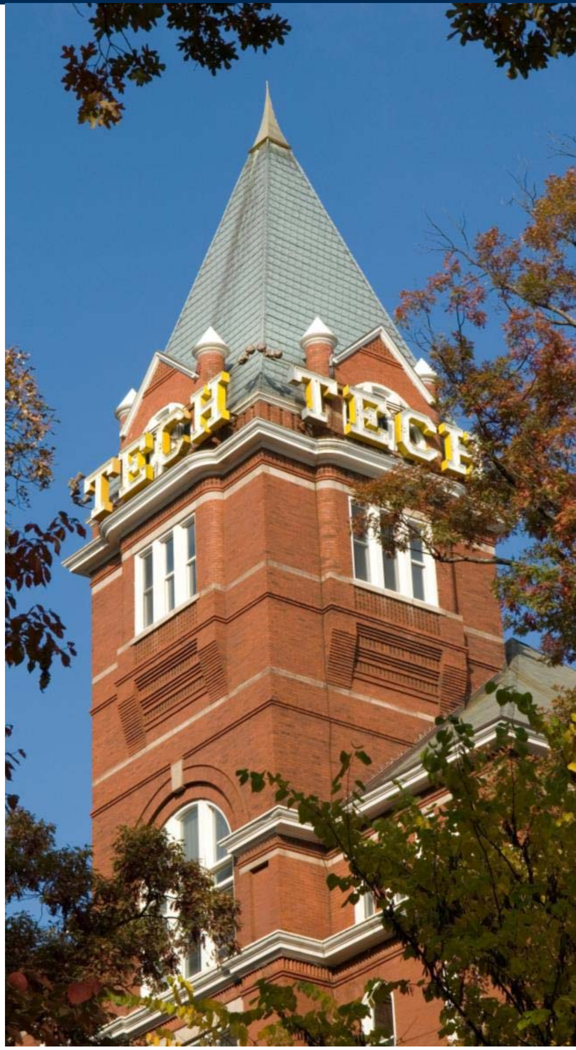
Sensor Intelligence & Visualization

Georgia Tech Research Institute

A Graph and Model-Based Analysis of the Openness of Functional Reference Architectures for Modular Open Systems



NDIA 22nd Annual Systems & Mission Engineering Conference

October 23, 2019

Jean Charles Domercant, Ph.D.

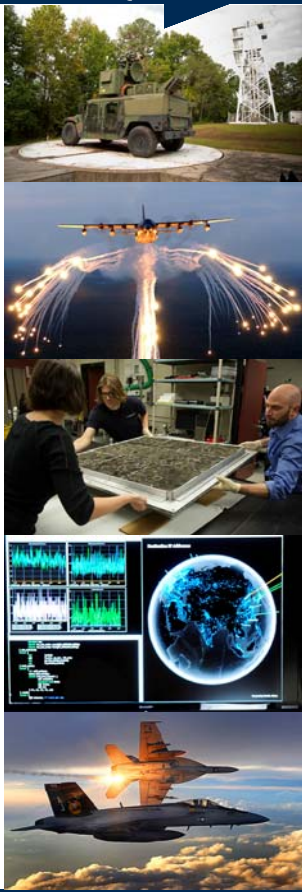
Erika Brimhall

Daniel Cooksey, Ph.D.

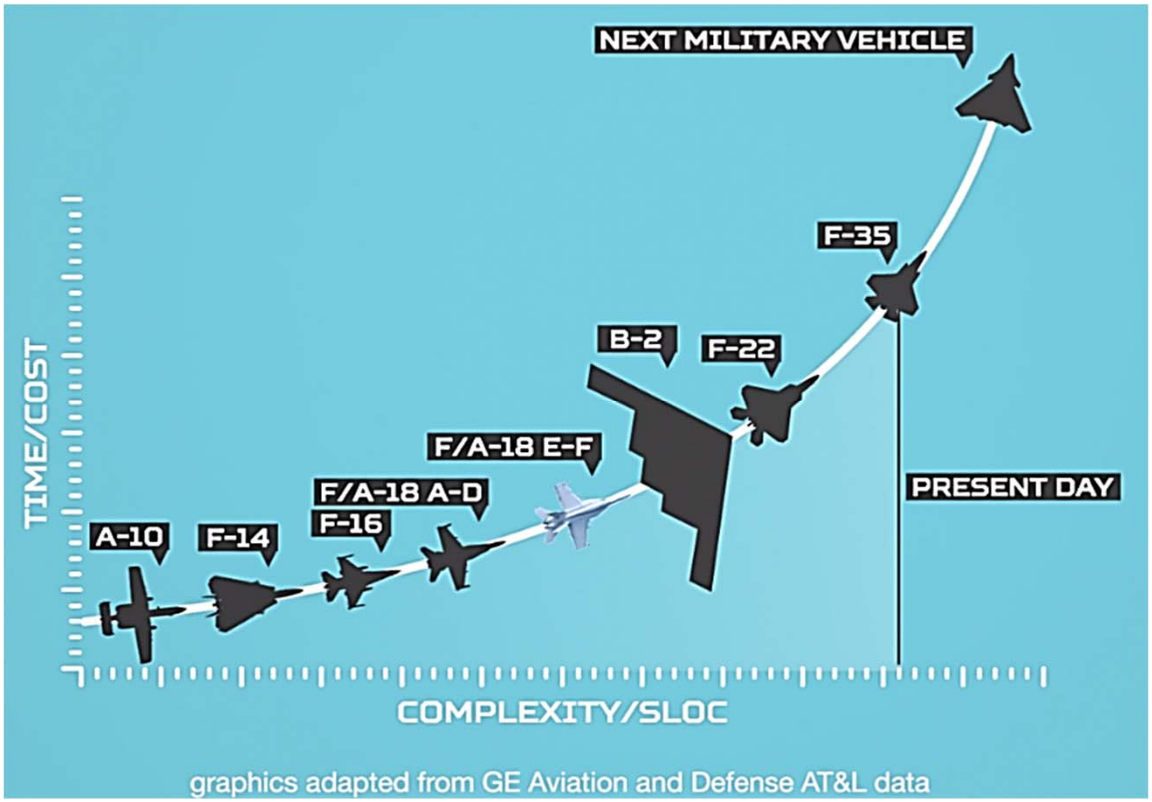
Georgia Tech Research Institute

Overview

- **Background & Motivation**
  - Increasing System Cost & Complexity
  - Modular Open Systems Architectures
  - Defining Severable Modules
- **Functional Reference Architectures**
  - Joint Common Architecture
  - Functional Architecture for STrategic Reuse – FASTR
- **How to Measure the Openness of a Functional Reference Architecture**
  - Metrics
  - Tool Development



