Model-Based Engineering for Supply Chain Risk Management

Dan Shoemaker, Ph.D.
University of Detroit Mercy

Carol Woody, Ph.D.
Carnegie Mellon University
Software Engineering Institute

***Abstract*—Expanded use of commercial components has increased the complexity of system assurance verification. Model-based engineering (MBE) offers a means to design, develop, analyze, and maintain a complex system architecture. Architecture Analysis & Design Language (AADL), which has tools for modeling and compliance verification, provides an effective capability to model and describe all component units in a sourced product and implement practical measures for their management and assurance throughout the acquisition life cycle.**

Supply Chain Context

The manner in which we develop complex software and systems has changed considerably since the dawn of the Internet age [1, 2]. Much of that change has occurred because software has become big business. The most recent estimate of the size of the industry is $407.3 billion per annum, with a 4.8% annual growth [3]. As a result of the vast increase in the scope of the marketplace, commercial components are ubiquitous in our infrastructure [4]. This is especially true in government where the Clinger-Cohen Act has directed federal agencies to maximize their use of commercial-off-the-shelf (COTS) products [5].

COTS applications, especially those that are developed for commercial purposes, are often vulnerable to exploitation [5]. This is especially true when the unique risks faced by government systems are factored in [5]. For instance, the conventional approach in most organizations is to look for a COTS product[1] to solve some functional need. That choice is understandable, due to the cost advantage and availability that COTS components represent [3]. However, implementing these products can represent a serious security challenge since we rarely have the option to fully understand what we are buying [6].

[1] Many of the same challenges apply to open source

More importantly COTS products are typically integrated up a sourced supply chain, which creates a problem of security assurance and control at every level. It is a well-documented fact that we have lost all visibility into what is going on at the bottom of that generic chain [7]. So we are left with "trust" as the only viable option for establishing assurance. But what evidence can we use to justify the trust? Since market forces favor functionality over security and reliability, the challenges of addressing this supply chain problem are increasing [6].

It would be valuable to have a formal, well-defined, standard and systematic means for evaluating the assurance of systems which may contain security vulnerabilities inserted through insecure information and communications technology (ICT) supply chains. That formal solution is a practical necessity if we ever want to be assured that our adversaries cannot, "destroy power grids, water and sanitary services, induce mass flooding, release toxic/radioactive materials, or bankrupt any business by inserting malicious objects into the (ICT) components that comprise our infrastructure"[8].

<u>The Problem: Why ICT Supply Chain Security is so Easy to Compromise</u>

Because of the potentially critical impact of insecure supply chains on the U.S. infrastructure, the General Accounting Office has placed ICT Supply Chain Risk Management on its annual "Key Issues, High Risk" list [9]. A primary contributor to security vulnerabilities is the standard approach used for integration [5]. We no longer build single purpose systems from the ground up using a conventional design-build-test structure of creating well-defined components assembled into a predicted whole. Instead, we integrate a system from existing and reusable components in a hierarchy that extends up from the modular level through an increasingly sophisticated larger collection of integrated modules.

Due to the cost advantage the components that we integrate are obtained through a world-wide ICT supply chain that favors low cost [4]. In effect our products can be composed of software artifacts from India, chips and programmed logic from Korea and small components from Vietnam and China [4, 7]. These components were not designed and built to work together smoothly and effectively, instead they are cobbled together using standardized interfaces for information passing. These components were not built to only perform the selected activities needed for a specific system and may provide a wide range of additional functionality that supports unintended consequences.

The organization implementing this assembled collection of international components needs to ensure consistency in performing all aspects of supply chain risk management by providing a uniform, disciplined repeatable assessment that establishes an appropriate level of trust. One source of evidence would be a means of confirming that the practices involved in developing each level of the final product were consistently executed using a uniform standard management process starting from the basic components through to the final assemble within a planned and documented environment. Another source of evidence would be an assessment of "what could go wrong in the construction process (i.e., assessing risks), determining which risks to address (i.e., setting mitigation priorities), implementing actions to address high-priority risks and bringing those risks within tolerance must be well defined and uniform" [10].

