26th annual INCOSE international symposium

Edinburgh, UK
July 18 - 21, 2016

A Systematic Process for Functional Decomposition in the Absence of Formal Requirements

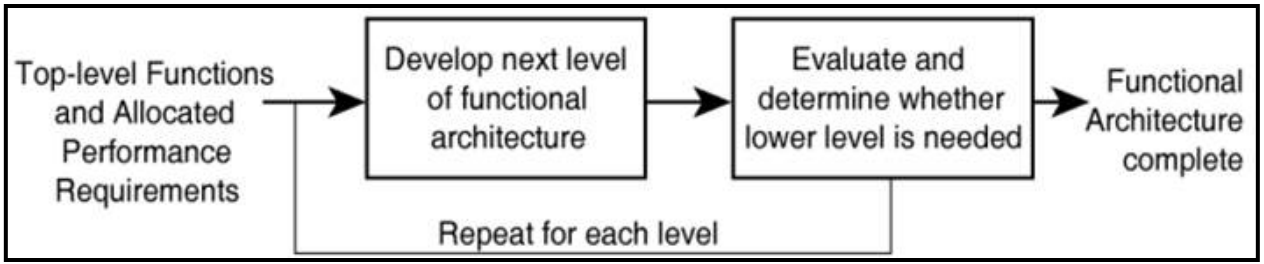
Ryan Simko, Richard Wise, Erika Brimhall,
John Huggins, Whit Matteson

Georgia Tech Research Institute

19 July 2016

Introduction

- One of the earliest steps in designing a new system is understanding its functionality: this is often achieved through a functional decomposition.
- Traditional guidance for decomposing system functionality relies on formal system requirements.
- This briefing introduces a process for developing functional decompositions in the absence of formal requirements



INCOSE Functional Analysis/Allocation Process (Haskins, 2011)

Process Goals

- The process is intended to produce a robust functional decomposition with the following characteristics:
  - **Coverage.** The key to ensuring full coverage is to identify and fill gaps in the functionality.
  - **Consistency.** This is achieved by preventing contradictions within the functional hierarchy and ensuring compatibility between structural and behavioral views.
  - **Reusability**. A functional decomposition should give implementation-agnostic functions that defines what the system must do rather than *how* the system must work.

What is a Function?

- The INCOSE Systems Engineering handbook defines a function as “a characteristic task, action, or activity that must be performed to achieve a desired outcome” (Haskins, 2011).
- The table represents an extension of the Integration Definition for Function Modeling (IDEF0) framework which was used for the standard functional definition.

Definition Component	Description
Parent Function Reference	A reference, whether by name or unique id, to the parent function
ID	An unique identifier assigned to the function
Name	A verb-noun phrase used to succinctly describe the function
Narrative Description	A brief, plain-English textual description of the function that explains its purpose and usage in the context of other functions
Input	The data or objects that are transformed by the function into output (FIPS PUBS 183, 1993)
Output	The data or objects produced by a function (FIPS PUBS 183, 1993)
Control	The conditions required to produce the correct output (FIPS PUBS 183, 1993)
Enabler (Mechanism)	The means used to perform the function (FIPS PUBS 183, 1993)
Relevant Decomposition Bin	The verb phrase(s) (i.e. decomposition bin) used to consistently describe the basic behavior exhibited by the function (Hayhurst, et al., 2007)
Applicable Mission Phase	The state(s) or significant condition(s) of the system applicable to the function (Friedenthal, 2015)
Applicable Platform Type	The category(s) of systems or technologies applicable to the function
Source Document Reference	A bibliographic reference to the source document containing the information used to define the function