Motivation – Increasing System Cost & Complexity



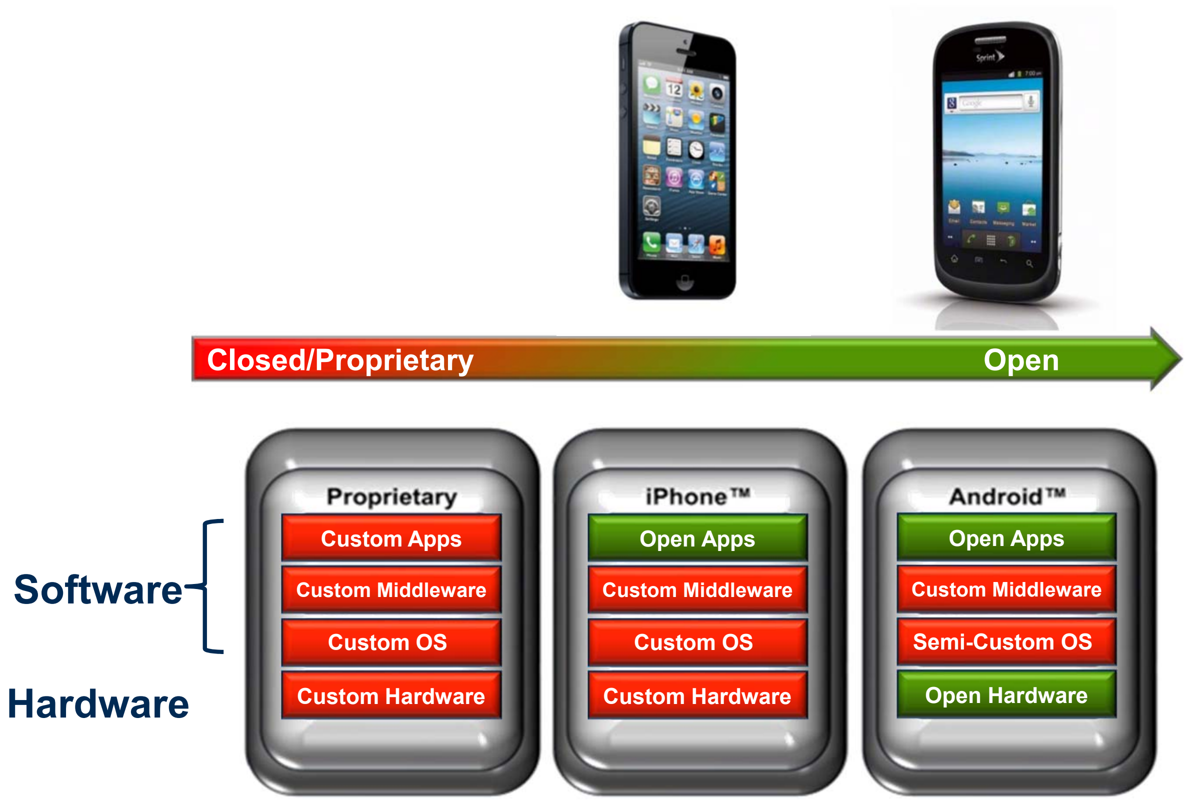
"In the year 2054, the entire defense budget will purchase just one aircraft.

This aircraft will have to be shared by the Air Force and Navy 3-1/2 days each per week except for leap year, when it will be made available to the Marines for the extra day."

Norman R Augustine,
Former Chairman/CEO, Lockheed Martin - 1984

Open vs. Closed Systems

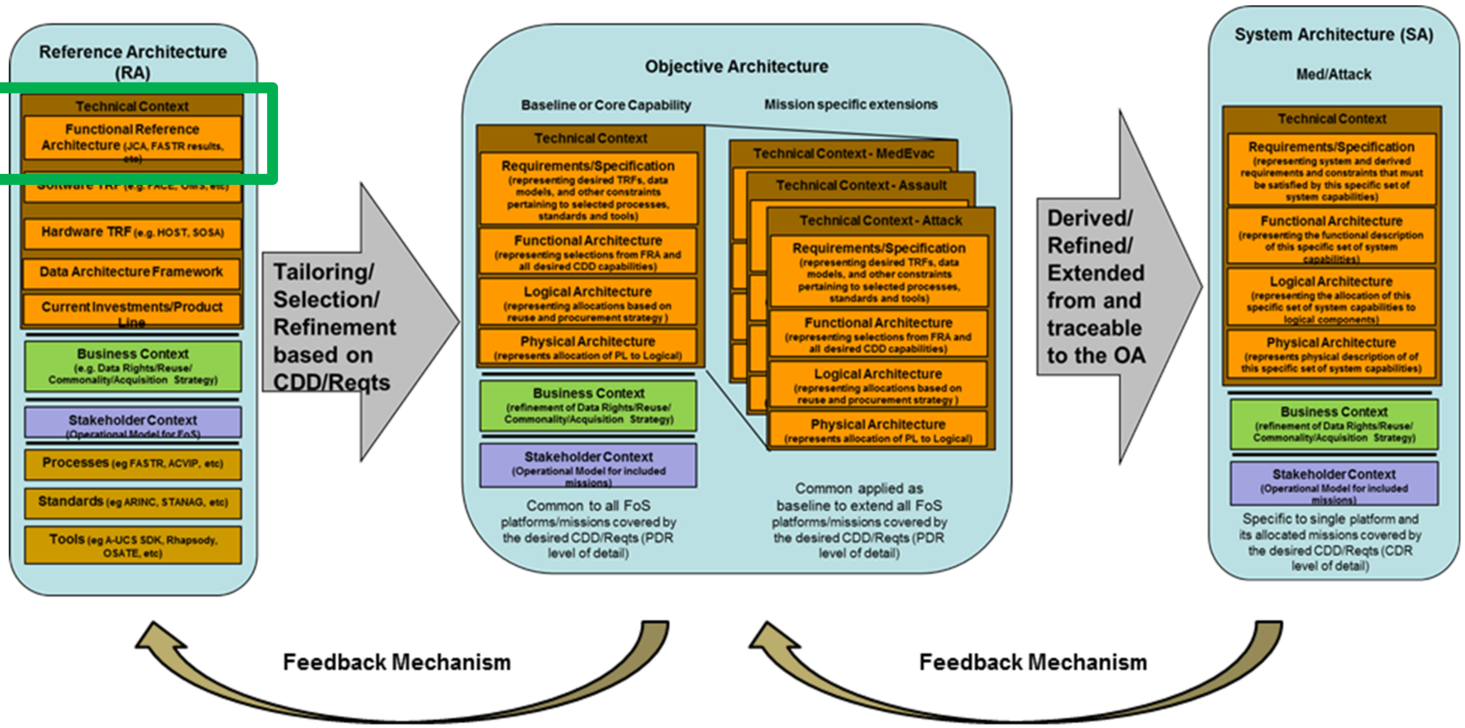
- Modular Open Systems Approach (MOSA)
- Both a business and technical strategy
  - Defines key interfaces
  - Uses consensus-based standards
- MOSA is not necessarily an “all or nothing approach” and the degree of openness can vary based on the modularity of the design.



