhapfami/). The reverse engineering produces SysML diagrams of what the code represents and how they are interconnected and can be reused in the new system design. Based on the requirements and reference architecture, the system designer develops a model of the system in SysML and UML.[18] Throughout the system design process, Simulink (http://www.mathworks.com) code can be generated/developed to simulate target components. After the system has been modeled, code that represents the system is automatically generated. Custom code can be written for any components where automatic code generation is not possible. The SysML tool then tracks any custom code inserted or added to include in future automatic code generation. Automatic test cases are also generated from the requirements stored in SysML. The automatic code generation can also be done for a partial system model or for subsystems to test the system as the model is being developed to support an incremental or spiral development.

The PoC PhP can be stressed by mechanical, electrical, or environmental conditions in an integrated hardware/software-in-the-loop V&V testbed (TB), as shown in Fig. 17. These tests can be conducted within a laboratory or extended to include field-test demonstrations. For instance, the communication V&V TB shown in Fig. 18 can be integrated with the concept exploration environment by applying the same stimulus used in the SM, and responses from the V&V TB can be compared to the simulated responses in the SM.

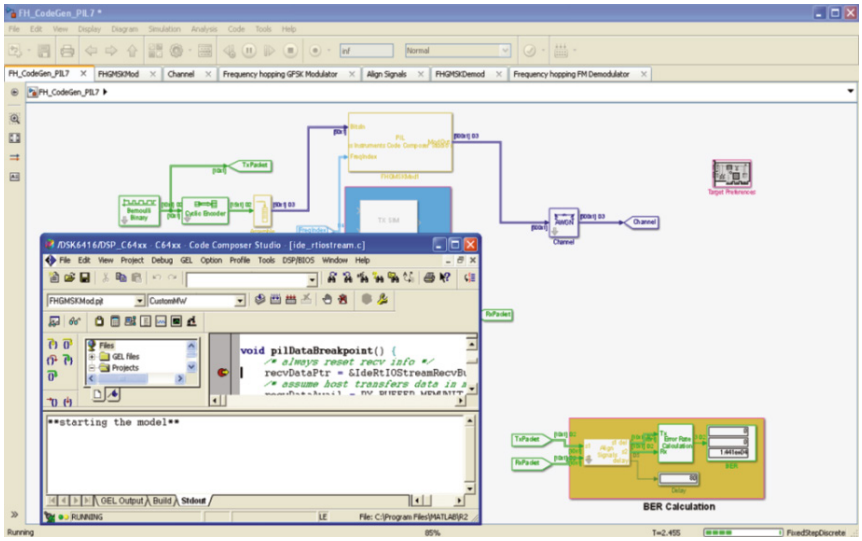


Figure 20. TTNT physical layer modulation PiL.

Consider the rapid prototyping example in Fig. 19; it is a software-in-the-loop (SiL) emulation, which is derived from the previous TTNT modulation SM. The SiL provides a capability to emulate the behavior of source code generated from the SM.

As a second example, consider the processor-in-the-loop (PiL) shown in Fig. 20; it is a digital signal processor hardware emulation derived from the same SM. It provides a capability to embed physical hardware running the SiL software within the SM. Implementation errors associated with fixed-point precision, timing, and clocking can be evaluated with the PiL. Both the SiL and

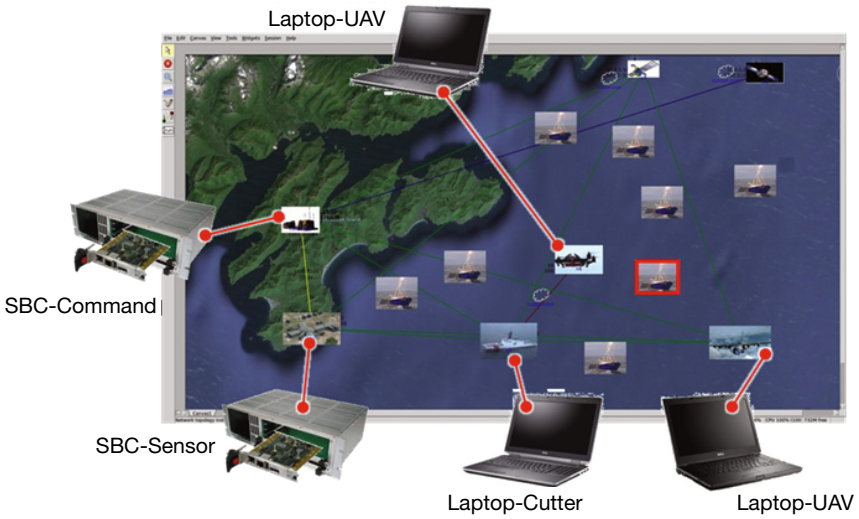


Figure 21. Netted ISR physical layer prototype. SBC, single-board computer.