Correct application of these control and assurance activities requires a detailed understanding of how the product will be built and the ability to monitor the construction process to assure security and correctness throughout. That is easier said than done since the basic component elements for a system are likely to be complex code segments that are sourced outside of the direct control of the system manufacturer, who is actually an integrator of a range of components from many sources.

Problem Statement: The Weakest Link

Typically, supply chains are hierarchical, with the primary supplier forming the root of a number of levels of parent-child relationships. From an assurance standpoint, what this requires is that every individual product of each individual node within that hierarchy be secure as well as correctly integrated with all other components up and down the production ladder. [7].

In the world of software, the locus of assurance is typically in the manufacturing process rather than the product itself. That is because the actual product is both too complex, as well as too virtual to be able to see and control. Ensuring a complex, distributed process like a supply chain would require a coordinated set of standard, consistently executed activities to enforce the requisite level of visibility and control. Yet, because the development process is usually occurring in a number of disconnected global locations, typically at the same time, the requisite level of understanding needed to assure the security and correctness of the component is hard to achieve [10].

For effective supply chain risk management, system engineering staff need to be able to evaluate the likelihood and impact of a potential vulnerability appearing at any stage in the product development. This evaluation mechanism is needed in order to identify appropriate mitigations. Best practice suggests that this identification and evaluation should take place during the early acquisition stages, because the form of the final product is being defined at that point [4, 5].   This challenge for the identification of potential security vulnerabilities extends the typical system engineering security considerations beyond the borders of the acquirer-supplier relationship. In that respect the correctness and security of the supply chain that comprises the development environment also impacts security [11]. To address complex supply chain risks, systems engineers need to be able to analyze the potential for the components to be exploited or subverted at any level of the construction process, and then consider the potential actions that need to be taken in order to detect exploitation and devise the necessary mitigations to continue to operate [11].

Systems engineers would want to have the ability to identify potential vulnerabilities early in the system-acquisition process. This would require that systems engineers be able to fully dictate the system concepts and critical functions and access paths of the entire product as it is being built. However, the typical approach in use today is to simply test for known common vulnerabilities of the system, supply chain, and development environment when all the pieces are assembled at integration. These tests are drawn from industry databases [1, 11], or in the case of the Federal Government from the Defense Acquisition Guidebook [12]. This is insufficient to evaluate and assign trust to the end product.   It is desirable, in some manner, to assure each component at the bottom of the supply chain process and then every successive integration to the final product. While we may not have the mechanisms to fully address this need, we can greatly improve current practice.

Utilizing Formal Structured Design Approaches to Establish Assurance Control

The identification and detailed understanding of all of the outsourced units in a supply chain can be an impossible task where the product might be composed of 10,000 individual components at the 4th or 5th level down in the integration process.

The logical way to draw up such a schematic is through a formal system modelling process, which will uniquely identify each component at all levels in the hierarchy. Such a modeling approach, if properly supported by a formal design language and kept under configuration management control, could create a very precisely detailed representation of the functional and security requirements for all components. And that schematic model would allow for the rational imposition of structured assurance activities at all levels in the integration process.

The acquirer must establish specific properties that the integrators must meet in order to satisfy the acquisition contract. If the system engineers for the acquirer assemble the first level decomposition of the system into AADL and incorporate the required properties that each component must contribute to the system into the model, the planned composition can be formally verified using available tools. The developer of each component must deliver an AADL model that establishes the properties of their component as built along with the actual product. In addition to executing acceptance testing of the component, the acquiring engineer can import this decomposition into the original model and confirm that the expected properties are still met. The result, from an assurance control standpoint, is the assembly of a complete and explicitly detailed set of design schematics that can be used to guide the process of monitoring the award and assurance of the outsourced work.