Implementing MOSA Using a Common Architecture

Creating the Next

Example of a Common Architecture



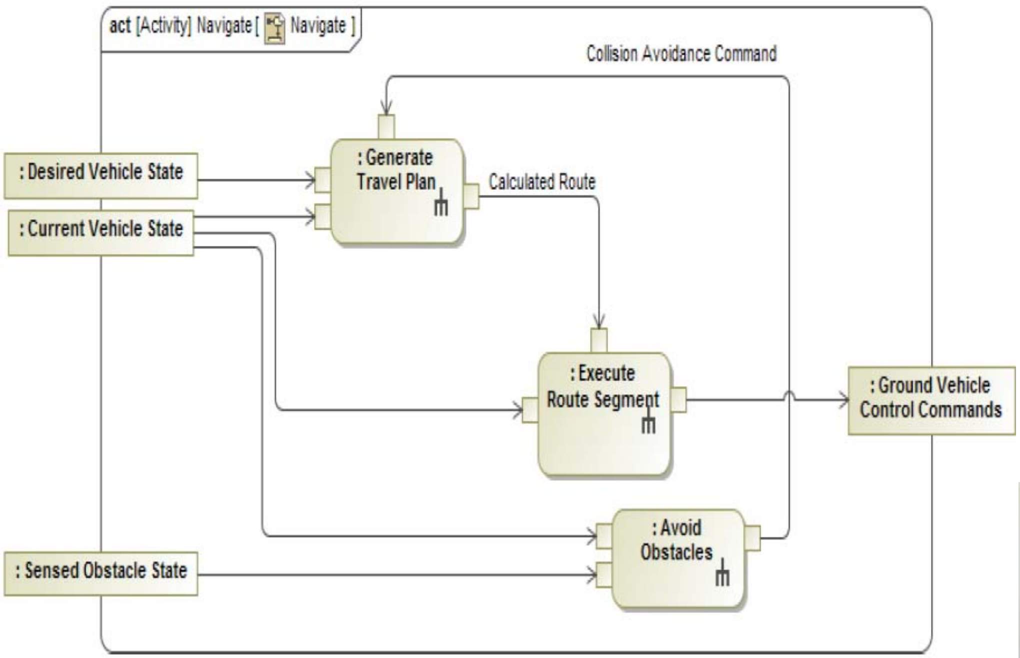
Joint Multi-Role Technology Demonstrator (JMR TD)Mission Systems Architecture Demonstration (MSAD) Capstone Demonstration Overarching Broad Agency Agreement (BAA), 2018

Functional Reference Architecture (FRA)

Creating the Next

- Intended to be used as a template
  - Objective Architectures: Aid in defining functionality for a family of systems
  - System Architectures: Specific to a single platform and its allocated missions
- Provides consistent, full coverage of the system functionality
  - System and implementation agnostic
  - Result is generic and reusable functions
  - Forms the basis for severable modules

Example Navigate Functional Activity



MH-60R


Image Credit: Naval Technology

HC-130


Image Credit: By Paul D. Charron

Future Vertical Lift

Image Credit: AVX Aircraft

Example FRA

Creating the Next

- **Joint Common Architecture (US Army Research, Development, and Engineering Command)**
  - Intended to define Reusable Software Components that reside on the mission computers of the Future Vertical Lift (FVL) fleet
  - Government-owned, implementation, and technology-independent conceptual framework
  - Provides a conceptual description of a set of generic avionics subsystems
  - Also provides a functionally decomposed mission computing subsystem comprising a functional model and a semantic data model

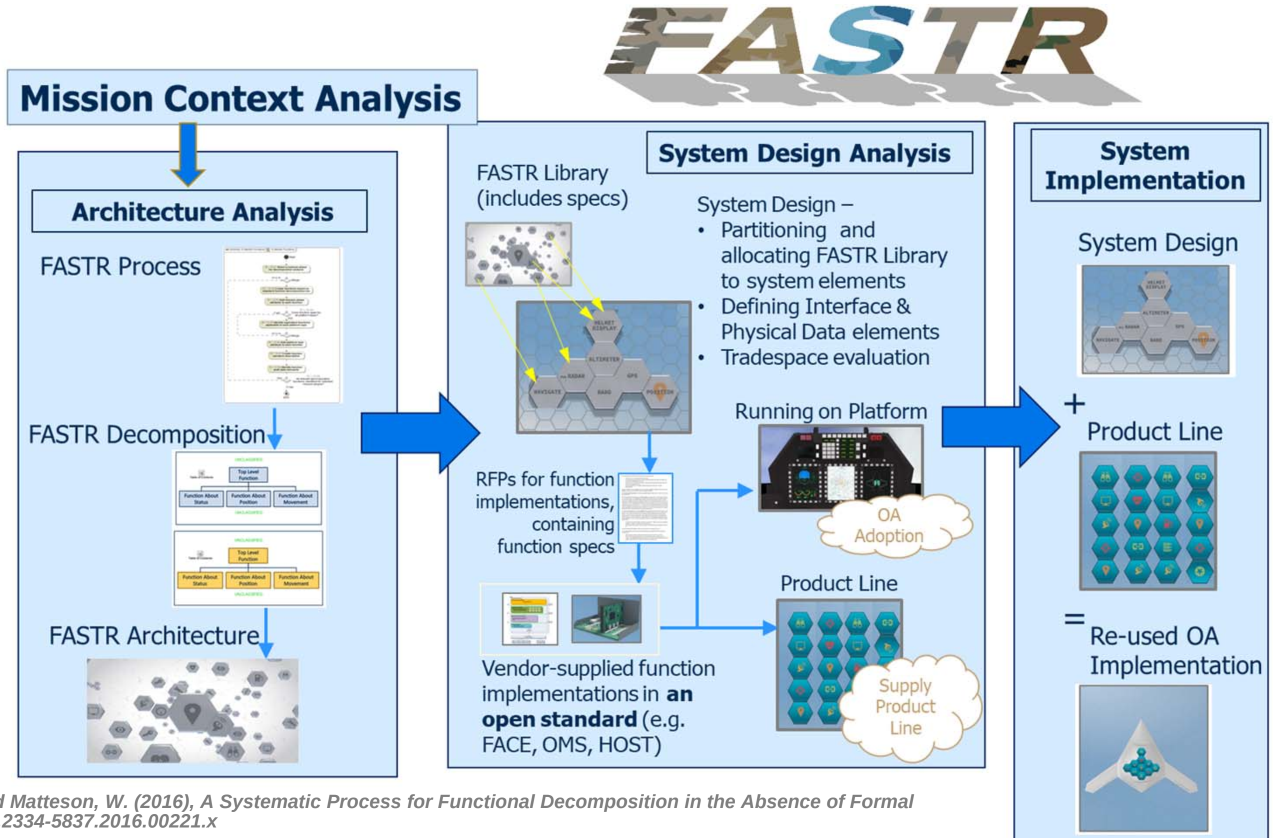
Example Future Vertical Lift Concepts

Image Credit: AVX Aircraft

Wigginton, Scott A., Joint Common Architecture Demonstration (JCA Demo) Final Report. TECHNICAL REPORT RDMR-AD-16-01. U.S. Army Research, Development, and Engineering Command, July 2016.

