Using Architecture and MBSE to Develop Validated Requirements

Ron Carson, PhD, ESEP, INCOSE Fellow

Seattle Pacific University, University of Washington
https://www.linkedin.com/in/ron-carson-phd-esep-573549b/

2020 INCOSE Western States Regional Conference – Seattle, WA

This began as a question from INCOSE Requirements Working Group

- How can/should we represent requirements in MBSE?
  - Functional requirements
  - “Non-functional” requirements
- Requirements incompleteness and ambiguity continue to plague many organizations. The introduction of MBSE provides an opportunity to relate the structure of the architecture model to the structure of requirements, and synchronize the data between them.
- References and sources:

  - Carson, Ronald S., “Implementing Structured Requirements to Improve Requirements Quality”, Proceedings of INCOSE 2015
  - Carson, Ronald S, and Robert A. Noel, “Formalizing Requirements Verification and Validation”, Proceedings of INCOSE 2018
  - Carson, Ronald S., “Using System Architecture Models to Populate Structured Requirements”, MBSE Lightning Round Presentation, INCOSE IS 2019
    IS2019 MBSE Lightning Round - Ron Carson – YouTube
  - Presentation to INCOSE RWG, 24 October 2019:
    https://connect.incose.org/WorkingGroups/Requirements/RWGMeetings/SiteAssets/SitePages/Home/RWG+10-24-19-2019-10-24T19_59_44.000Z.mp4
  - Carson, Ronald S., et al., “Structured Requirements Generation and Assessment”, US Patent #8,886,588, November 2014.

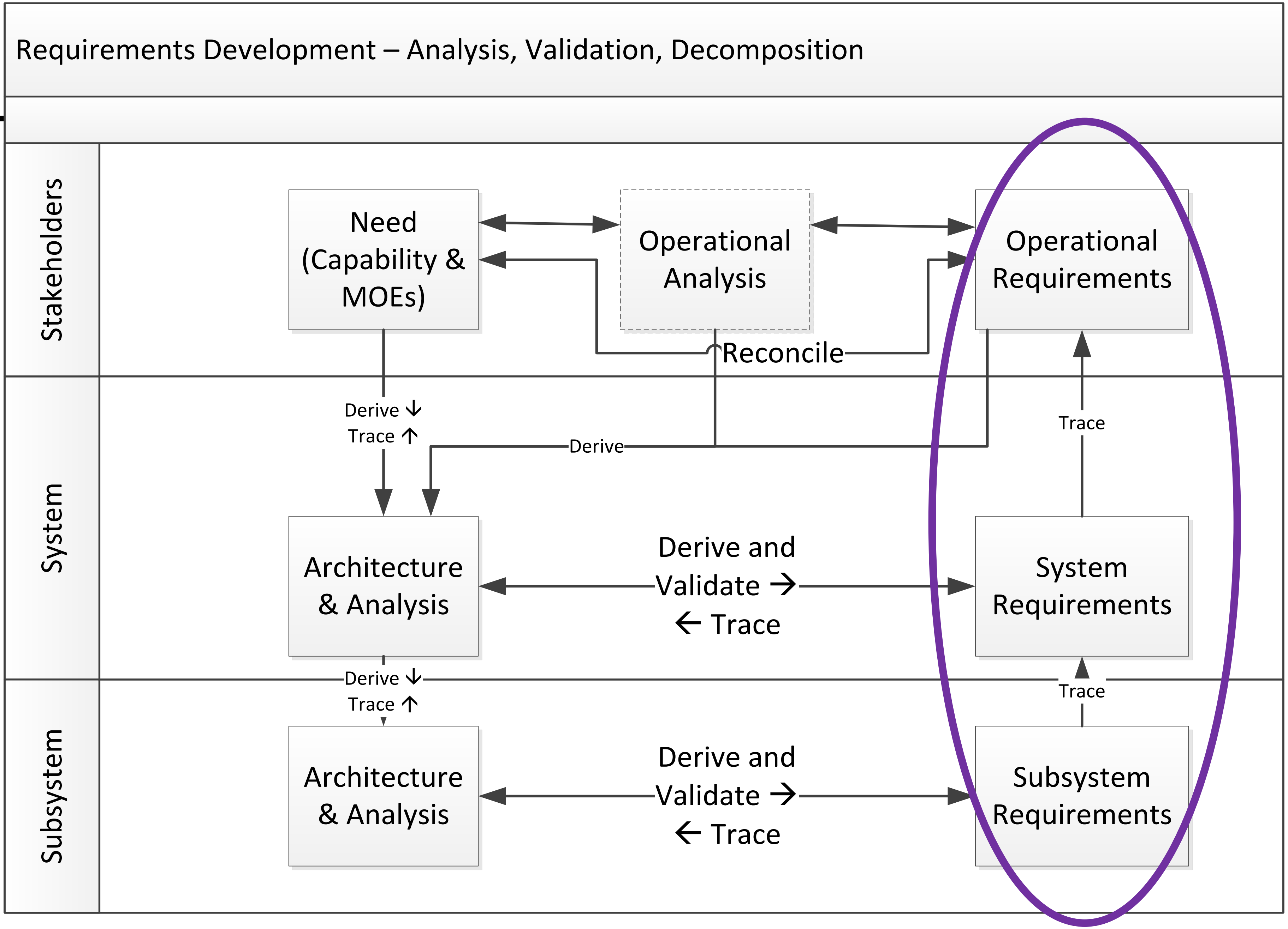
Outline

- The bases of requirements: from where do requirements arise?
- What do we mean by "Validated"?

