Systems Engineering a Model Based Systems Engineering Tool Suite: The Boeing Approach

Article *in* INCOSE International Symposium · July 2016

DOI: 10.1002/j.2334-5837.2016.00167.x

3 authors, including:

Brittany Friedland
The Boeing Company
5 PUBLICATIONS **46** CITATIONS
SEE PROFILE

Robert Malone
The Boeing Company
5 PUBLICATIONS **46** CITATIONS
SEE PROFILE

26th Annual INCOSE International Symposium (IS 2016)
Edinburgh, Scotland, UK, July 18 - 21, 2016

Systems Engineering a Model Based Systems Engineering Tool Suite: The Boeing Approach

Brittany Friedland
The Boeing Company
3180 139th Ave SE
Bellevue, WA 98005

Robert Malone
The Boeing Company
3003 W Casino Road
Everett, WA 98204

John Herrold
The Boeing Company
5000 E McDowell Rd
Mesa, AZ 85215

Abstract. Given the scope of some of the large scale system architecture models created at The Boeing Company, it is currently necessary for Boeing to customize its own Model Based Systems Engineering (MBSE) tool suites from Commercial-Off-The-Shelf (COTS) platforms to accommodate these models. Boeing successfully applies a modified set of Systems Engineering (SE) architectures to specify the required MBSE tool suites and also has developed a unique SE skill set for engineers creating these specifications and developing and deploying the subsequent tool suites. Furthermore, Boeing uses an MBSE tool suite to create these tool suite specifications as models, and also manages the configuration of the MBSE tool suite within the models.

Model Based Systems Engineering at Boeing

The Boeing Company has employed models to aid its development and design activities for decades, and began deploying Model Based Systems Engineering (MBSE) capabilities in earnest 15 years ago. One of the greatest successes Boeing has achieved with MBSE is the utilization of large scale system architecture models to manage program cost and schedule risk (Malone, Friedland, Herrold, & Fogarty, 2016).

In specific, within Boeing Commercial Airplanes (BCA), system architecture models are sufficiently detailed that conceptual models are matured to the point where they achieve the fidelity of detailed design data. As a result, the underlying MBSE tool suites that produce these models are accredited to produce authoritative design reference data for the airplanes they support. Furthermore, the results derived from analysis of the models are of sufficient quality to be used for civil certification purposes. The accreditation of the tool suites is, in large part, based on the configuration management model and process described later in this paper.

To achieve this model fidelity, a robust, uniquely tailored, MBSE tool suite based on a COTS platform is required that can capture large scale systems architecture models in very fine detail. Boeing performs extensive customization on COTS MBSE platforms in house to produce the required tool suites.

As depicted in Figure 1 below, the core model elements that must be supported by an MBSE tool suite are common ones; integrated requirements, functions and logical system elements that drive, and are ultimately consumed by the physical design of the aircraft. The models also incorporate measurement and control and verification and validation elements.

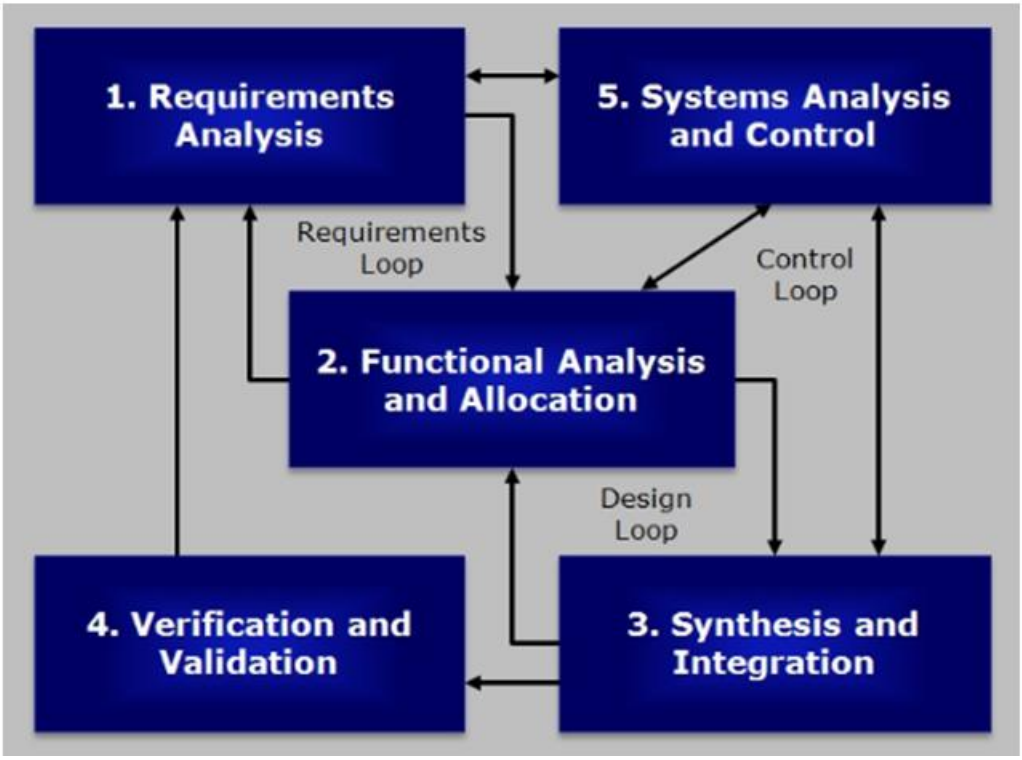

Figure 1. Core Elements of BCA System Architecture Models

To build these MBSE tool suites in house, it was necessary that Boeing develop a set of processes to guide the specification, development and deployment of the tool suites, and a unique set of SE skills within the cadre of engineers who execute these processes. This paper outlines this set of processes and SE skills.