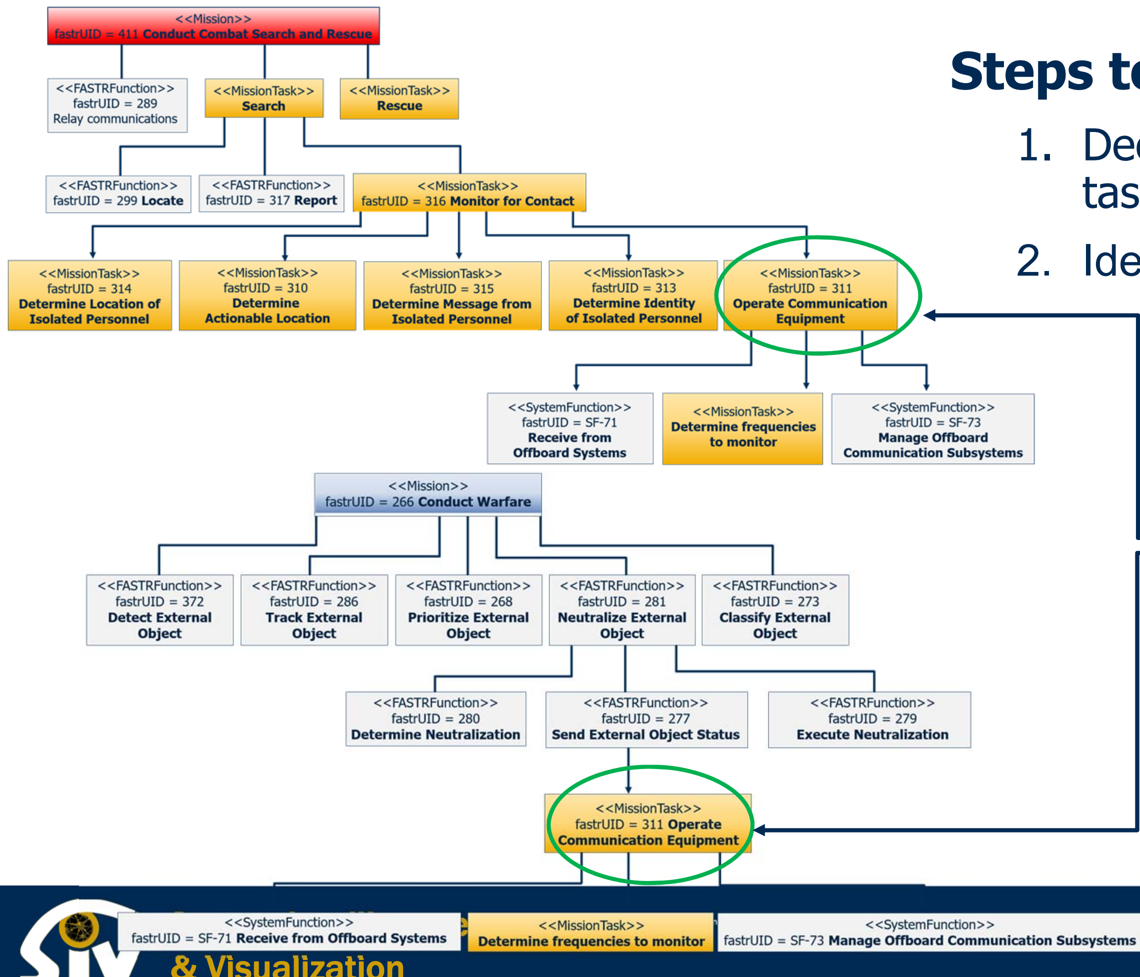
Functional Architecture for STrategic Reuse – FASTR

- FASTR is a systematic approach for developing implementation-agnostic functional decompositions
- Approach works even in the absence of formal requirements
- Includes Model Based Systems Engineering decomposition and recomposition processes and tools
- Supports the creation of a FRA and severable modules to support MOSA



INCOSE SYMPOSIUM PAPER: Brimhall, E., Wise, R., Simko, R., Huggins, J. and Matteson, W. (2016), A Systematic Process for Functional Decomposition in the Absence of Formal Requirements. INCOSE International Symposium, 26: 1204–1218. doi:10.1002/j.2334-5837.2016.00221.x

Defining Severable Modules



Steps to Defining Severable Modules:

1. Decompose Mission Threads to lower-level tasks and functions
2. Identify common tasks/functions

"Operate Comms. Equipment" appears in both mission threads

Defining Severable Modules

<<MissionTask>>
fastrUID = 311
Operate Communication Equipment

MH-60R

Image Credit: Naval Technology

HC-130

Image Credit: By Paul D. Charron

Steps to Defining Severable Modules:

1. Decompose Mission Threads to lower-level tasks and functions
2. Identify common tasks/functions
3. Define a portable Software/Hardware Product
   - Meets the data requirements
   - Can be reused for different platforms instead of re-implemented

Severable Modules Enable Functional Reuse

Functional Decomposition

Result of this process: develop a common function library that is system and implementation agnostic

Functional Reuse

Integration of functions into systems to meet design requirements

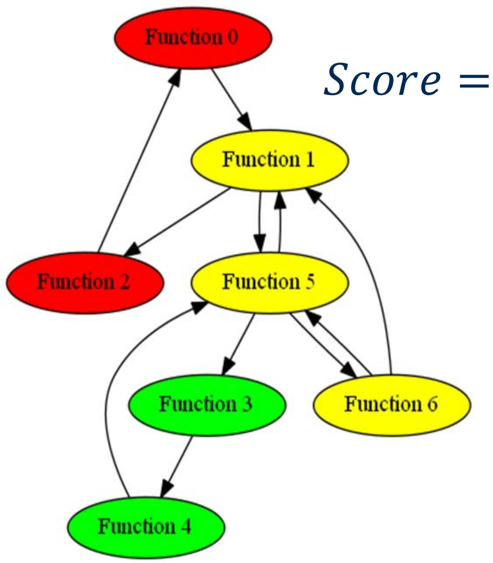
Measuring the Openness of a FRA

- The “Openness” of a FRA must be assessed
- 1) How well does the FRA supports the development of severable modules?
  - Quantitatively assess the level of modularity that can be achieved using the resulting functional decomposition
- 2) What is the quality of the FRA in terms of the following?
  - Consistency and completeness of the function definitions and data elements
  - Are the functions specified to the correct level of abstraction?
  - Etc.
- A FRA was developed using FASTR for Aviate, Navigate, Communicate (ANC) aircraft functions – FASTR ANC FRA

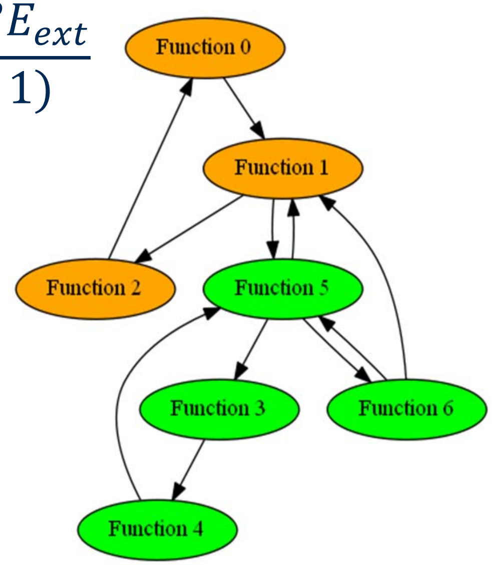
A FRA is an integral part of implementing MOSA, thus, it is crucial to develop metrics to assess how well the FRA supports an open approach.