1. Basic requirements structure
2. Types of requirements and their structures and data elements
3. Architecture sources of data for structured requirements, by type
   – Example Implementation in CORE
4. Simulation for Requirements Validation
5. Architecture level recursion and design
6. Summary

Why do this?

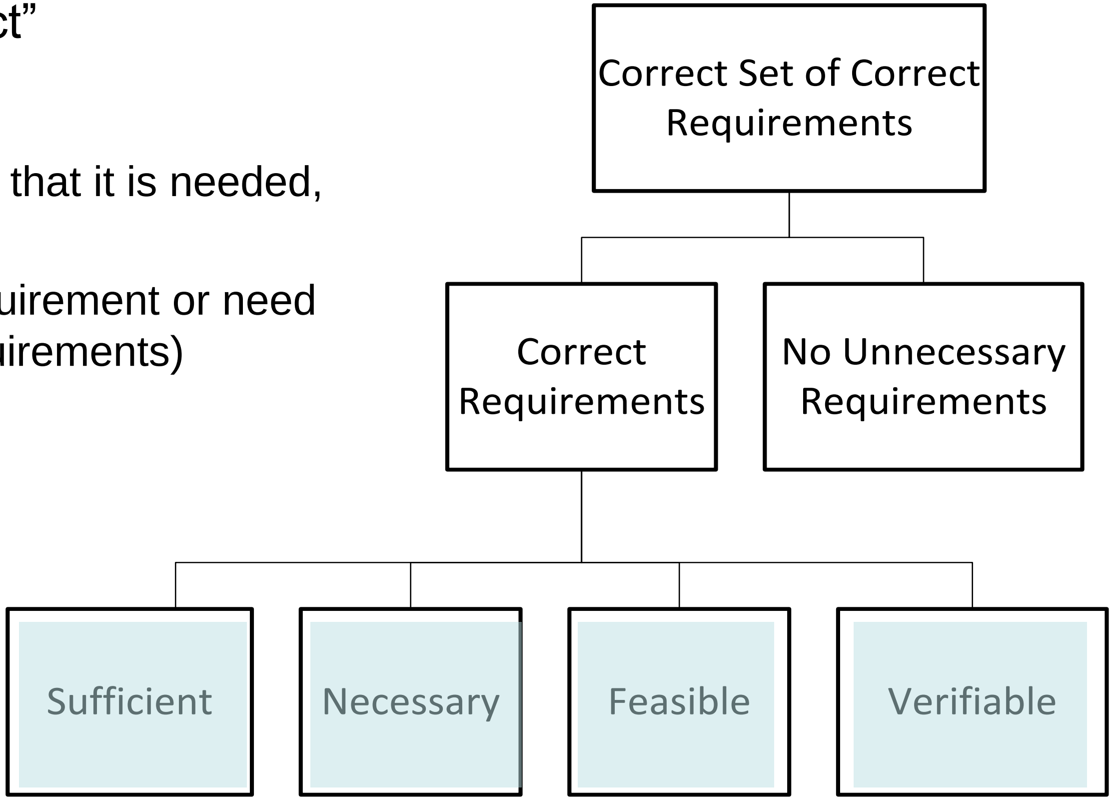
- Because requirements come *after* analysis*
- Then *analysis* can be the source of information for *requirements*



- *Saturday tutorial, "Correcting Misperceptions of Systems Engineering Practices"

“Validated Requirements” means “Correct Set of Correct Requirements*

- *Each* requirement and the *set* of requirements are “correct”
- Correct (type 2): no missing information
  - Verifiable: sufficiently clear and complete to be able to prove that it is needed, sufficient, and feasible
  - Sufficient: if this requirement is satisfied, then the parent requirement or need *will be satisfied* (perhaps in combination with other child requirements)
- Correct (type 1): no erroneous information
  - Necessary: parent requirement or need *cannot* be satisfied without this requirement
    - No unnecessary requirements
  - Feasible: requirement can be satisfied within the program constraints with acceptable risk
- These characteristics are disjoint, concise, and complete (necessary and sufficient to determine “correct and complete”)
- From these we realize a “Correct Set of the Correct Requirements”



*Carson and Noel, INCOSE 2018

1. Structured Requirements

Basic structure:

The *who* shall *what*, *how well*, *under what conditions.*

- INCOSE RWG Guide, ISO 29148
  - The <subject clause> shall <action verb clause> <object clause> <optional qualifying clause>, when <condition clause>.'   [INCOSE Guide to Writing Requirements]

- These documents do not address "non-functional" requirements, e.g., suitability, design, environments

- ISSUES:
  - Can we define structured requirements for "non-functional" requirements and instantiate them in MBSE?
  - Can we find data elements in the architecture for these "non-functional" requirements?

2. Identify Types of Requirements – Boeing (Carson 2015)

The *who* shall *what*, *how well*, *under what conditions.*

- **Functional/Performance**: mission-oriented characteristics; includes “interface requirements” regarding inputs/outputs.
- **Design**: constraints on solution: parts, materials, processes, physical allocations (size, weight, power); includes interface requirements regarding implementation constraints.
- **Suitability**: non-mission-specific characteristics addressing fitness for use (safety, “ilities”, transportation, storage)
- **Environment**: condition statements applicable to different states/modes and functions.