Function Attributes

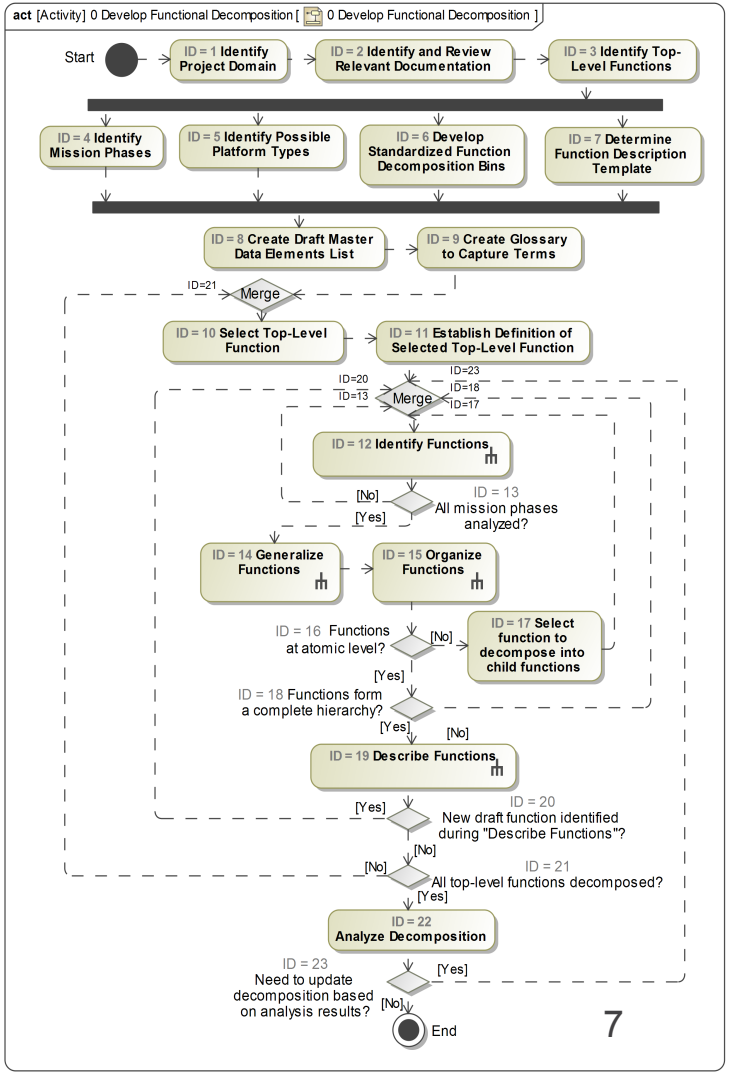
- The process enforces examining and capturing the functionality from multiple orthogonal perspectives, including:
  - **Decomposition Bins:** verbs clearly defined to describe basic behaviors found in function names throughout a particular decomposition (e.g. Determine, Receive, Execute, Convey Status).
  - **Mission Phases:** a way of capturing the basic ideas behind system states. Friedenthal et al. says that, “a state represents some significant condition” of the system. A state typically “represents some change in how the [system] responds to events and what behaviors it performs” (Friedenthal, 2015). (e.g. Taxi, Takeoff, Landing)
  - **Platform Types:** categories of systems or technologies for which a functional decomposition is intended to cover. (e.g. Fixed-wing Aircraft, Rotorcraft, or Vertical and/or Short Take-off and Landing (V/STOL) Aircraft)

Model-Based System Engineering (MBSE) Approach

- MBSE served as the basis for capturing and documenting the functional decomposition process.
  - UML Activity Diagrams were used to provide precise semantics in documenting this process.
  - The process was broken down into a series of actions and decisions, to provide details about:
    - What needs to be done,
    - how to do it, and
    - an example of the action being performed.
  - Control flows were used to illustrate the process flow.