1) Modularization Scoring

$$Score = \sum_{m=1}^{M} \frac{E_{int} - \beta E_{ext}}{n(n-1)}$$



Genetic Algorithm-derived modularization scheme with low cost penalty for external module-to-module connections



Genetic Algorithm-derived modularization scheme with high cost penalty for external module-to-module connections

2) Function Inputs/Outputs

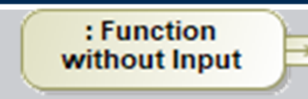
Creating the Next

Metrics to Minimize

Functions without Inputs

Description: Functions that do not have any inputs.

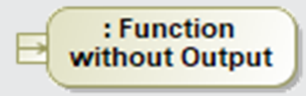
Justification: If there's no input, the function is generating something from nothing.



Functions without Outputs

Description: Functions that do not have any outputs.

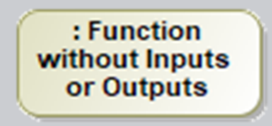
Justification: If there's no output, there's no need for the function.



Functions without Inputs or Outputs

Description: Functions that have neither inputs nor outputs.

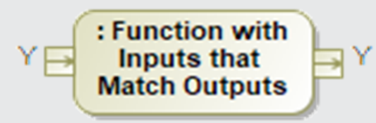
Justification: If there's no input, the function is generating something from nothing.
If there's no output, there's no need for the function.



Functions with Inputs that Match Outputs

Description: Functions with either no I/O at least one input that matches at least one output.

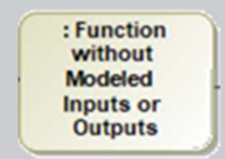
Justification: Functions transform inputs into outputs.
If there's no transformation, there's no need for the function.



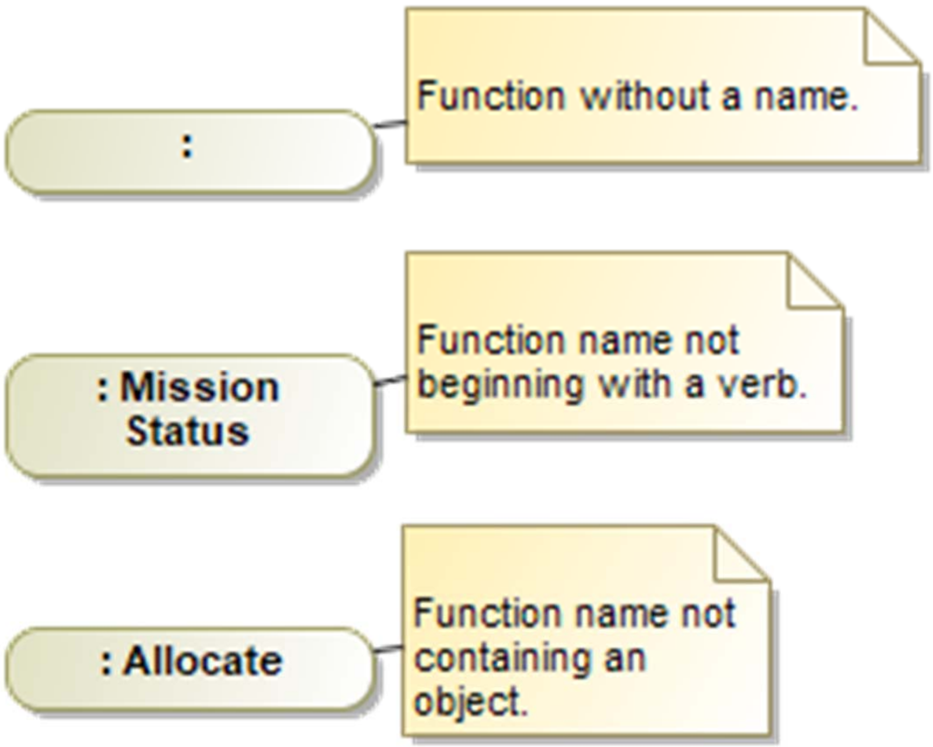
Functions without Modeled Inputs or Outputs

Description: Functions with either no I/O or I/O as text or graphically depicted but no modeled relationship between the functions and the data. Note that this is NOT the same as functions that use data elements not data modeled.

Justification: If the inputs or outputs aren't modeled, there's no point of modeling the function.



2) Function Names



Metrics to Minimize

Functions Not Named

Description: Functions with a blank name or a meaningless placeholder name.

Justification: Function names help humans and analysis software understand the main intent of a function.

Functions Names Not Beginning with Verb

Description: Functions with names that do not begin with a verb.

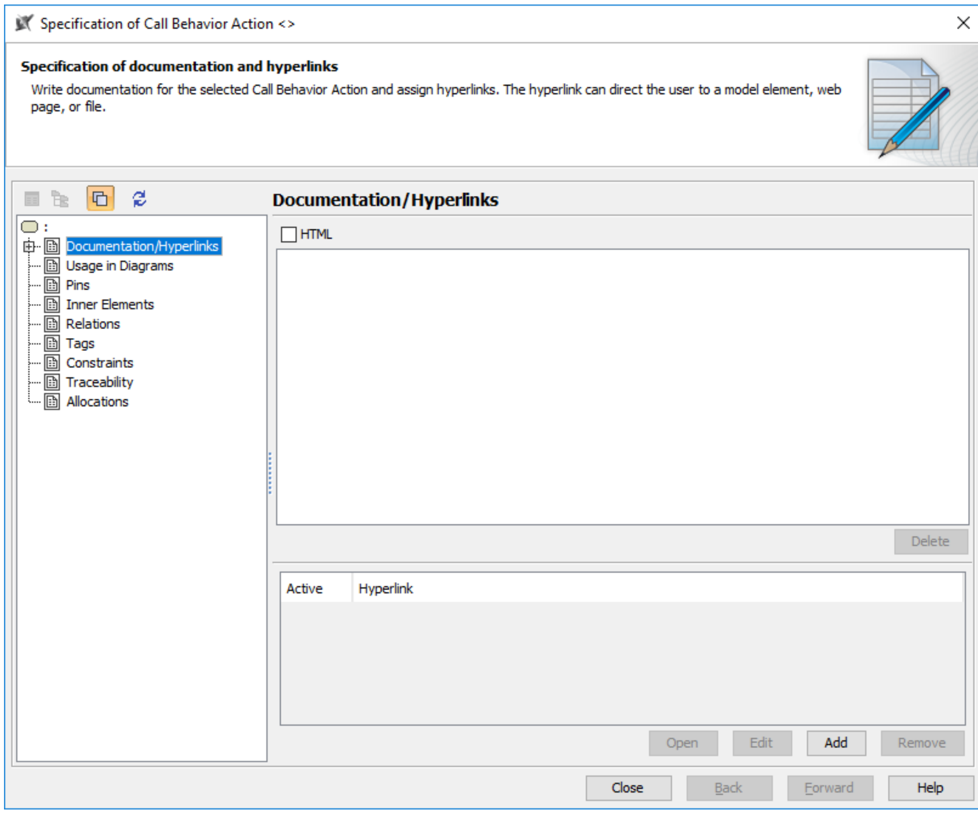
Justification: Functions transform inputs into outputs by doing something; a verb helps capture what the function is doing to accomplish that.