MBSE Tool Suite Development Process

An MBSE tool suite is, in itself, a system, and, as such, needs to be architected and specified using SE processes, albeit processes slightly different from those classically employed. Figure 2 below depicts the commonly developed core of integrated SE architectures. Requirements, functions, and functional interfaces are developed that inform the logical architecture (or design synthesis). By verifying and validating the logical architecture model against the requirements and functional architecture, it can be assured that the logical architecture complies with the requirements and fulfills the functionality, and that the requirements and functional architecture completely define the logical architecture. The physical architecture can then be built to comply with the specification defined by the logical architecture with a high level of assurance that the final physical architecture will fulfill its intended purpose.

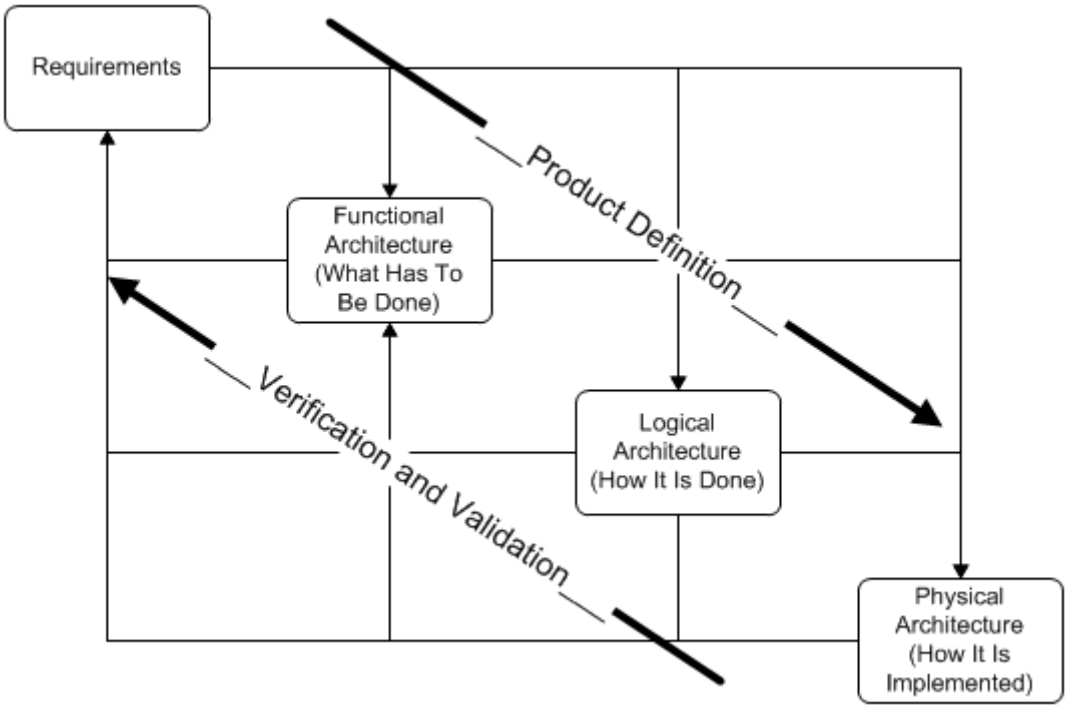

Figure 2. Classical Integrated SE Architectures

The MBSE tool suite specification processes that Boeing implemented are derived from Information Technology (IT) development processes, and essentially produce a modified version of these classic integrated SE architectures. These MBSE tool suite specification processes produce three main deliverables: Work Processes; Unit Task Specifications; and, Information Models. Compared to standard integrated SE architectures, Work Processes are the functional architecture of the MBSE tool suite; they specify what the MBSE tool suite needs to do to support systems engineers as they execute engineering processes to create models. Unit Task Specifications are a hybrid of requirements and functional architecture; they specify what the MBSE tool suite needs to do for the user in much greater detail, and also enumerate the business requirements that must be enforced and/or supported by the MBSE tool suite as users create models. An information model is then derived that specifies the data and relationships that need to be captured in the MBSE tool suite as models are created. The information model for the MBSE tool suite is analogous to a system design synthesis or logical architecture.

The relationship between MBSE tool suite specification deliverables and classic SE integrated architectures is illustrated below in Figure 3.