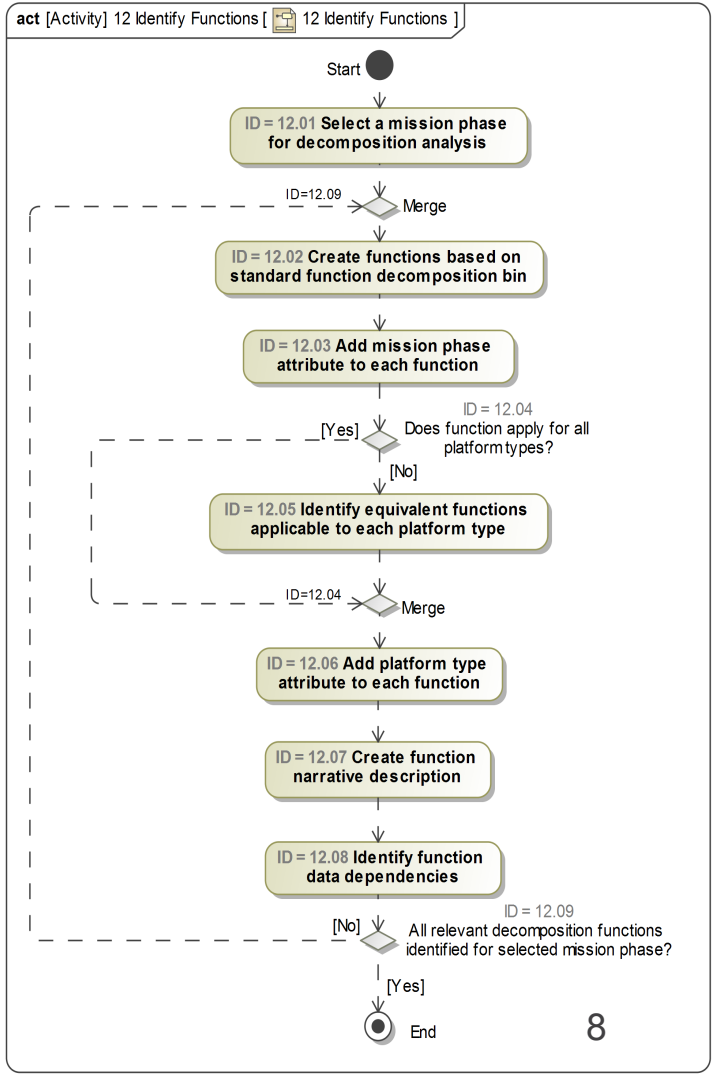
Top-Level Process

- The top-level process consists of 23 steps
- 4 of those steps contain sub processes;
  - Identify Functions
  - Generalize Functions
  - Organize Functions
  - Describe Functions
- Total of 64 steps across five key diagrams



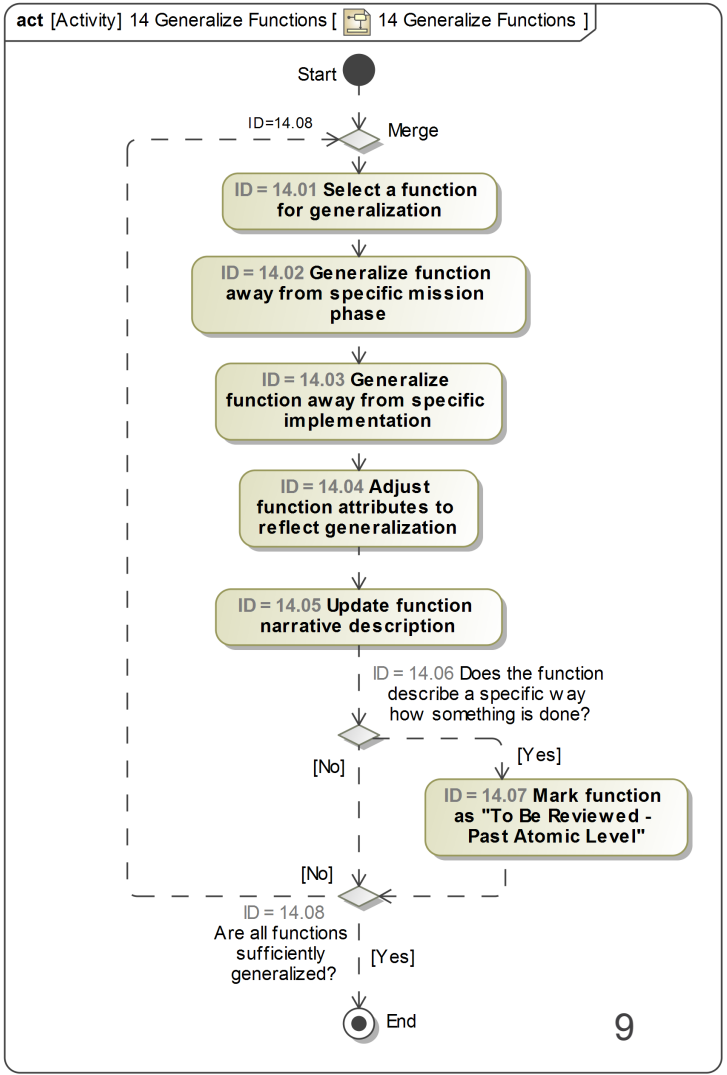
Identify Functions

- *Identify Functions* is where the domain specific attributes developed earlier in the process are populated and used to help guide the generation of necessary functions.
- Functions are developed based on mission phase, decomposition bins, and platform type.
- When this process sub-step is complete, functions will be appropriately tagged with attribute data, have a base definition, and some notional data dependencies for sequencing later on in the process.