2. Define Structured Requirements based on their Data Elements

The *who* shall *what*, *how well*, *under what conditions.*

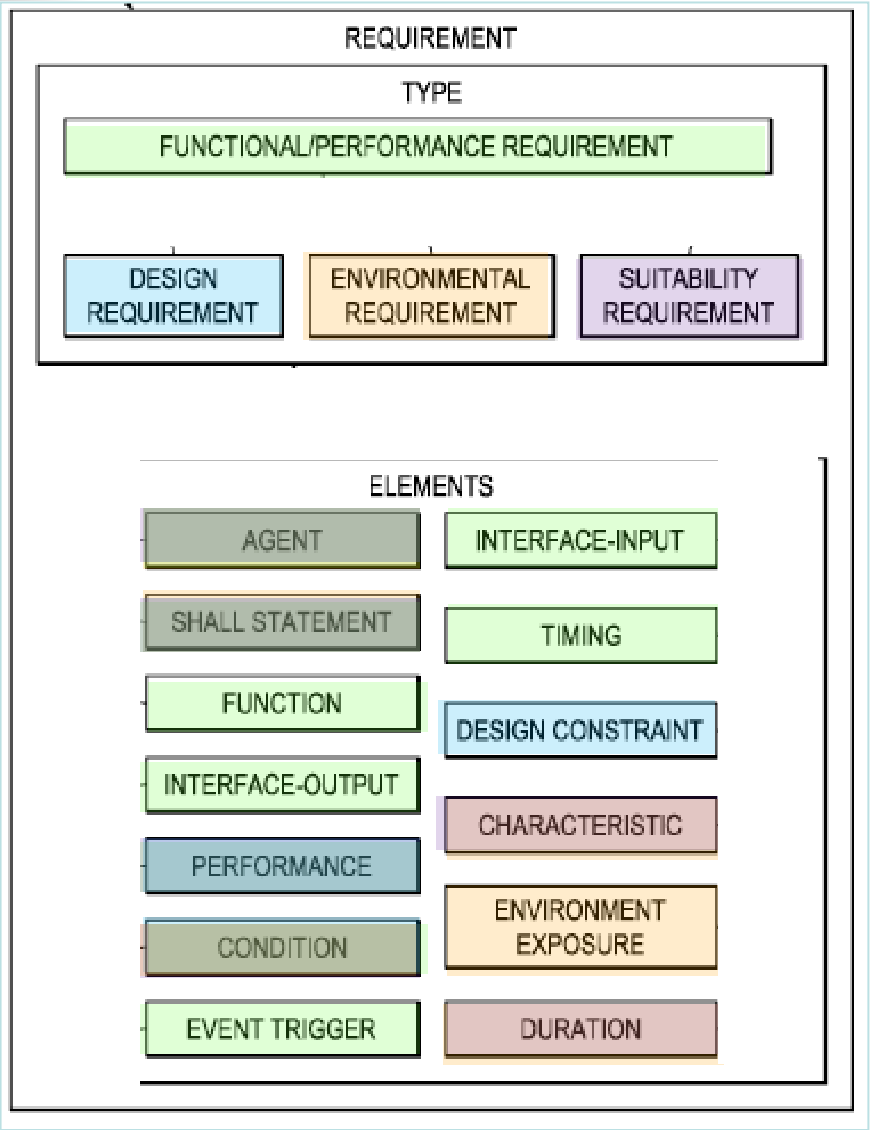
The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.
[Boeing: Carson INCOSE 2015]

The AGENT shall exhibit DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION].

The AGENT shall exhibit CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION].

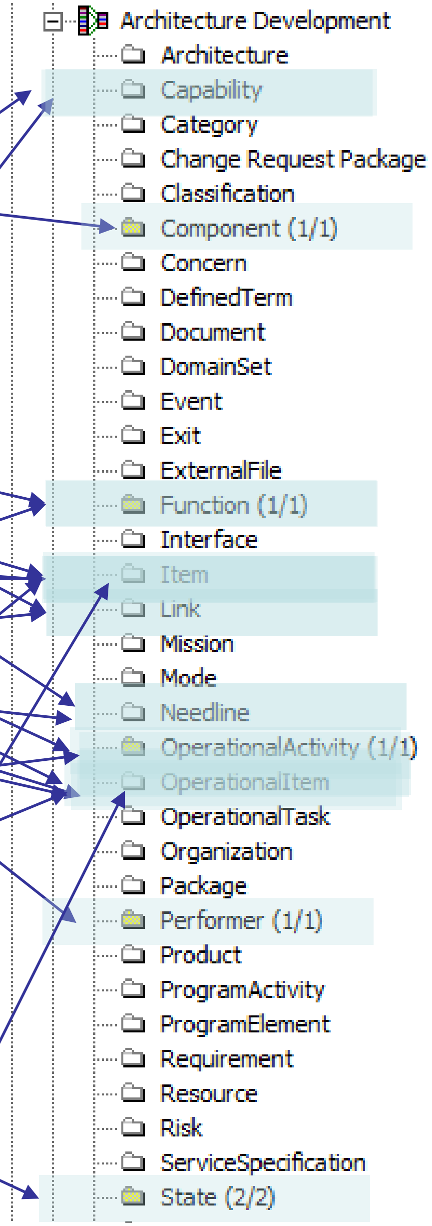
The AGENT shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION].