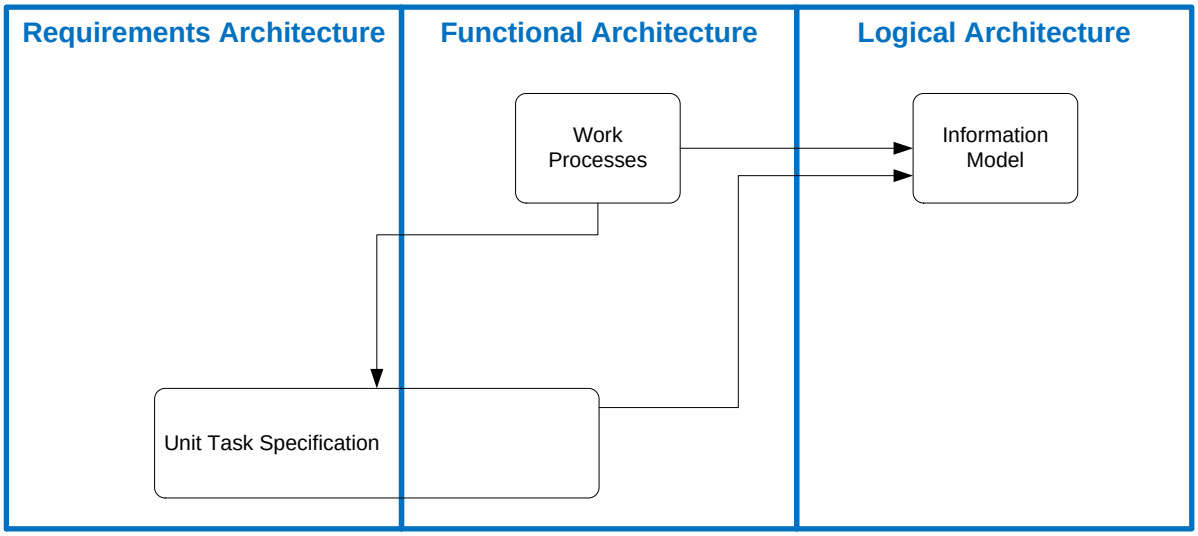

Figure 3. Relationship between MBSE Tool Suite Specification Deliverables and Classic SE Architectures

Work Processes. Defining work processes begins with listing the use cases that define the high level business functionality the MBSE tool suite supports. The use cases are displayed on use case diagrams that depict the relationships and dependencies between the use cases and also detail which actors (engineering roles) execute the use cases. The use cases are then decomposed into process steps. The process steps are displayed on process flow diagrams that depict the sequence in which the process steps are executed, the data required by each process step and their source, and the data each process step produces and their consumers. The process steps are then decomposed into unit tasks that are detailed in the Unit Task Specification.

To properly reflect organizational business processes, the process steps included in the work processes should be culled from mature, well defined business policies and procedures. Any new process steps defined during work process development should be fed back into the business processes and procedures.

The high level content of the Work Processes deliverable is detailed in Table 1 below:

Table 1. Work Process Deliverable Outline

Introduction
• Purpose
• Assumptions
• References
• Background
Requirements
• Use Case Title
○ Overview
○ Data Model
○ Process Flow
○ Functional Requirements
▪ Sub-Use Case
• Overview
• Unit Task Enumeration

Unit Task Specification. Unit tasks describe the process steps and their flow in fine detail. The high level content of the unit task specification deliverable is detailed in Table 2 below. Roles define who is permitted to execute the process step in the MBSE tool suite. Preconditions specify the state in which the MBSE tool suite must exist when a user attempts to execute the unit task. For example, the user must have privileges in the area of the model being manipulated, and that area of the model must be the appropriate location to create or modify data of the type being manipulated. Post conditions specify the state in which the MBSE tool suite will exist following execution of the unit task, and where the user will be left in the MBSE tool suite interface after execution of the unit task. Expected results specify what will be added to or modified within the model after execution of the unit task. Exceptions include actions taken by the MBSE tool suite if pre-conditions are not met, or the user is attempting to violate a business requirement, and the accompanying notifications that will be displayed to the user. Quality criteria usually specify usability requirements. Level of automation specifies any data that will be created, deleted or modified by the MBSE application on behalf of the user as the unit task is executed. Business rules are captured that specify how the model will be modified as the user executes a unit task and the constraints the MBSE tool suite must impose upon the users as they attempt to create, modify or delete data. From the business rules, model data integrity is ensured by embedding a set of rules controlling model creation and modification within the MBSE tool suite. Examples of these rules are: object type A can only be linked to one of object type B by using link C; and, data of type integer are forced to be entered as integers.

Initially, Boeing MBSE tool suites applied these rules a priori; users were required to enter model data with constrained composition and content. Experience demonstrated that this data integrity approach is quite cumbersome for users. To improve usability, Boeing MBSE tool suites are evolving to apply these rules posteriori; model correctness is enforced in the release process. That is, if a model not correct, it cannot be formally released. The release process invokes “must have” and “check” rules for construction and content. These rules can also be invoked through user request preceding data release and a checklist of currently invalid model elements provided. The user would then correct the errors, or provide a rationale for the design anomalies, before the model is formally reviewed and released.

Table 2. Unit Task Specification Deliverable Outline

Unit Task
• Roles
• Pre-Conditions
• Post-Conditions/Expected Results
• Exceptions
• Quality Criteria
• Level of Automation
• Business Requirements

Information Model. One of the key deliverables in the development of the MBSE tool suite is the information model. Once the unit tasks and their flows have been created, they can be translated into an information model represented as a Unified Modeling Language (UML) 2.0 class diagram and captured in the Information Model deliverable. The information model for the MBSE tool suite comprises a set of sub-typed objects, relationships (links) and attributes (properties). An example of an object would be a person. Attributes of a person would be height, weight and hair color. An example of a relationship between two people would be a sibling link; person 1 is a sibling of person 2. A sample UML 2.0 class diagram is depicted in Figure 4 below. Business rules from the unit task specification are then applied within the MBSE tool suite to constrain model manipulation. For example, the MBSE tool suite might require that height is entered in inches, weight in pounds, and that a sibling relationship can only be created between objects of type person. Business rules are also applied to relationship cardinality; how many objects of one type can be related to objects of another type. For example, if there were a relationship of type "parent" in the MBSE tool suite, the link might be constrained so that a person can only have two parents, but a parent can have many children.