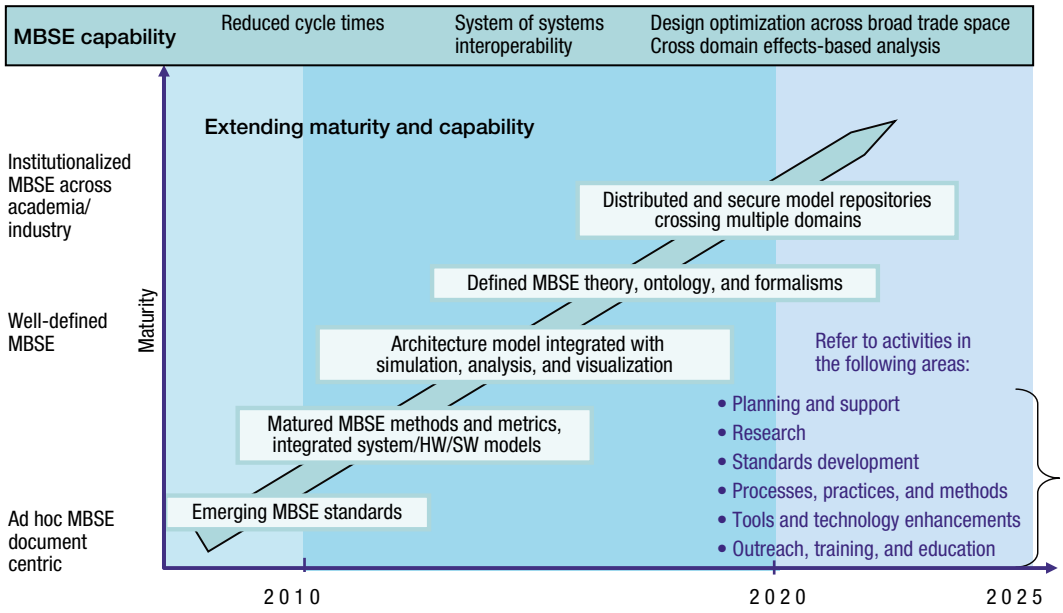


Figure 22. INCOSE MBSE methodology maturity. (Reprinted with permission from Ref. 19, © INCOSE.)

the PiL were auto-generated and compiled in minutes. Changes to the SM can be rapidly transferred to both emulations for verification testing.

As a third example, consider the netted ISR network PoC PhP shown in Fig. 21. In this case, physical hardware has been connected to nodes in the previously discussed VP. Operational hardware hosting operational software can execute in real time. Communication traffic is generated by the PhP, exchanged between the VP nodes, and processed by the destination PhP nodes. In this manner, the hardware-in-the-loop enables verification of node application, networking, and communication services while retaining a low-cost-of-change environment for the location, ranges, and wired and wireless link effects. Link latency, quality of service, and throughput can rapidly and easily be manipulated to quantify performance sensitivities of an operational scenario.

Integrated Validation Environment

The SE IDE PhP can be extended to field demonstrations (for example, consider the UAV). The netted ISR PoC PhP can be deployed on a constellation of airborne platforms and demonstrated under actual operational conditions to validate the concept. A similar test stimulus used in the SM can be applied to the field demonstration hardware, and responses can be measured and compared to the simulated responses in the SM.

CONCLUSION

The APL MBSE IDE has been introduced and applied to the anti-access/area-denial netted ISR case and a growing list of other SE projects. The IDE infrastructure facilitates effective, agile, rapid, and affordable concept exploration, verification, and validation for complex SoS such as netted ISR. For the netted ISR case, a rapid innovation cycle project over the course of 4 months used an IDE to define, model, prototype, and produce initial results that indicated that 30% fewer sensor platforms and at least 25% fewer operators were necessary for one scenario, contributing to the potential for significant, compounded life-cycle savings with netted compared to traditional ISR SoS.

Studies conducted by the National Defense Industrial Association, INCOSE, and other SE organizations recommend adoption of MBSE methodologies based on member-documented project cost and schedule reductions.[19–21] The INCOSE MBSE roadmap is shown in Fig. 22. APL is midway through maturing its brand of MBSE methodology.

REFERENCES

[1]Shaffer, A., *Current S&T Priorities and the Future of DoD S&T*, Department of Defense, Washington, DC (29 Oct 2013), http://www.acq.osd.mil/chieftechnologist/publications/docs/Current_ST_priorities_and_the_Future_of_DOD_ST.pdf.
[2]Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, *Report of the Joint Defense Science Board Intelligence Science Board Task Force on Integrating Sensor-Collected Intelligence*, Defense Science Board, Washington, DC (Nov 2008).
[3]Dunaway, D., "Creating Integrated Warfighting Capabilities," *Proceedings Magazine* 139/8/1,326 (2103).
[4]*Joint Operational Access Concept (JOAC)*, version 1.0, Joint Chiefs of Staff, Washington, DC (Jan 2012).
[5]United States Joint Forces Command, *Commander's Handbook for Persistent Surveillance*, version 1.0, Joint Warfighting Center, Joint Doctrine Support Division, Suffolk, VA (20 Jun 2011).
[6]Joint Publication 3-60 (JP 3-60), *Joint Targeting*, Joint Chiefs of Staff, Washington, DC (31 Jan 2013).
[7]Newman, A. J., and Mitzel, G. E., "Upstream Data Fusion: History, Technical Overview, and Applications to Critical Challenges," *Johns Hopkins APL Tech. Dig.* **31**(3), 215–233 (2013).