Note: these structures are not directly supported by SysML.



3. Map Structured Requirement Elements to MBSE Data Model – **Tool Specific**

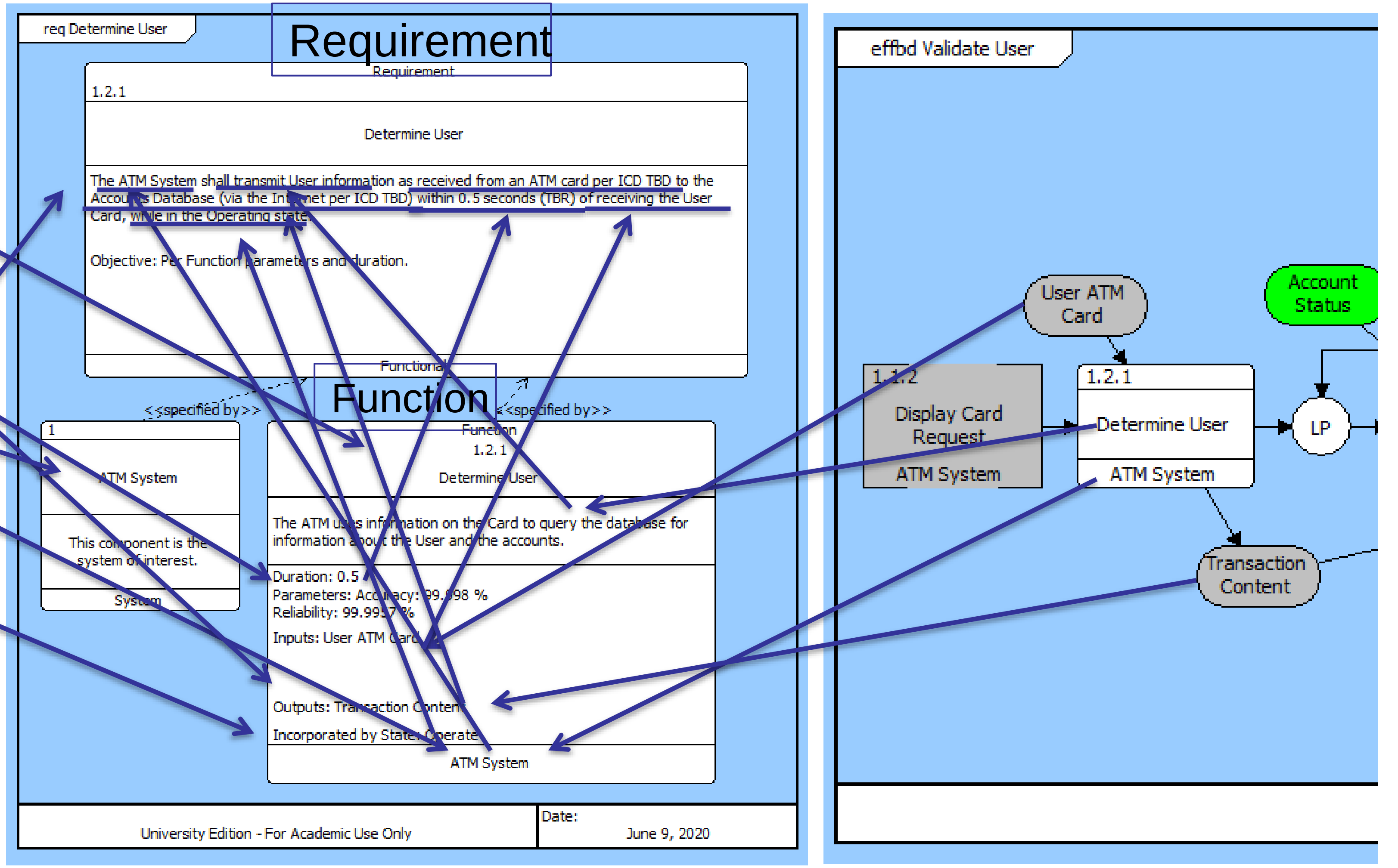
- Agent: the implementing element (noun)
- Shall: indicating a requirement – mandatory characteristic (no separate element)
- Function: observable action and output
- Interface-Output: constraint on observable manifestation of the Function
- Performance: Measurable attributes of observable Function [an attribute of an element – more later; SysML Parametric Diagram]
- Condition: states, environments, continuous input
- Event Trigger: Input condition initiating Function
- Interface-Input: constraint on observable manifestation of an Event Trigger
- Timing: Time for Function (duration, delay) – attribute of Function
- Design Constraint: Design characteristic for Design type [derived from MOE(Capability)] or Parametric Diagram
- Characteristic: Required attribute ("is" "has" or "does") for Suitability and Environment types - Function or MOE
- Environment Exposure: Relates Characteristic to the Environment (input) exposure as "during", "after" or "during and after"
- Duration: Time of Condition exposure for Suitability and Environment types – Attribute of Item or Operational Item