Functions Names Not Containing an Object

Description: Functions with names that do not contain an object.

Justification: If it’s unclear what the verb is acting upon, the name is insufficient for communicating what a function does.

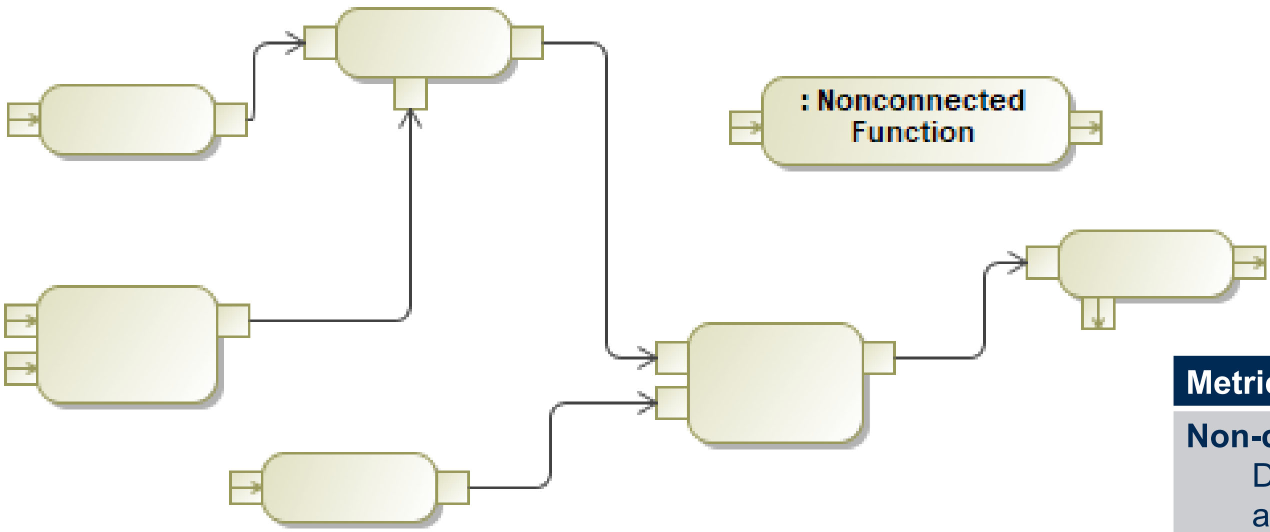
2) Function Documentation



Function not documented.

Metrics to Minimize
Functions Not Documented Description: Functions without descriptions. Justification: Functions without descriptions are missing an important way to convey their functionality to humans and analysis software.

2) Non-connected Functions



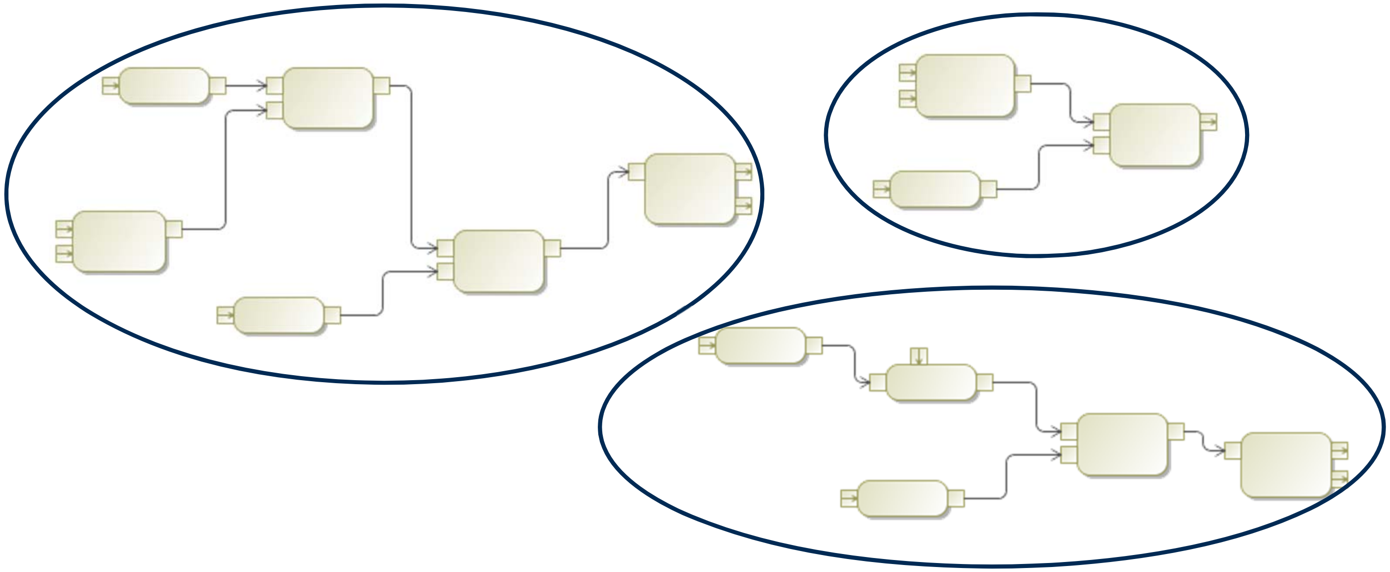
Metrics to Minimize

Non-connected Functions

Description: Functions that do not connect to any other functions via data flow (i.e., the inputs and the outputs are not used by other functions)