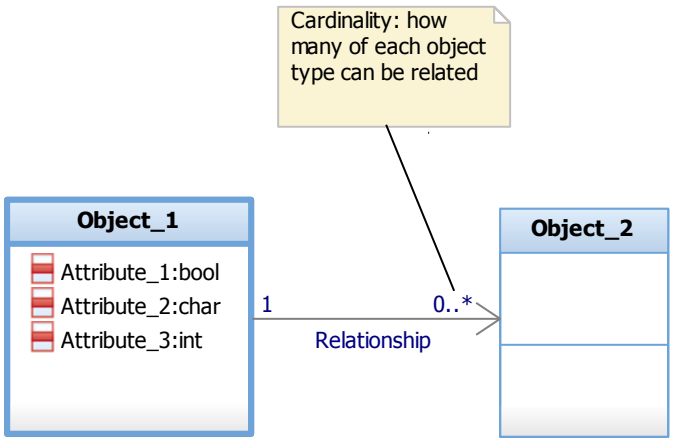

Figure 4. A Sample UML 2.0 Class Diagram

Transforming Work Processes and Unit Tasks into an Information Model. Figure 5 below provides an illustrative example of how information models are derived from work process flows and end user requirements. The work process illustrated describes how a systems engineer would create a functional architecture, allocate requirements to the functional architecture, and then allocate functional architecture elements to logical architecture elements. To create functions within the MBSE tool suite, there must be an object of type function that can be decomposed. The functions must be able to have child inputs and outputs, and it must be possible to create a relationship between the inputs and outputs. There must also be requirement object types, and it must be possible to create an allocate relationship between

requirement objects and function objects. Finally, there must be an object of type system that can be decomposed, and it must be possible to create an allocate relationship between function objects and system objects. The system objects must be able to have child inputs and outputs, and it must be possible to create a relationship between the inputs and outputs on function objects to inputs and outputs on system objects.

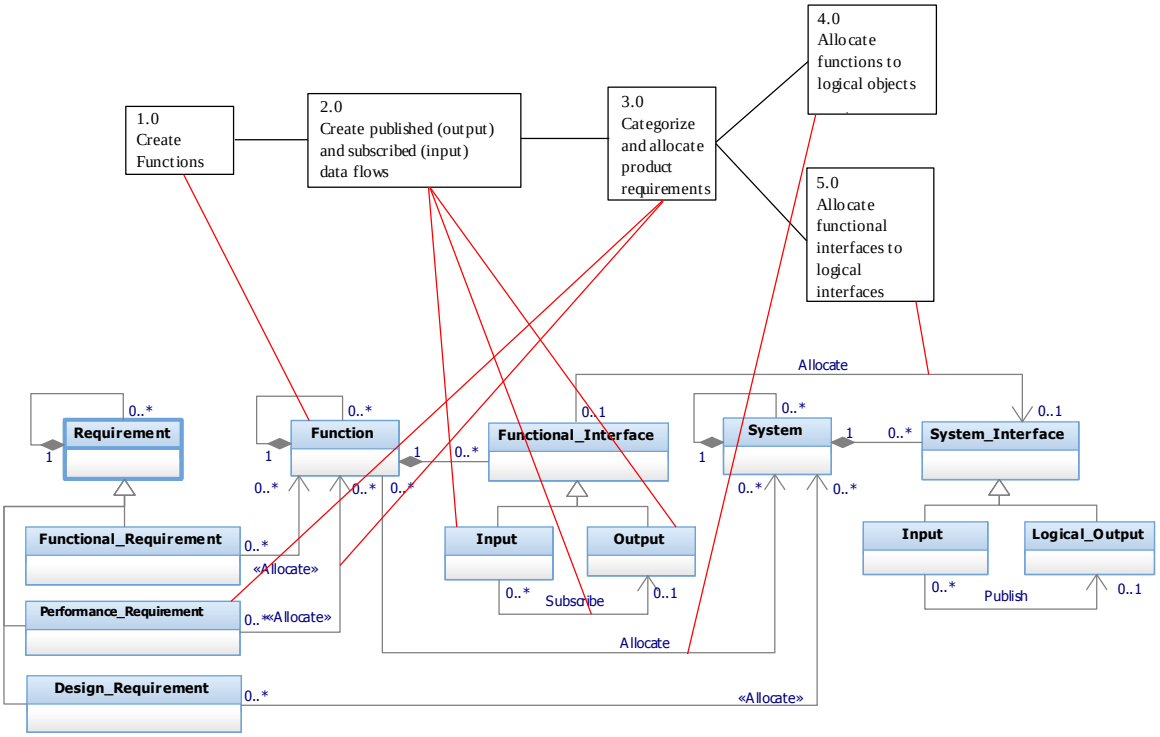

Figure 5. A Process Flow Related to Information Model Elements

The work processes, however, do not completely convey engineering end user modeling requirements and expectations. To enumerate these requirements and expectations, instance diagrams were developed, similar to that in Figure 6 below, using a custom notation that captures abstract representations of the modeling elements and relationships the end users need to create to make the actual models useful.