Vitech CORE Classes

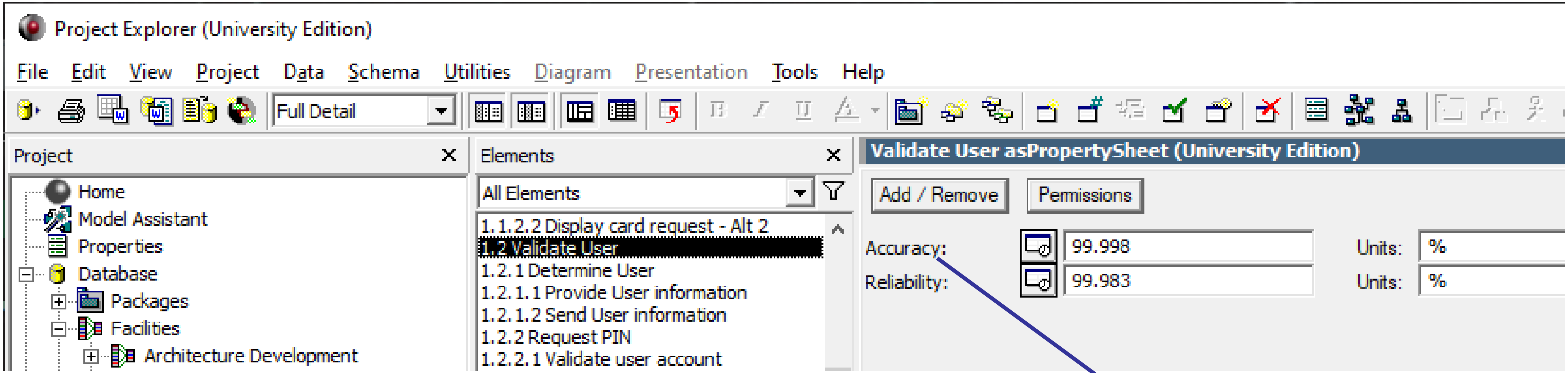
3. Mapping to Architecture Elements – Functional/Performance Requirements

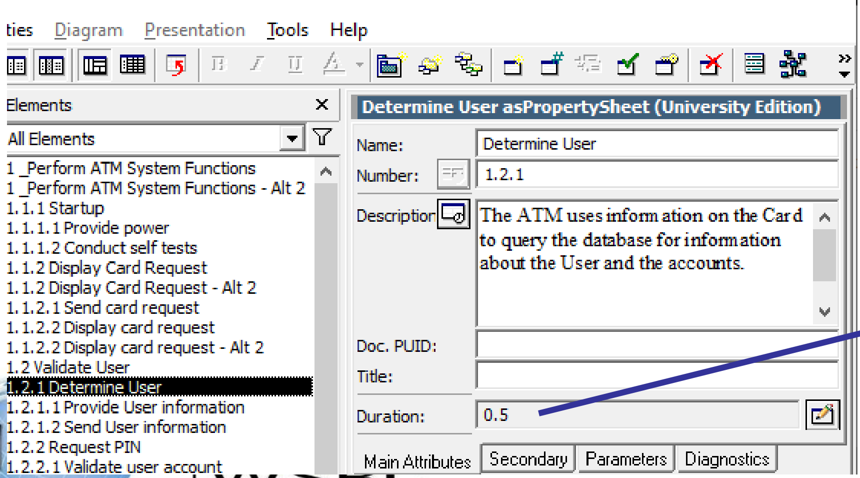
- Function
- Interfaces (inputs and outputs)
- Performance attributes
- Component
- Condition
- The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.



3. Performance Parameters – 1

- Measurable attributes of observable Function (“Parameter” in CORE)