[8]Newman, A. J., and DeSena, J., "Closed-Loop Collaborative Intelligence, Surveillance, and Reconnaissance Resource Management," *Johns Hopkins APL Tech. Dig.* **31**(3), 183–214 (2013).
[9]Connor, T., and Wong, H., "Emergent Properties," *The Stanford Encyclopedia of Philosophy*, Spring 2012 Ed., E. N. Zalta (ed.), last modified February 12, 2012, http://plato.stanford.edu/entries/properties-emergent/.
[10]Alberts, D. S., and Hayes, R. E., *Understanding Command and Control*, Command and Control Research Program Publication Series, Washington, DC (2006).
[11]*U.S. Navy Information Dominance Roadmap, 2013–2028*, U.S. Navy, Washington, DC (Mar 2013).
[12]Seymour, J. S., and O'Driscoll M. J., "APL Applied Systems Engineering: Guest Editors' Introduction," *Johns Hopkins APL Tech. Dig.* **29**(4), 306–309 (2011).
[13]Valinoto, T., "Application of MBSE Theory in a World of Practical Deadlines and Deliverables: Lessons Learned," in *Proc. MBSE Symp.: INCOSE Chesapeake Chapter and JHU/APL* (29 Mar 2014).
[14]Defense Advanced Research Projects Agency (DARPA), DARPA-BAA-14-02, "Communications in Contested Environments (C2E)," Solicitation Number DARPA-BAA-14-02 (Jan 2014).
[15]Open Group, *Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.0* (Feb 2013), http://www.opengroup.org/face/tech-standard-2.
[16]U.S. Department of Defense, "Waveform Specification JTRS Software Waveform," *Tactical Targeting Networking Technology (TTNT)*, Draft Revision 4.0 (Jul 2005).
[17]Object Management Group, "Documents Associated With SysML Version 1.2," http://www.omg.org/spec/SysML/1.2/.
[18]Object Management Group, "Index of /spec/UML/2.3/Superstructure," http://www.omg.org/spec/UML/2.3/Superstructure/.
[19]Friedenthal, S., Griego, R., and Sampson, M., "INCOSE Model Based Systems Engineering (MBSE) Initiative," in *Proc. INCOSE 2007 Symp.*, San Diego, CA, p. 17 (2007), https://www.incose.org/enchantment/docs/07Docs/07Jul_4MBSEroadmap.pdf.
[20]Hause, M., Stuart, A., Richards, D., and Holt, J. "Testing Safety Critical Systems with SysML/UML," in *Proc. 15th IEEE Intl. Conf. on Engineering of Complex Computer Systems*, Oxford, UK, pp. 325–330 (2010).
[21]National Defense Industrial Association (NDIA) Systems Engineering Division M&S Committee, "Final Report of the Model Based Engineering (MBE) Subcommittee," NDIA, Arlington, VA (10 Feb 2011).

THE AUTHORS

K. Dewayne Brown is a communications systems engineer and Supervisor of the Resilient Tactical Communications Section in APL's Asymmetric Operations Sector. He provided MBSE expertise for the ADI C2E and netted ISR projects. **Mary Beth A. Chipkevich** is a Principal Professional Staff member in the Force Projection Sector and Program Manager for Unmanned Systems and Autonomous Technologies in the Precision Strike Mission Area. **Robert J. Bamberger** is a member of APL's Principal Professional Staff and Supervisor of the Autonomy Section in the Research and Exploratory Development Department. He was the Technical Lead for the Defense Advanced Research Projects Agency's Airborne Dominance Initiative (ADI) Communications in a Contested Environment (C2E) project. **Tam-Thanh C. Huang** is a member of APL's Senior Professional Staff and is a communications engineer with extensive experience in software-defined radio design and implementation of various military waveforms. **Mark A. Matties** is a member of APL's Senior Professional Staff in the Communication and Networking Systems Group and Supervisor of the Advanced Software Defined Networking Section. He currently researches the security and advanced applications of software-defined networking to provide secure, resilient networks. **James D. Reeves** is a member of APL's Senior Professional Staff and is a systems, network, and information systems security engineer for advanced technologies in the Aerospace Systems Analysis Group of the Precision Strike Mission Area. He was the Lead Systems Engineer for the netted ISR project. **Christopher A. Rouff** is a member of APL's Senior Professional Staff in the Communication and Networking Systems Group. He is a computer scientist doing research in automatic code generation from system models to quickly produce simulations and prototypes of new system concepts. The ADI and netted ISR teams can be contacted through the Program Manager, Mary Beth Chipkevich. Her e-mail address is mary.beth.chipkevich@jhuapl.edu.