Figure 6 depicts the constructs and terms used by BCA digital network designers as they capture their design data. System components (Line Replaceable Units (LRUs)) may be modeled simply as hardware or, if they contain loadable software, as hardware with Hosted Applications. Signals, of varying Aeronautical Radio, Incorporated (ARINC) protocols, comprise data packets that are detailed as a Signal Format. LRUs and Hosted Applications transmit output Signals and receive input Signals. To enforce reconciliation between transmitted output Signals and received input Signals, inputs and outputs are linked by a Subscribe relationship. Since output Signals, and subscribed input Signals, must adhere to the same Signal Format, a single Signal Format is created that is owned by the output Signal. Input Signals can be transformed to different protocols and re-transmitted as output Signals; in these cases Signal Formats from one protocol are Embedded in the Signal Formats of other protocols. Signals are transmitted and received (Realized) on Application Ports that, in turn, are Transmitted on Physical Ports. Physical ports are then connected to some sort of Transport Element, in this case an A429 Data Bus.

These graphic notations created a common language that is used by the systems engineers and system designers to structure the MBSE tool suite so that the design engineers can closely

follow their normal work processes and capture their required data deliverables within the models.

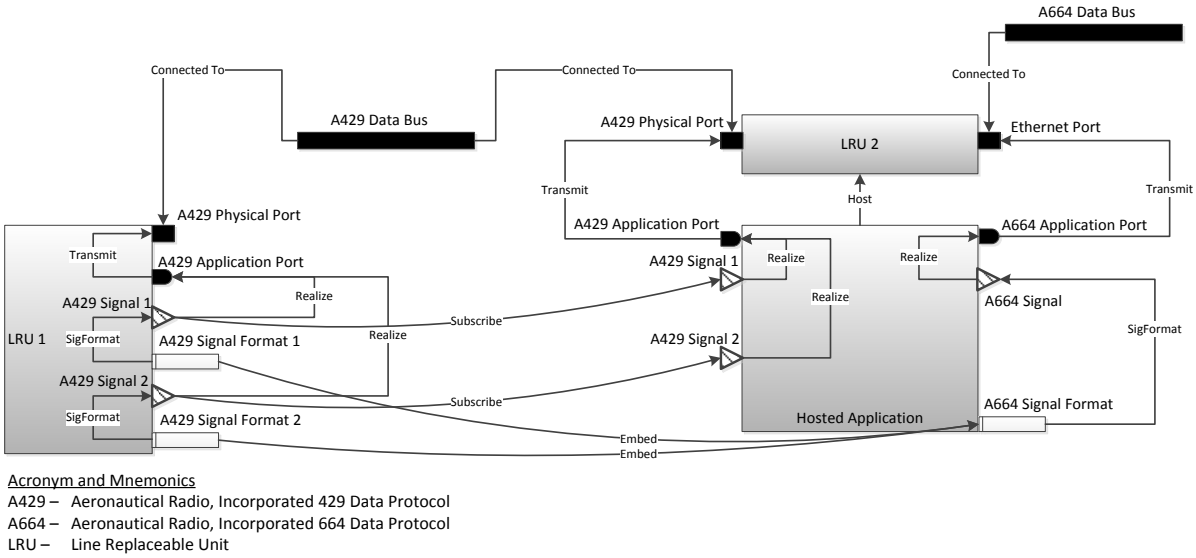

Figure 6. Developing an Understanding of Engineering End User Modeling Requirements

MBSE Tool Suite Specification Deliverables. Since the MBSE tool suite development deliverables are, essentially, SE deliverables, it is not surprising that the MBSE tool suite development processes benefit from the application of MBSE. Boeing models and integrates use cases, process steps, process flows, unit tasks, business requirements and information models within a MBSE tool suite. A sample information model that defines and relates these data elements is depicted in Figure 7 below.

Although complete and accurate specifications are required for both tool suite testing and civil accreditation, actual tool suite development follows a fairly agile process. The tool suite architects are co-located with IT tool suite developers, and tool suite functionality is developed through continuous, real time collaboration between the architects and IT developers. The specification process ensures that the results of this collaboration are fed back into the final set of complete, rigorous tool suite specifications.

Not only does MBSE aid the development of the MBSE tool suite specification deliverables, it also provides two ancillary benefits. The MBSE tool suite specification models themselves demonstrate the value of MBSE. Furthermore, the MBSE tool suite architects enhance their modeling skills and their skills in using MBSE tool suites by creating these specification models.