Safety-critical verification of cyber-physical systems (CPS) has benefited from the use of architecture fault modeling capabilities provided by Architecture Analysis & Design Language (AADL). Architecture led hazard analysis using architecture description languages (ADLs) such as AADL has become an effective capability in safety fault management. The cost of successfully addressing safety compliance has been greatly reduced through the use of extensions to AADL that automate safety analysis and produce safety assessment reports to meet recommended practice standards (such as SAE ARP4761) [16]. AADL is a relatively well known architectural description language that was first developed as the Avionics Architecture Description Language [14]. It was eventually standardized by the Society of Automotive Engineers as AADL [14]. The Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) has used AADL to effectively address design verification for the qualities of safety, reliability, and performance [18, 19]. Other researchers have incorporated selected attack scenarios into modeling languages (e.g., OCL [20], OWL-DL and SWRL [21], and SysML [22]). Avionics vendors have successfully used formal modeling to ensure safety properties from components developed by multiple vendors.

Acquirers can leverage these successful capabilities to address security as well as safety in supply chain risk management. The AADL modeling process ensures all three of the requisite criteria for successful management of a system product under development. These are, understanding, through modeling, assurance through formal validation and verification and finally accurate control of the integration of components, through reference to well-defined baseline expected behaviors.

Understanding: it is essential to have a detailed description of the structural elements of the system architecture in order to enforce assurance. The major problem with ensuring trustworthy product security at the component level using standard systems engineering approaches is the inability to precisely document the behavior of the specific contents of secured products along with the exact process by which they are built. Consequently, it would seem impossible to verify that the design requirements and tenets of best practice have been satisfied for each unit, especially if the system code has already been written, which is the case with commercial products. However, expected operational properties can be described and evaluated across the composition. The behaviors for each component can be formally described using a modeling language that captures the detail evidence. AADL is supported by tools that can capture properties about each component as well as the selected integration mechanisms for each component into the composition. Assumptions can be checked for consistency across all components to meet critical system behaviors.

Assurance: The necessary level of visibility for assurance requires knowing what the code that comprises each component is supposed to do and how that will affect the assembled whole. The confirmation of correctness of a complex system is normally presented in the form of an assurance case [11]. An assurance case is a well-thought-out argument, which is often represented notionally. An assurance case proposes and supports whatever claims are made about a given set of planned behaviors for a system [11]. An assurance case can be structured as a proof of trustworthiness based on evidence assembled through a formal model.

The supporting evidence for an assurance case may combine different kinds of documentation and data to justify its generic claim [11]. So a structured model of the system architecture, down to the unit level, is needed to let the engineer confirm the correctness of each component in the system [11]. In that respect, the assurance claim starts with the development of a general statement about a system goal such as an operational timing requirement. It is then decomposed down to a detailed, structured system model, which provides the concrete description needed to evaluate the detailed system architecture and its security properties at all levels in the development process as well as up and down the supply chain [11]. Claims for supply chain assurance can also be decomposed into an assurance case that assigns various desired (and necessarily avoided) characteristics to each component.

Control: In most approaches for systems engineering, modeling and validating of the attributes of a system are typically done using existing threat modeling and analysis technologies which are separate from the actual components and known characteristics of the system. The focus is on the creation of threat models to inform requirements and development decisions instead of formally evaluating the system composition to see how well it addresses the needed assurance. This provides no mechanisms for verification of delivered results to determine if threat concerns have been appropriately addressed. Moreover, these models are typically not maintained or updated throughout the life cycle, making it difficult to predict the impact of downstream change on attributes that cut across unit boundaries within the system [13]. AADL provides a means to model expected system behaviors that address threat concerns and formally verify consistency of these behaviors in modelled components that are outsourced.

The impact of component decomposition choices, which can include COTS, open source, and other outsourced subcontractors, need to become the responsibility of the integrator assigned to deliver the component to prove that desired system properties are met. The model of their resulting product must include downstream alterations made during the coding and even testing stages that are likely to produce discrepancies between the original design and its assurance case [13]. If the lower level components can embody undesired and unanticipated emergent behavior that can propagate up the ladder as the system evolves to higher levels of integration [11] this can be identified if the product behavior is appropriately modelled. Modelled properties must be applied to both component development and their interconnections.