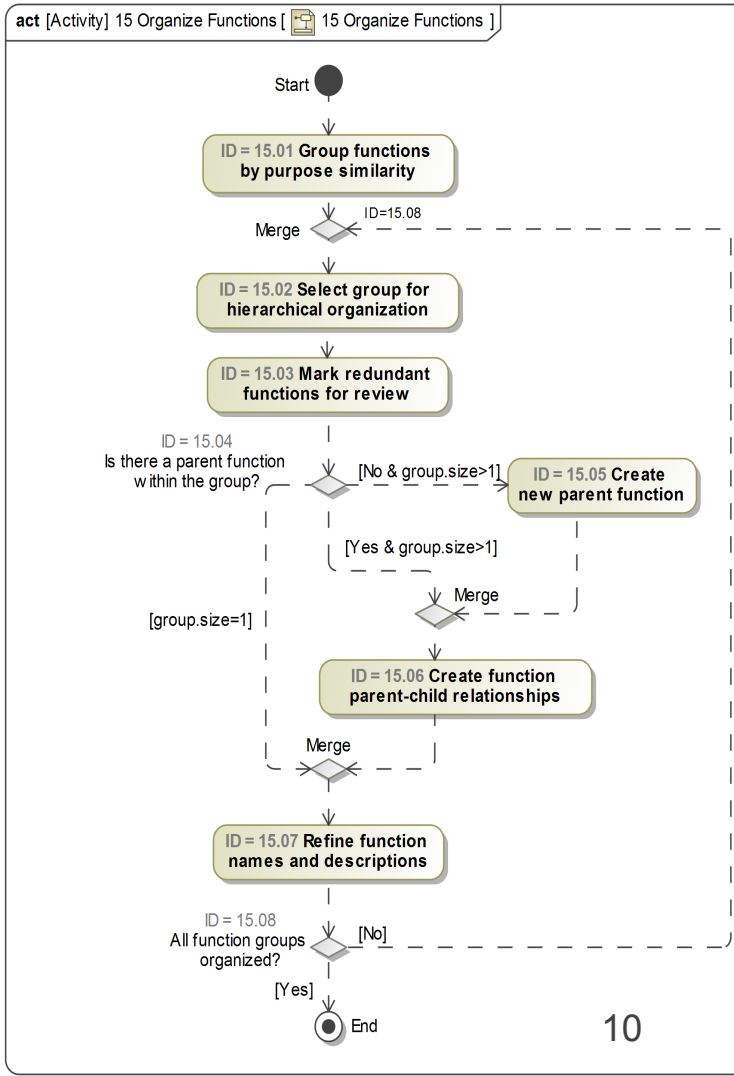
Generalize Functions

- *Generalize Functions* takes the functions developed in *Identify Functions* and works to make them more generic (applicable across multiple mission phases and possible implementations).
- As an example, if you have aircraft functions that are identical except for the mission phase of the aircraft, these functions can be generalized and combined into one function.
- Functions must be updated appropriately once they have been generalized to reflect the new definition, data dependencies, or name.