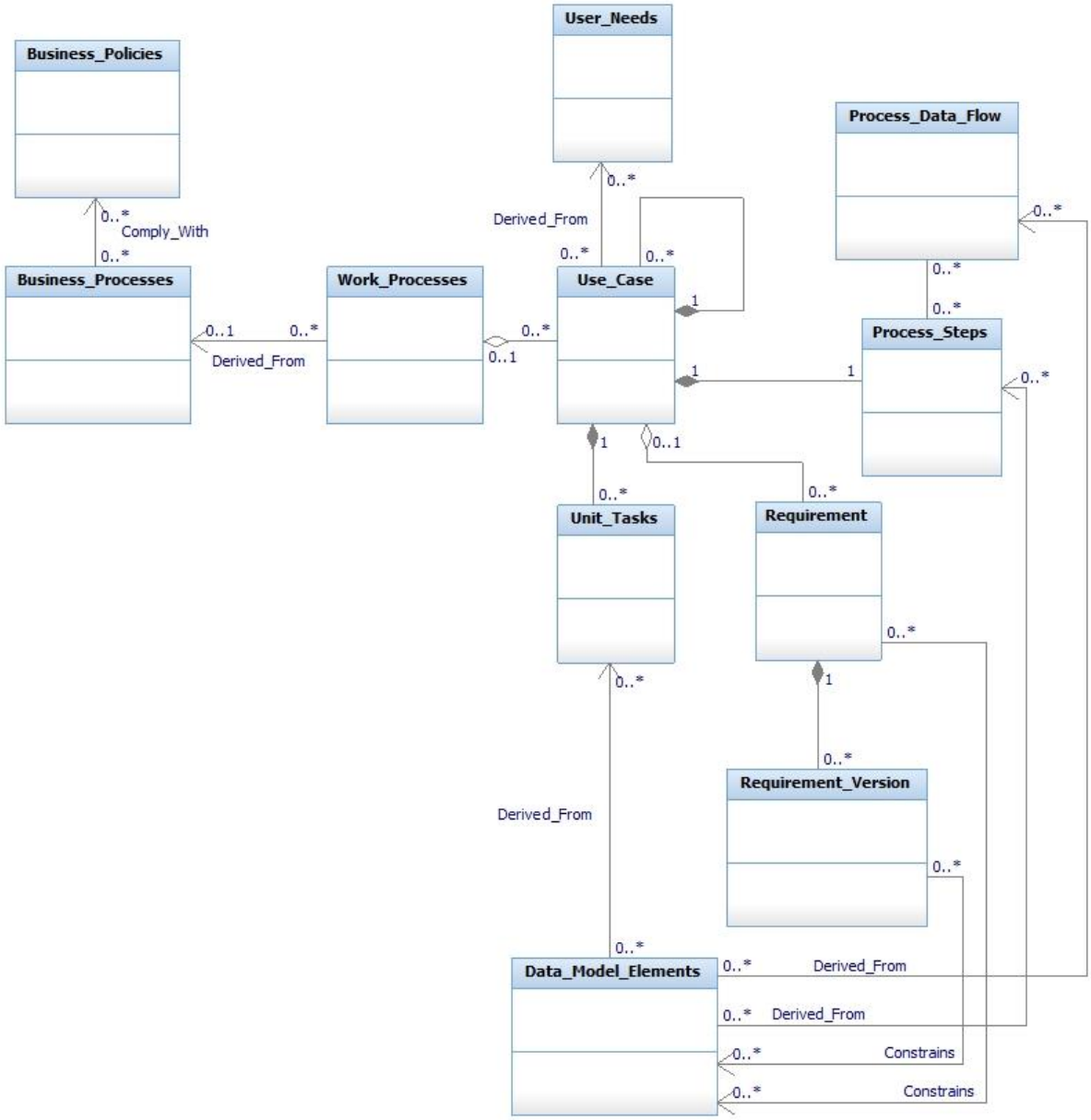

Figure 7. Sample MBSE Tool Suite Specification Information Model

Table 3 below details the type and volume of data required to completely specify an MBSE tool suite to meet Boeing's needs.

Table 3. Typical MBSE Tool Suite Specification Model Data Volume

Use Cases	~1K
Business Requirements	~25K
Data Object Subtypes	~300
Data Relationship Subtypes	~75
Data Attributes	~2K

Systems Engineering Skills Required to Develop an MBSE Tool Suite

The Boeing experience supports the assertion of Evans, (2003) that the most significant complexity of engineering tools is not technical. It is in the domain itself, the activity or business of the user. He states "Engineering processes are complex. The breadth of knowledge required to fully grasp some engineering processes can be daunting. Engineers are usually not aware of how complex their mental processes are in the course of their work; they mentally navigate all the rules that must be employed in their processes, reconcile contradictions, and fill in gaps with common sense."

As depicted in Figure 8 below, in order to grapple this complexity and properly specify the desired MBSE tool suite, a cadre of MBSE "Business System Managers" (BSMs) was developed who form the bridge between the engineering user community and the IT community developing the tool suite. The members of this cadre became skilled at interviewing the user community regarding its processes, data, and modeling needs, and distilling and conveying this information as a set of specification deliverables in a form that the IT community could readily understand and consume. Natural language needed to be translated into use cases, processes steps, task steps, information models, and business requirements that comprise the specification deliverables. Since use case and information model development are not part of the standard SE skill set, the BSMs also developed proficiency in these areas.

These MBSE tool suite specifications not only detail the data structure and behavioral constraints imposed on the MBSE tool suite, but also define any utilities and automation required within the MBSE tool suite, and the BSMs also became skilled at specifying these features.

BSMs also need to be skilled at developing queries to analyze the model content.