AADL implements a basic language construct that centers on use of a standardized notation. The existence of a single common notation makes AADL relatively easy to automate since it underwrites a single standard view of all aspects of the system [14]. The notation language allows the engineer to specify system-specific characteristics by describing the unique set of user properties as well as any behavior involving state machines. It also allows the engineer to specify all of the associated error and dissemination concerns along with the specific data constraints [15].

The value of the AADL modeling scheme lies in the ability to isolate the basic components of a system to the right level of abstraction. This is called kernelling [4]. Kernelling allows for specific understanding and verification of the correctness of all components at a specified level of the design. The ability to describe discrete units within security levels in the architecture enables the practical work of validation and other forms of black-box testing. It is possible to consider each unit as an individual component of the proper functional level. Interactions of components both within that level and with objects at a higher level can be understood for the purposes of targeted testing of each individual component, or the testing of an integrated set of functions at whatever level designated.

AADL provides tools that capture the assurance case so that it can be verified against the formal design to identify inconsistencies. By keeping both the model and the assurance case current throughout the development life cycle and across the supply chain, this consistency checking can be used when COTS components are replaced with updated versions or other components, when new components are added that change the overall composition, and when segments of a system are modernized to meet changing business needs.

AADL is not a simplistic solution to supply chain risk management. To take advantage of this modeling capability, the system and its expected properties must be clearly established within the model prior to outsourcing. The assurance case must be assembled and verified against the system model to ensure the desired properties are consistently applied to the components. Security concerns expressed in a threat model must be analyzed and design mitigations characterized as expected properties within the model. Trust boundaries between components must be clearly described with properties that formalize allowed behaviors. Current SEI research is exploring the range of security behaviors that can be modelled to establish a level of effectiveness and the potential for code generation capabilities to carry formally defined behaviors directly into the resulting product.

Conclusion: Application of AADL to Assurance in the Supply Chain

The use of AADL to provide a model-based engineering approach (MBE) offers a better way to design, develop, analyze, and maintain a system architecture that is supported by a supply chain. AADL, provides an effective capability to model and describe all component units in a sourced product and implement practical measures for their management and assurance throughout the acquisition life cycle [14, 15]. Modeling enhances the engineer's ability to identify and address potential design weaknesses, an important category of security problems as noted in the common weakness enumerations [23].

Through the application of MBE system engineers, architects and product developers can reduce risk by performing early and repeated analysis of the system architecture [14] as outsourced components are delivered. This level of control and assurance can reduce cost by ensuring fewer system integration problems as the product moves up the supply chain.

A formal model can also simplify and ensure more effective evaluations of the organization-wide impacts of architectural choices and, by increasing understanding, make for simpler life-cycle support [14]. MBE can also increase confidence in the security of the implemented product because the assumptions that are captured in the modeling can be shown to have been validated as the product moves up to implementation in the operational system [14].

Formal modeling of all system components at all levels of decomposition gives the engineer the opportunity to define the detailed security characteristics of a delivered sourced product. Thus the MBE approach based on AADL design descriptions should provide the potential for use in the overall verification and validation work through confirmation of an assurance case to confirm the correctness of all components in a product [14, 15]. AADL modeling can be used to evaluate the correctness of components as well as the integration of components provided through the supply chain from multiple suppliers [13].

AADL has been successfully utilized to model both software and hardware applications in industries where the need for safety and reliability is paramount [14, 15]. SEI research is working to expand the security analysis capabilities of AADL [14, 15] to incorporate the design characteristics and constraints needed in formal modeling to anticipate and avoid two critical security design challenges: acceptance of tainted input and allowing inappropriate elevation of privileges. AADL offers opportunities for improvement in supply chain risk management.

Acknowledgment
Copyright 2015 Carnegie Mellon University.
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.
Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

References

[1] MacDonald, Neil; Valdes, Ray Gartner Says IT Supply Chain Integrity Will Be Identified as a Top Three Security-Related Concern by Global 2000 IT Leaders by 2017. Gartner 2012

[2] Mitre Corporation. 2012. Common Weakness Enumeration: A Community-Developed Dictionary of Software Weakness Types. http://cwe.mitre.org/.