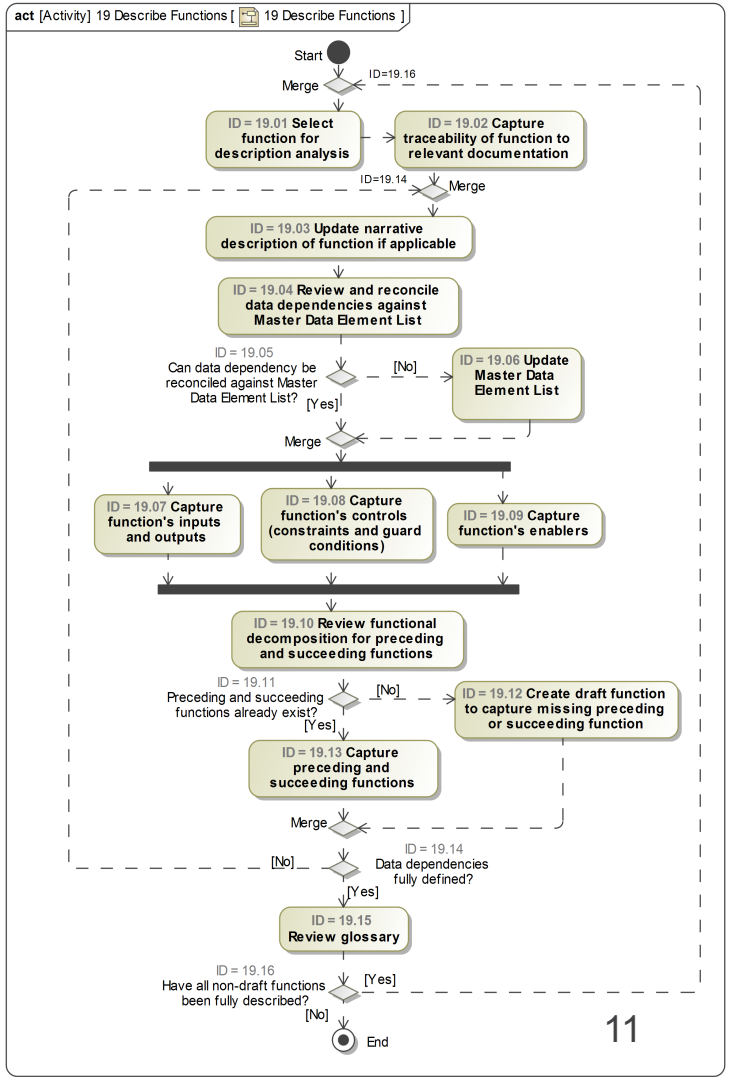
Organize Functions

- *Organize Functions* is where the list of functions developed to this point is structured and assigned a hierarchy based on parent-child relationships.
- The goal is to create groupings of functionality based on similar purpose.
- The end product should be a hierarchical function structure consisting of multiple parent-child relationships.
- Since functions were generalized in the previous sub-process, some functionality is potentially redundant and is marked for review.



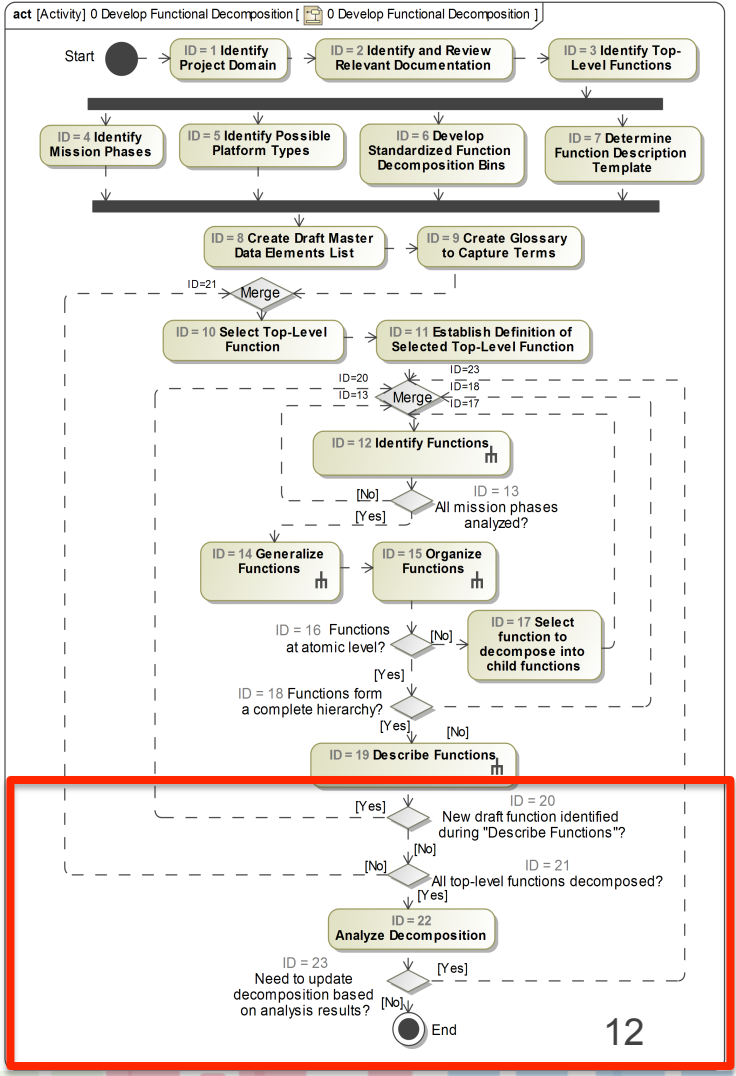
Describe Functions

- *Describe Functions* updates the narrative description from the standard function definition and captures the additional fields that have yet to be addressed earlier in the process.
- Capturing preceding and succeeding functions can identify gaps in the functional decomposition.
- The Master Data Element List and Glossary are reviewed and updated as needed.