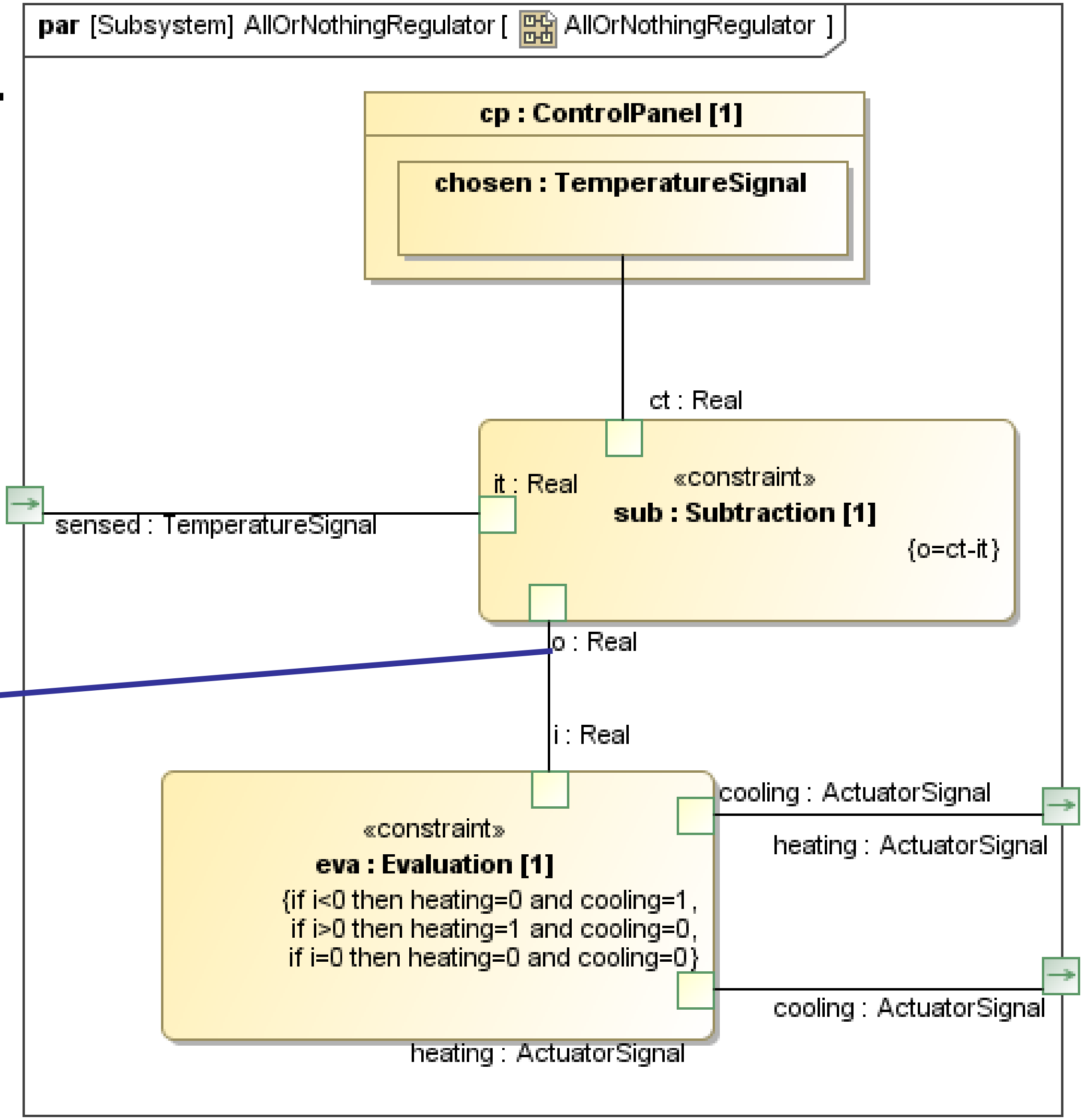


The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.

3. Performance Parameters – 2

- SysML Parametric Diagram establishes constraints by equations
  - Performance values are Outputs of constraint blocks

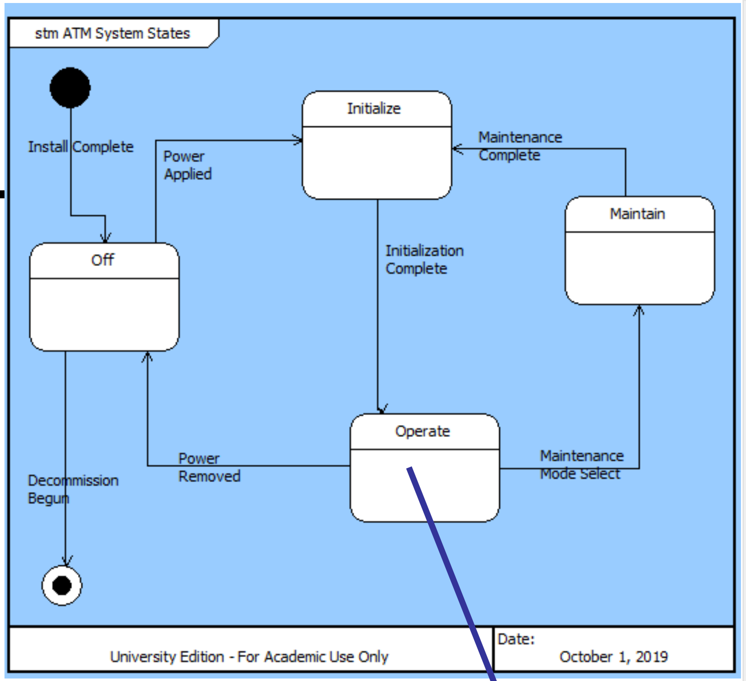
The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.