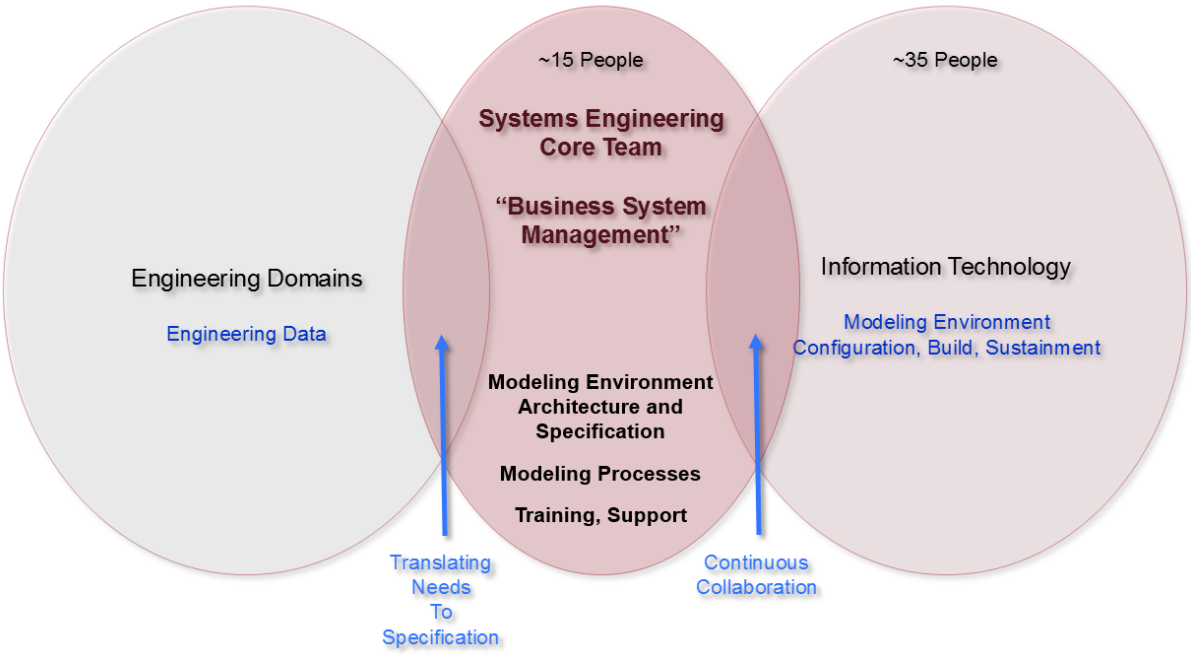

Figure 8. Business System Managers: the Bridge between Engineering and Information Technology

Furthermore, BSMs need to be skilled at verification and validation of the MBSE tool suite against its driving requirements and work processes.

Finally, BSMs need to be capable of defining the content of MBSE tool suite updates and also need to be capable of managing and maintaining configuration control over release baselines. Configuration management of MBSE tool suite update releases is described in the next section of this paper.

Table 4 below lists the duties and responsibilities of a BSM at Boeing:

Table 4. Business System Management Duties and Responsibilities

• Represents the needs of the MBSE tool suite owner and the stakeholders
• Links the needs of the owner and users, and the MBSE tool suite
• Provides business-related knowledge that orients the architecture of the MBSE tool suite
• Defines and refines business processes
• Establishes the architecture (integrated processes, unit tasks, business requirements, information model) of the MBSE tool suite from the user perspective, ensuring its overall viability
• Maintains the scope of the MBSE tool suite in compliance with the formally established requirements
• Defines user training content
• Elaborates integration test plans
• Defines system test and acceptance test scripts and plans
• Validates the MBSE tool suite being delivered and ensure it corresponds to the needs of the owner and users
• Raises project issues and risks, as well as quality and scope deviations
• Escalates issues, risks and problems to the identified authority when the situation so requires
• Documents change requests for the program organization and participates in impact analysis
• Assigns and approves severity and priority of CRs
• Approves completion status and closure of CRs
• Ensures timely delivery of and quality of business requirements, processes and training content
• Ensures the delivered solution meets user needs
• Ensures adherence to approved development and delivery processes

MBSE Tool Suite Configuration Management

Boeing manages the configuration of the MBSE tool suite within an MBSE tool suite specification model. Figure 9 below depicts how the tool suite specification information model is extended to support tool suite configuration management.