Process Summary

- Upon completion of *Describe Functions*, there are some final feedback loops to address process completion.
- Any changes to the decomposition, require that the process be followed again to assess the impact.



Validating the Process Goals

- The process is intended to produce a robust functional decomposition with the following characteristics:
  - Coverage
  - Consistency
  - Reusability
- A functional decomposition for avionics software was developed using this process to validate these goals.

Validation - Coverage

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

- Ensured functionality would be added to address multiple orthogonal perspectives, including the decomposition bins, mission phases, and platform types (steps 12.02-12.06 and 14.01-14.03).
- Identified and filled gaps in the decomposition. Gaps were most commonly identified when piecing together complete parent/child relationships (steps 15.04-15.06) and when determining the source of data that was required as inputs into identified functions (steps 12.08 and 19.10-19.14).
- Enforced functions being decomposed into sub-functions until they could not be further decomposed without becoming implementation-specific (steps 14.06-14.07 and 16).

Validation - Consistency

- Enforced each function in the hierarchy being composed of its child functions, which ensures consistency throughout the hierarchy (steps 15.04-15.06).
- Activity Diagrams depicting behavioral views of the functionality were developed (step 19.10) and the process ensured they were consistent with the hierarchical view of the functions.
- A data model at a conceptual level was developed for the functional decomposition (steps 8 and 19.04-19.07), resulting in consistent inputs and outputs between functions.

Validation - Reusability

- Ensured the functions would remain implementation-agnostic (steps 14.01-14.08).
- Decomposed the functions to the lowest possible level, while remaining implementation-agnostic (steps 14.06-14.07 and 16).
- In addition, the avionics functional decomposition was reviewed by avionics Subject Matter Experts who agreed the decomposition was an accurate representation of avionics software functionality.

Future Work

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

- Analyze the avionics decomposition to understand the amount of coupling and cohesion occurring between functions.
- Implement the functional decomposition process to develop function libraries for other domains beyond avionics software.
- Identify process improvement opportunities by having non-systems engineers implement the process and provide feedback.

Conclusions

- This briefing presented a systematic process to perform an implementation-agnostic functional decomposition in the absence of formal requirements.
- The systematic process ensures the functional decomposition has full coverage of the system's functionality, is consistent throughout the functional architecture, and is reusable across various system domains and organizations.
- The functional decomposition process has been used in the avionics software domain to successfully develop a functional decomposition exhibiting the characteristics of coverage, consistency, and reusability.

Questions?

Contact Information:

Ryan.Simko@gtri.gatech.edu, Georgia Tech Research Institute

References

- Camarinha-Matos, L. M., ed. 2002. *Collaborative Business Ecosystems and Virtual Enterprises*. New York: Springer Science and Business Media.
- Department of Defense, Systems Management College. 2001. “Systems Engineering Fundamentals.” Supplementary Text, Defense Acquisition University Press.
- Faisandier, A. 2013. *Engineering and Architecting Multi-Disciplinary Systems, Volume 3: Systems Architecture and Design.* Belberaud, France: Sinergy’Com.
- FIPS PUBS (Federal Information Processing Standards Publications). 1993. Draft Federal Information Processing Standards Publication 183: Integration Definition for Function Modeling (IDEF0). Gaithersburg, MD.
- Friedenthal, S., A. Moore, and R. Steiner. 2015. *A Practical Guide to SysML, 3rd Edition.* Amsterdam: Elsevier.
- Haskins, C., ed. 2011. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* Version 3.2.2. Revised by K. Forsberg et al. San Diego, CA: INCOSE.
- Hayhurst, K. J., et al., 2007. “Preliminary Considerations for Classifying Hazards of Unmanned Aircraft Systems.” NASA Langley Research Center.
- ISO/IEC (International Organization for Standardization/International Electrotechnical Commission). 2007. ISO/IEC 26702:2007. Systems engineering – Application and management of the systems engineering process. Geneva, CH: ISO/IEC
- No Magic, Inc. 2013. “UML Profiling & DSL, Version 18.0, User Guide.” Allen, US-TX.
- The Open Group. (2014). “Technical Standard for Future Airborne Capability Environment (FACE™), Edition 2.1”. Burlington, MA: The Open Group.