[3] Pettey, Christy, "Gartner Says Worldwide Software Market Grew 4.8 Percent in 2013" Gartner, Stamford, Connecticut, March 31, 2014

[4] Reifer, Donald J. "Malicious Code in COTS: A Quantitative Study, Reifer Consultants, Inc. June 2007

[5] Baldwin, K., J. F. Miller, P. R. Popick, and J. Goodnight. "The United States Department of Defense Revitalization of System Security Engineering through Program Protection." Proceedings IEEE Systems Conference (SysCon), Vancouver, CA-BC, March 2012

[6] Wilshusen, Gregory C., "IT SUPPLY CHAIN: Additional Efforts Needed by National Security-Related Agencies to Address Risks." United States Government Accountability Office, Testimony before the Subcommittee on Oversight and Investigations, Committee on Energy and Commerce, House of Representatives, March 27, 2012

[7] Evans, G., "Flying fraudulently – How a Weak Supply Chain Became the USAF's worst Enemy," 2012, URL: http://www.airforce-technology.com/features/featuresupply-chain-us-air-force-fraudulent-parts.[Accessed April, 2015].

[8] Clark, R.A. and Schmidt, H.A., "National Strategy to Secure Cyberspace, Washington, D.C.", The President's Critical Infrastructure Protection Board, 2003

[9] United States Government Accountability Office, “HIGH-RISK SERIES An Update”, Report to Congressional Committees, February 2015

[10] John Steven, "Adopting an Enterprise Software Security Framework", *IEEE Security & Privacy*, vol.4, no. 2, pp. 84-87, March/April 2006

[11] SEI (Software Engineering Institute). 2009. CMU/SEI-2009-TR-010. SEI Secure Design Patterns. Pittsburg, US-PA: Carnegie-Mellon University, Software Engineering Institute.

[12] United States Department of Defense, Defense Acquisition Guidebook, Chapter 4 – Systems Engineering, Production Date: 15 MAY 2013

[13] Woody, Carol, Robert Ellison and William Nichols, “Predicting Software Assurance Using Quality and Reliability Measures” Technical Note, CMU/SEI-2014-TN-026 CERT Division/SSD, December 2014

[14] Software Engineering Institute, Architecture Analysis and Design Language, http://www.sei.cmu.edu/architecture/research/model-based-engineering/aadl.cfm , accessed April 2015

[15] Gacek, Andrew, John Backes, Darren Cofer, Konrad Slind and Mike Whalen, “Resolute: An Assurance Case Language for Architecture Models,” Cornell University, arXiv: 1409.4629v1 [cs.SE] 16, Sep 2014

[16] Cervin, A. & Lund U., Impact of Scheduler Choice on Controller Stability, CCACSD 2006

[17] Firesmith, D., Common Concepts Underlying Safety, Security, and Survivability Engineering, CMU/SEI-2003-TN-033, http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6553

[18] Bruce Lewis, Jérôme Hugues, Lutz Wrage, Peter Feiler, John Morley, "Model-Based Verification of Security and Non-Functional Behavior using AADL", *IEEE Security & Privacy, 2009*

[19] Julien Delange Wheel Brake System Example using AADL; Feiler, Peter; Hansson, Jörgen; de Niz, Dionisio; & Wrage, Lutz. *System Architecture Virtual Integration: An Industrial Case Study* (CMU/SEI-2009-TR-017). Software Engineering Institute, Carnegie Mellon University, 2009 http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9145

[20] Almorsy, M.; Grundy, J.; Ibrahim, AS., Automated software architecture security risk analysis using formalized signatures, Software Engineering (ICSE), 2013, pp.662,671, 18-26 May 2013

[21] Asnar, Y., Paja, E., Mylopoulos, J., Modeling design patterns with description logics: A case study (2011) Lecture Notes in Computer Science, 6741 LNCS, pp. 169-183.

[22] Ouchani, S., Jarraya, Y., Ait Mohamed, O., Model-based systems security quantification (2011) 2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011, art. No. 5971976, pp. 142-149.

[23] Mitre, Common Weakness Enumerations, http://cwe.mitre.org/