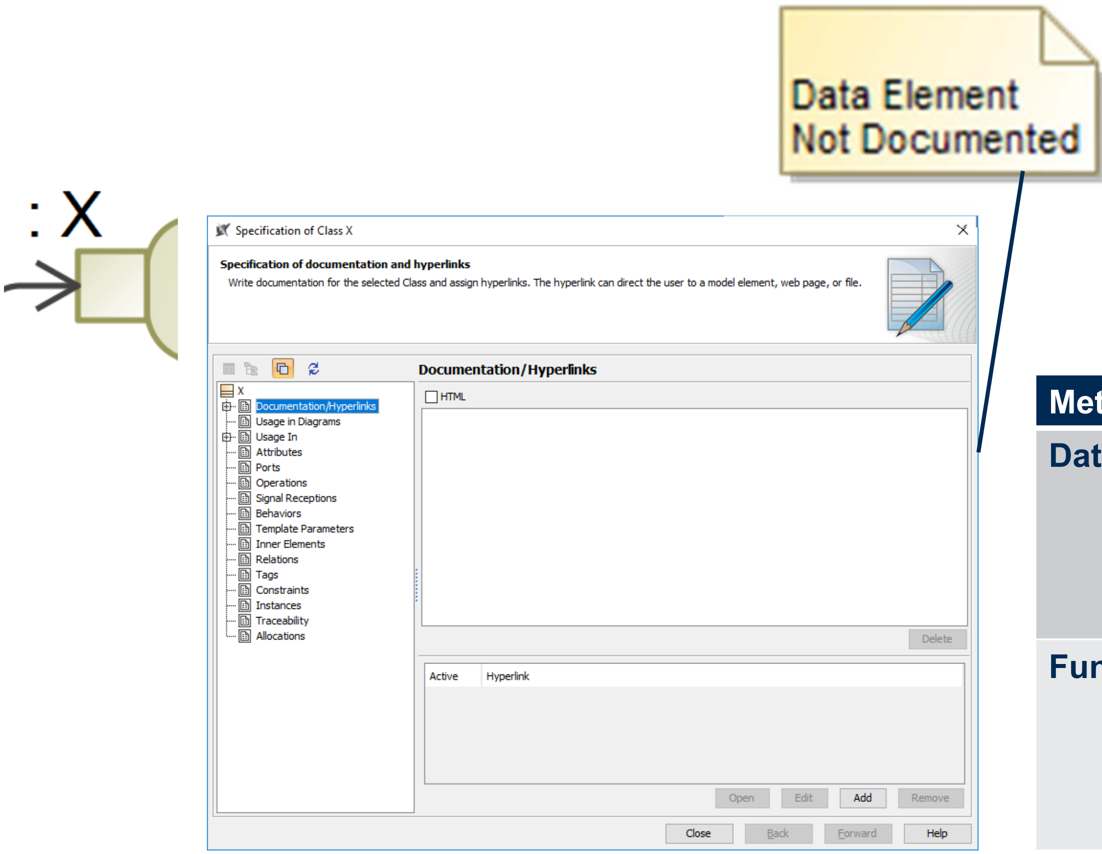
Justification: Modular functions are intended to connect to each other to serve a purpose. Functions that cannot connect to others are limited in their usefulness.

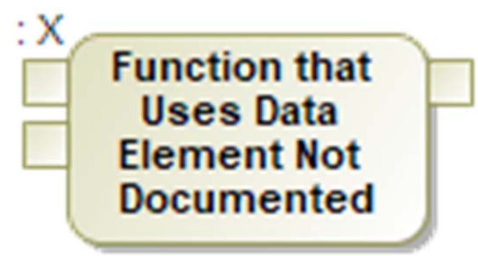
2) Distinct Function Networks



Metrics to Minimize
Distinct Networks of Atomic Functions Description: Number of separate networks formed by input/output connections between atomic functions. Justification: If there's no behavior path that can connect specific functions, there could be a gap in functionality.

2) Data Elements Not Documented





Metrics to Minimize

Data Elements Not Documented

Description: Data elements that do not have descriptions.

Justification: Descriptions are important for communicating the purpose/content of a data element.

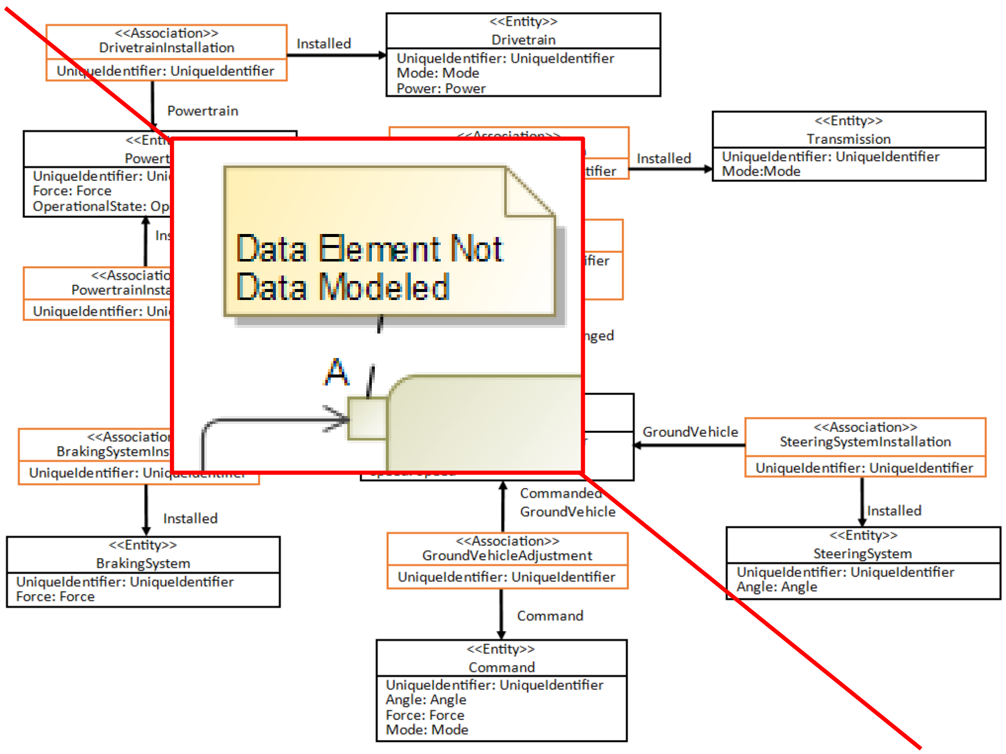
Functions that Use Data Elements Not Documented

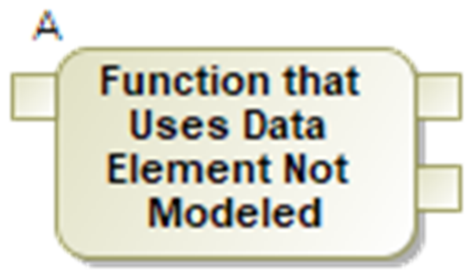
Description: Functions that use the data elements that do not have descriptions.

Justification: Descriptions are important for communicating the purpose/content of a data element.

2) Data Elements Not Data Modeled

Creating the Next





Metrics to Minimize

Data Elements Not Modeled

Description: Data elements that are not data modeled using some formal language.

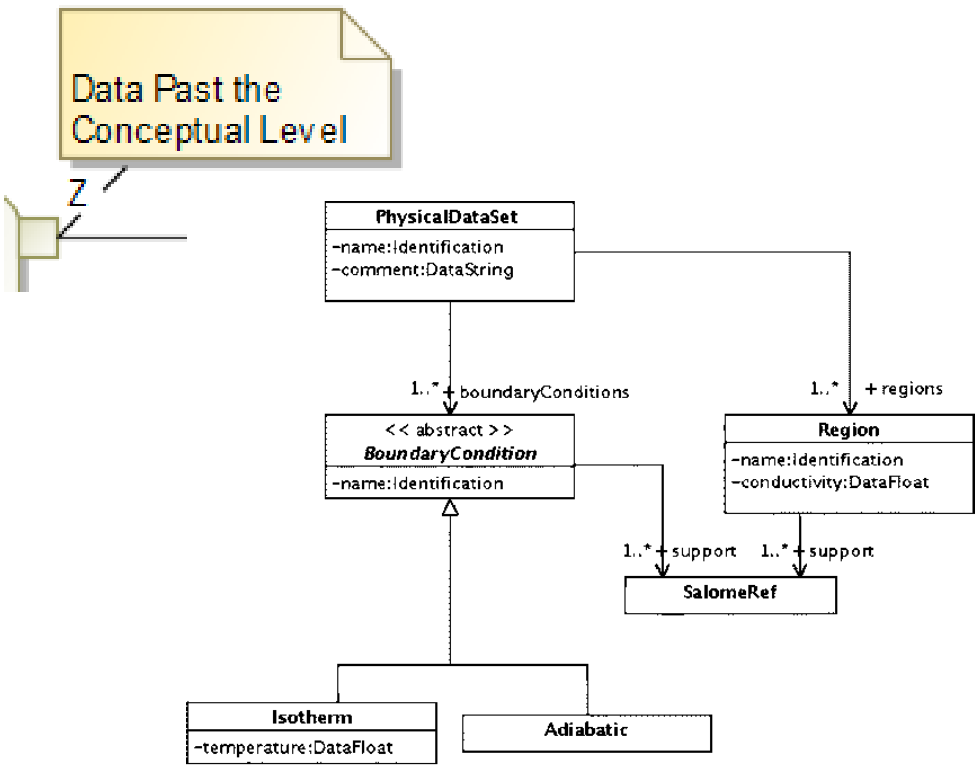
Justification: Formal data modeling reduces ambiguity of interpretation and allows automated analysis.