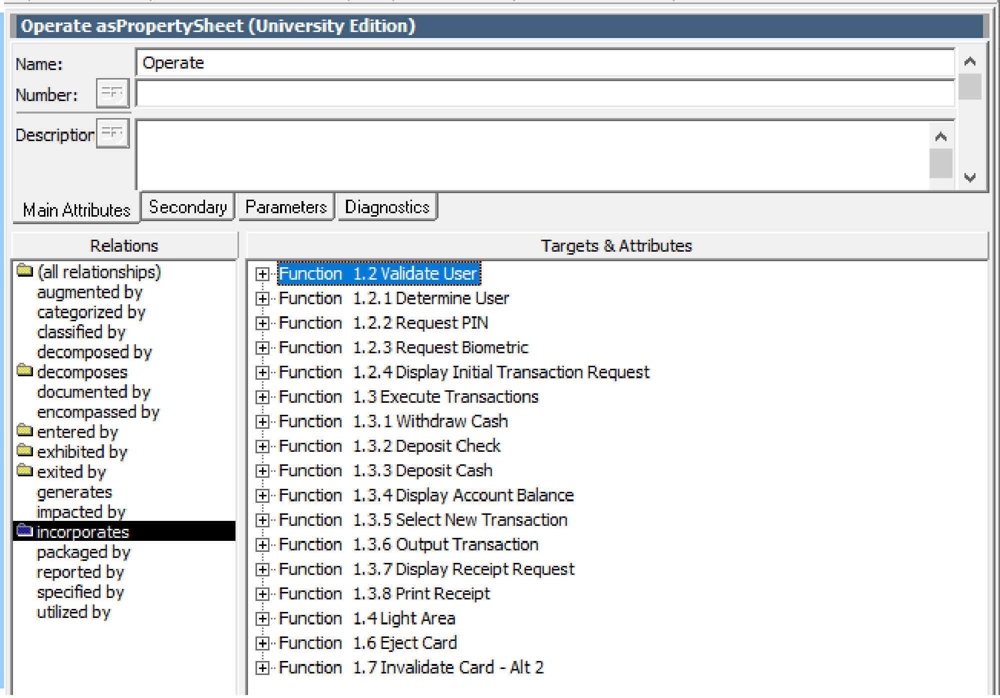


https://docs.nomagic.com/display/SYSMLP184/SysML+Parametric+Diagram

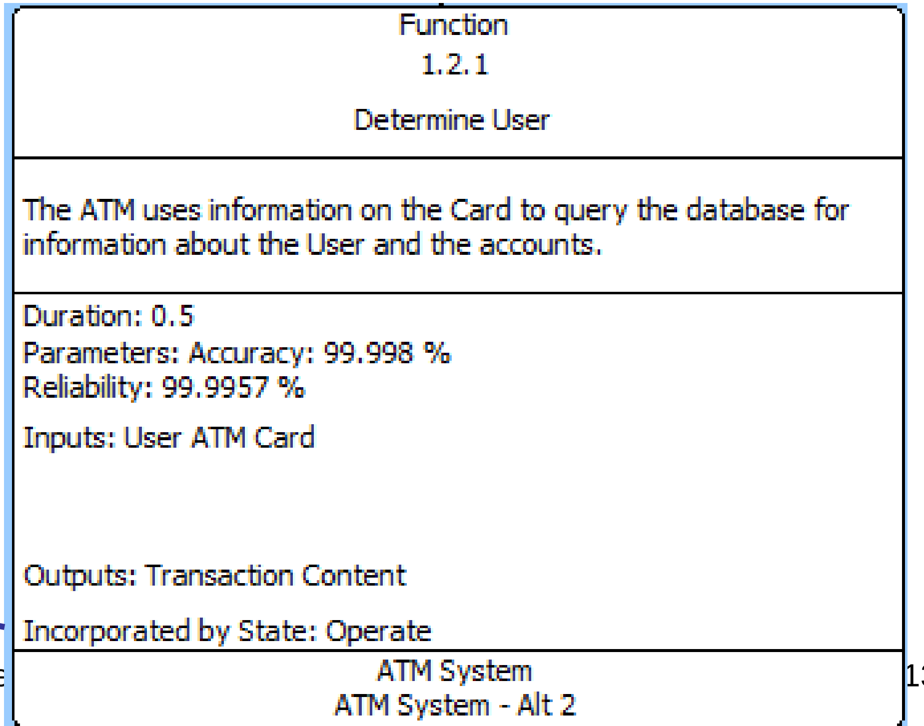
3. Conditions – 1

- States – Map Functions to States to assign applicability

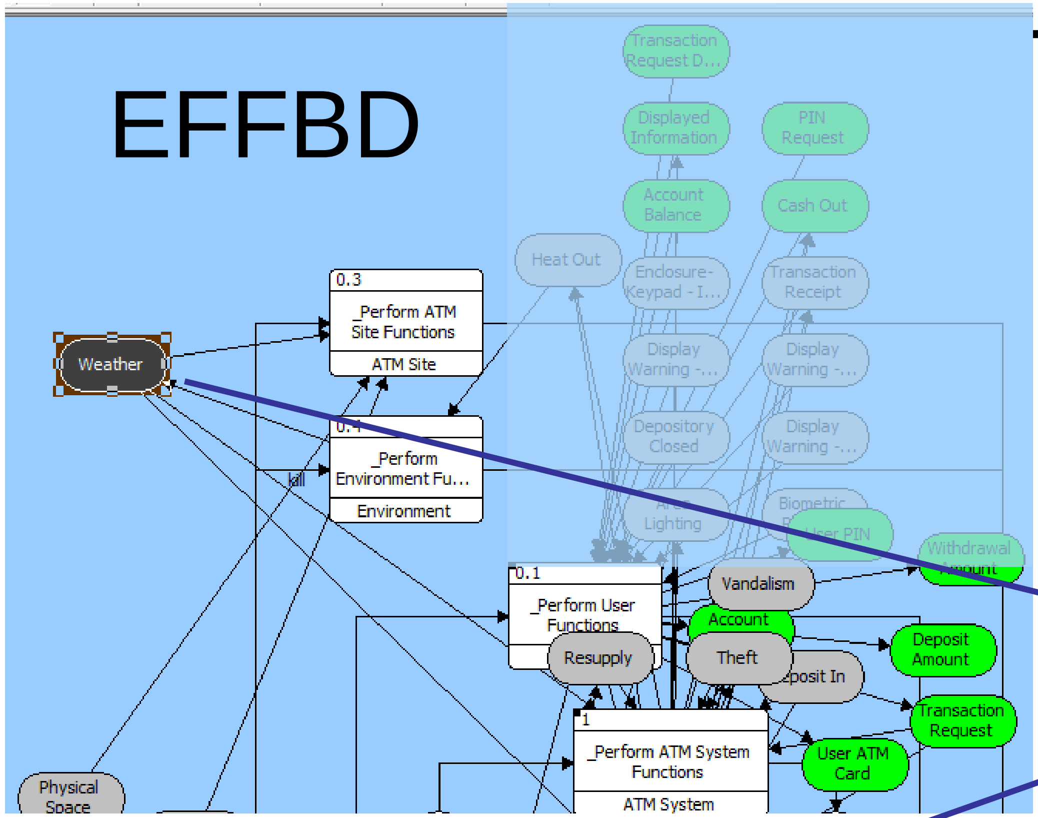