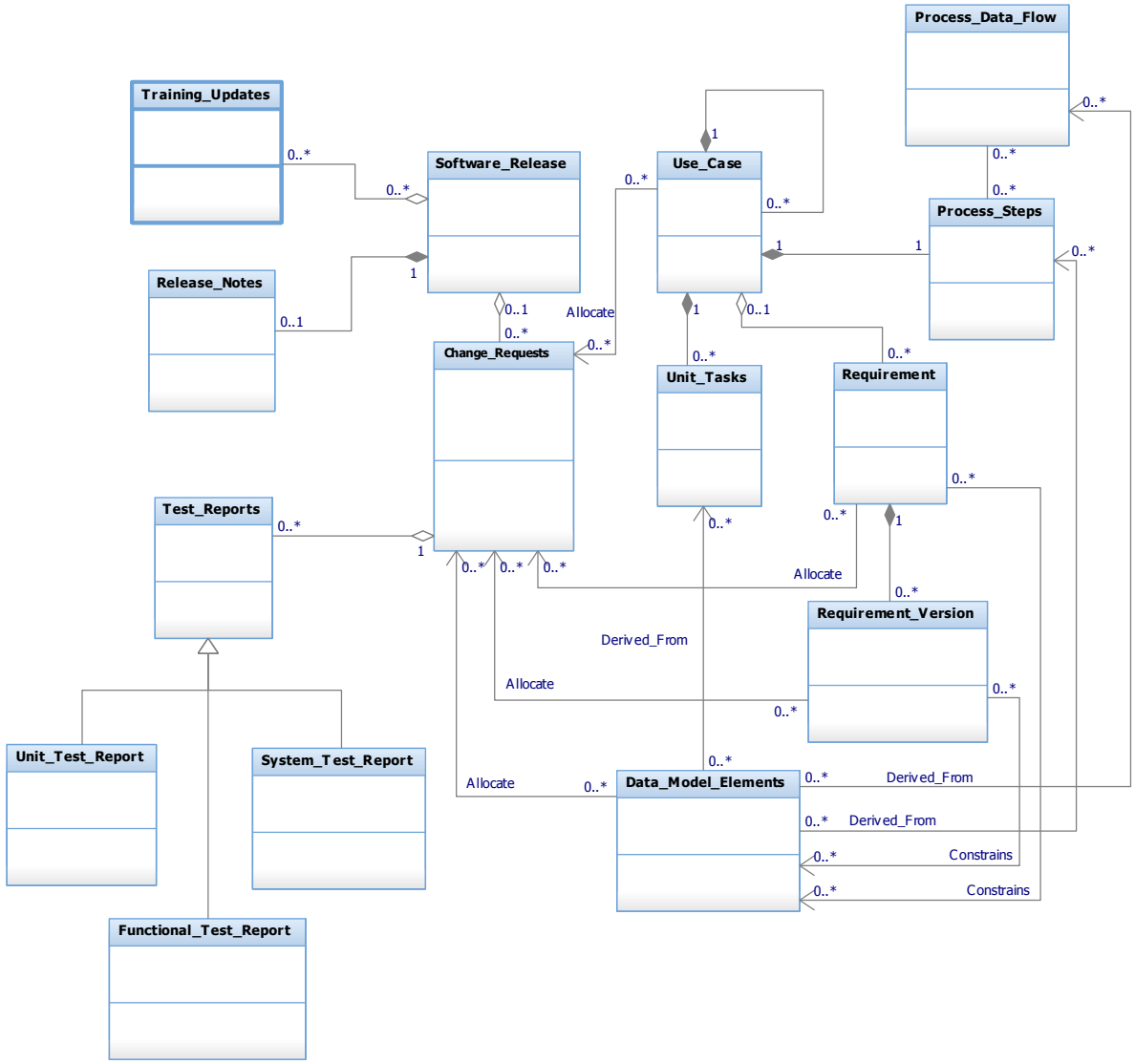

Figure 9. MBSE Tool Suite Configuration Management Information Model

An MBSE tool suite update release is defined by Change Requests (CRs) raised against the last base lined MBSE tool suite release. A software release object is created within the model, and CR objects are created and assigned to the software release object. A CR impacts tool suite use cases, and the affected use case objects are linked to the CR object. Since not all business requirements included within a decomposed use case are affected by the CR, those business requirements affected by the CR are directly linked to the CR object. If a new business requirement is defined, the requirement object is linked to the CR. If a CR modifies a business requirement, a version of the requirement is created, and the requirement version object it linked to the CR object. If the CR modifies the tool suite information model, the information model element objects impacted by the CR are also linked to the CR object. Test reports generated during the verification and validation of the MBSE tool suite release are linked to the CR object. Finally, training and process updates, and the release notes accompanying the MBSE tool suite software release are linked to the CR object. Tool suite accreditation is achieved by demonstrating the ability to detail, for each software release, what CRs were implemented in the release, what Use Cases were impacted by each CR, the requirements impacted by each CR, the version of each requirement impacted by each CR, all previous

versions of the impacted requirements, all software changes made within the modeling environment, and the results of all verification activities.

Conclusion

Since an MBSE tool suite is a system, its architecture and specification benefits from the application of SE processes. It is relatively straightforward to modify classic SE architectures to specify MBSE environments and to transfer SE skills to MBSE tool suite architecture and specification activities. Furthermore, since MBSE aids the development of SE architectures, an MBSE tool suite is also well suited to create MBSE tool suite specification models and to manage the configuration of MBSE tool suite software releases.

References

Evans, E. 2004. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Indianapolis, US-IN: Addison-Wesley Professional.

Object Management Group. 2005. Unified Modeling Language Version 2.0. Needham, US-MA. Object Management Group.

Malone, R., Friedland, B., Herrold, J., & Fogarty, D. 2016." Insights from Large Scale Model Based Systems Engineering at Boeing." Paper presented at the International Symposium of INCOSE, Edinburgh, UK, 18-21 July.

Biography

Brittany Friedland spent the beginning of her career in the Oil and Gas Industry before making a career change to work in the Aerospace Industry. She currently works for Boeing developing and deploying Model Based Systems Engineering tools and processes with the Boeing Enterprise (Commercial and Defense).

Robert Malone has spent his entire thirty-five year career as an aerospace engineer and has specialized in systems engineering for the past twenty-six years. He has held positions in aircraft maintenance operations, aviation security system integration, human factors, reliability, maintainability and testability. His current focus is on developing computer-based tools and processes supporting systems engineering, large scale system integration, and system integration modeling.

John Herrold is currently the System Architect for the Integrated Product Architecture enterprise systems engineering program that provides a service ready model based systems engineering solution (process, tool and training) for systems and design engineers. John has been a Boeing employee for 35 years and has worked mostly in the engineering analysis domain, supporting many of the Boeing Commercial and Military Airplane products. John is a designated Boeing Technical Lead Engineer and a member of the International Council on Systems Engineering (INCOSE). John has a BSEE from the University of Washington.