Functions that Use Data Elements Not Data Modeled

Description: Functions that use data elements that are not data modeled using some formal language.

Justification: Formal data modeling reduces ambiguity of interpretation and allows automated analysis.

2) Over-specified Data



Metrics to Minimize

Data Elements Beyond Conceptual Level (Over-specified)

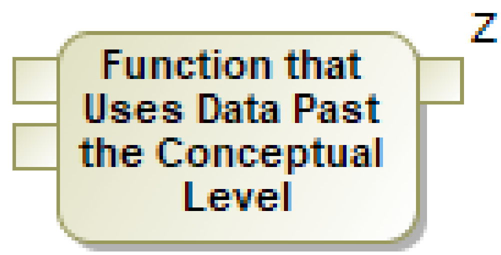
Description: Data elements that use non-conceptual concepts such as a frame of reference or units.

Justification: Over-specifying data elements makes them less portable/modular.

Functions that Use Data Elements Beyond Conceptual Level (Over-specified)

Description: Functions that use data elements that are non-conceptual.

Justification: Over-specifying data elements makes their associated functions less portable/modular.



Metrics for FASTR ANC FRA

Creating the Next

Metric	Total	Percent
Total Functions	**100**	
Functions without Inputs	1	1%
Functions without Outputs	1	1%
Functions without Inputs or Outputs	1	1%
Functions with Inputs that Match Outputs	1	1%
Functions without Modeled Inputs or Outputs	1	1%
Nonconnected Functions	16	16%
Functions Not Documented	0	0%
Functions Not Named	0	0%
Functions Names Not Beginning with Verb	0	0%
Functions Names Not Containing an Object	3	3%

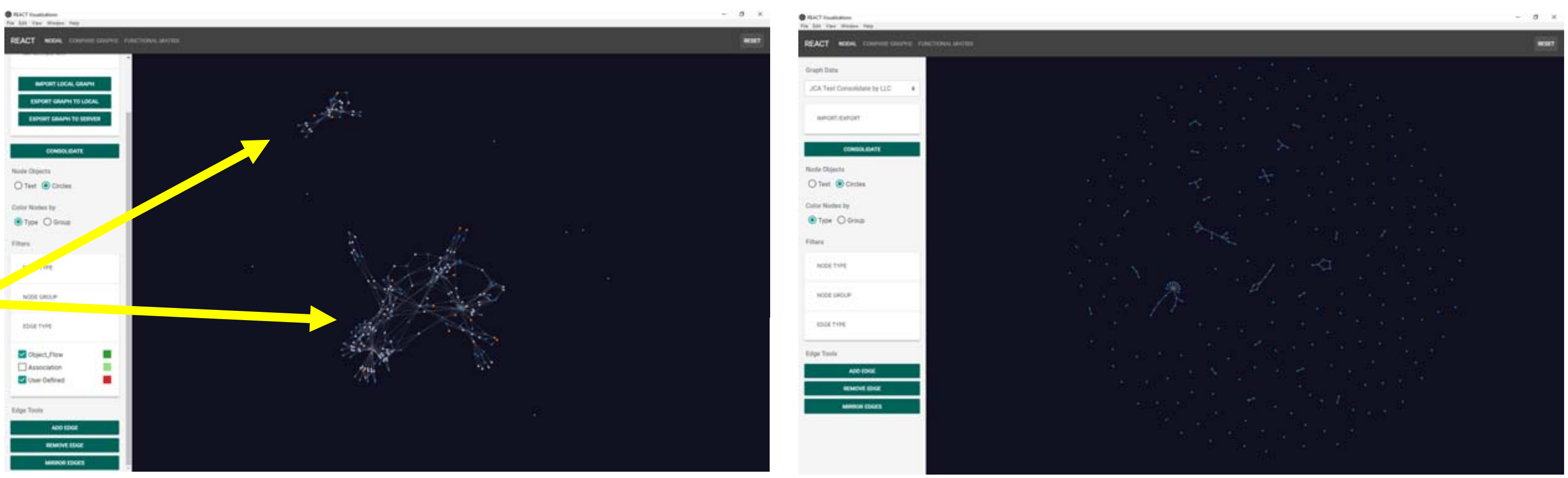
Metric	Total	Percent
Total Atomic Functions	**70**	
Atomic Functions without Inputs	0	0%
Atomic Functions without Outputs	0	0%
Atomic Functions without Inputs or Outputs	0	0%
Functions with Inputs that Match Outputs	0	0%
Atomic Functions without Modeled Inputs or Outputs	0	0%
Atomic Nonconnected Functions	14	20%
Atomic Functions Not Documented	0	0%
Functions Not Named	0	0%
Atomic Functions Names Not Beginning with Verb	0	0%
Atomic Functions Names Not Containing an Object	0	0%
Distinct Function Graphs of Atomic Behavior	9	13%

Metric	Total	Percent
Total Data Elements	**128**	
Data Elements Not Documented	0	0%
Functions that Use Data Elements Not Documented	0	0%
Atomic Functions that Use Data Elements Not Documented	0	0%
Data Elements Not Modeled	14	11%
Functions that Use Data Elements Not Data Modeled	31	31%
Atomic Functions that Use Data Elements Not Data Modeled	13	19%
Data Elements Beyond Conceptual Level (Overspecified)	0	0%
Atomic Functions that Use Data Elements Beyond Conceptual Level	0	0%

Tool Development

- FRAs can include hundreds of functions and data elements
- ARC tool developed to aid in the analysis of FRAs – Graph Based Analysis Approach
- Thorough analysis of FRA Openness can transform FRAs from templates to true analytical tools.

Distinct Networks of Atomic Functions



Comparing the Connectedness of Different FRAs

Summary

- System complexity and cost continues to increase at a rapid pace to meet desired capability needs
- A Modular Open Systems Approach (MOSA) is necessary to combat this negative trend
- The degree of Openness can vary based on the modularity of the design, and Functional Reference Architectures form the basis for severable modules
- Two approaches and corresponding metrics presented to assess how well a FRA supports Openness

Thank You!

?