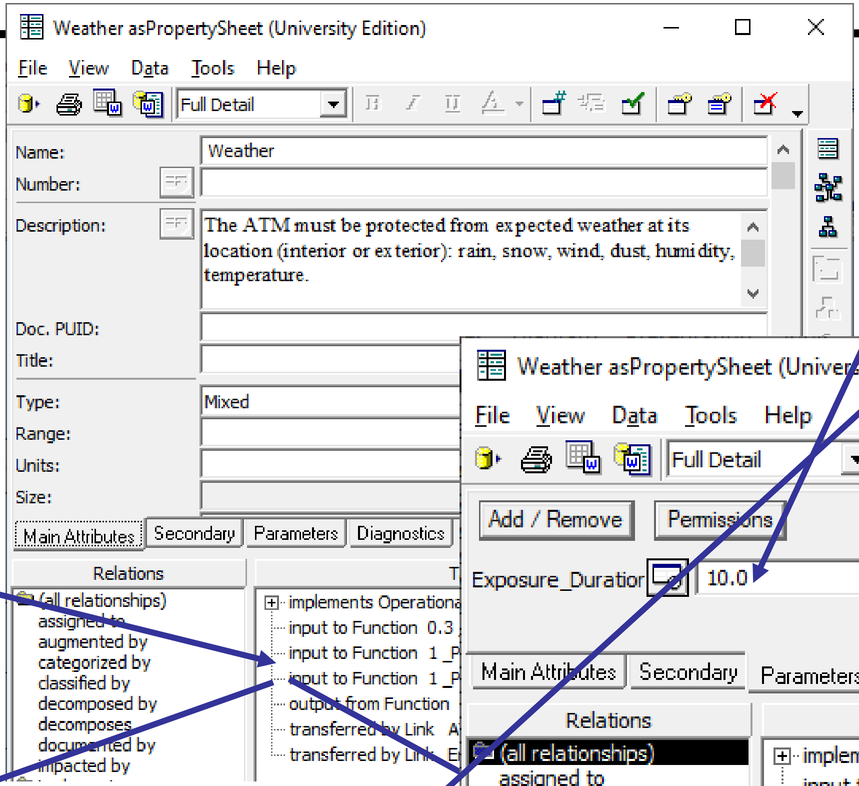


The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.

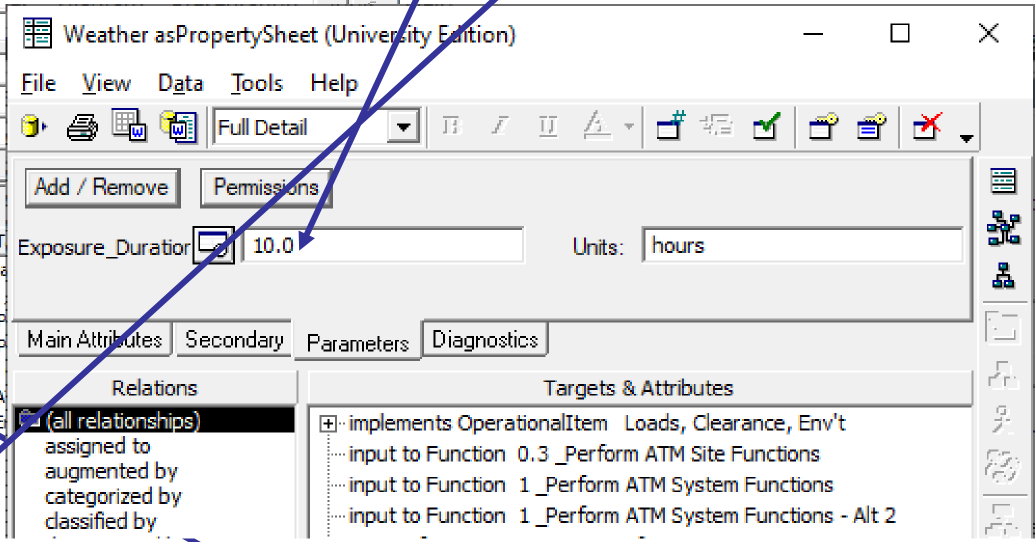


3. Conditions – 2: Environments as Inputs to Functions





“EXPOSURE DURATION” can be a Parameter on the Input Item

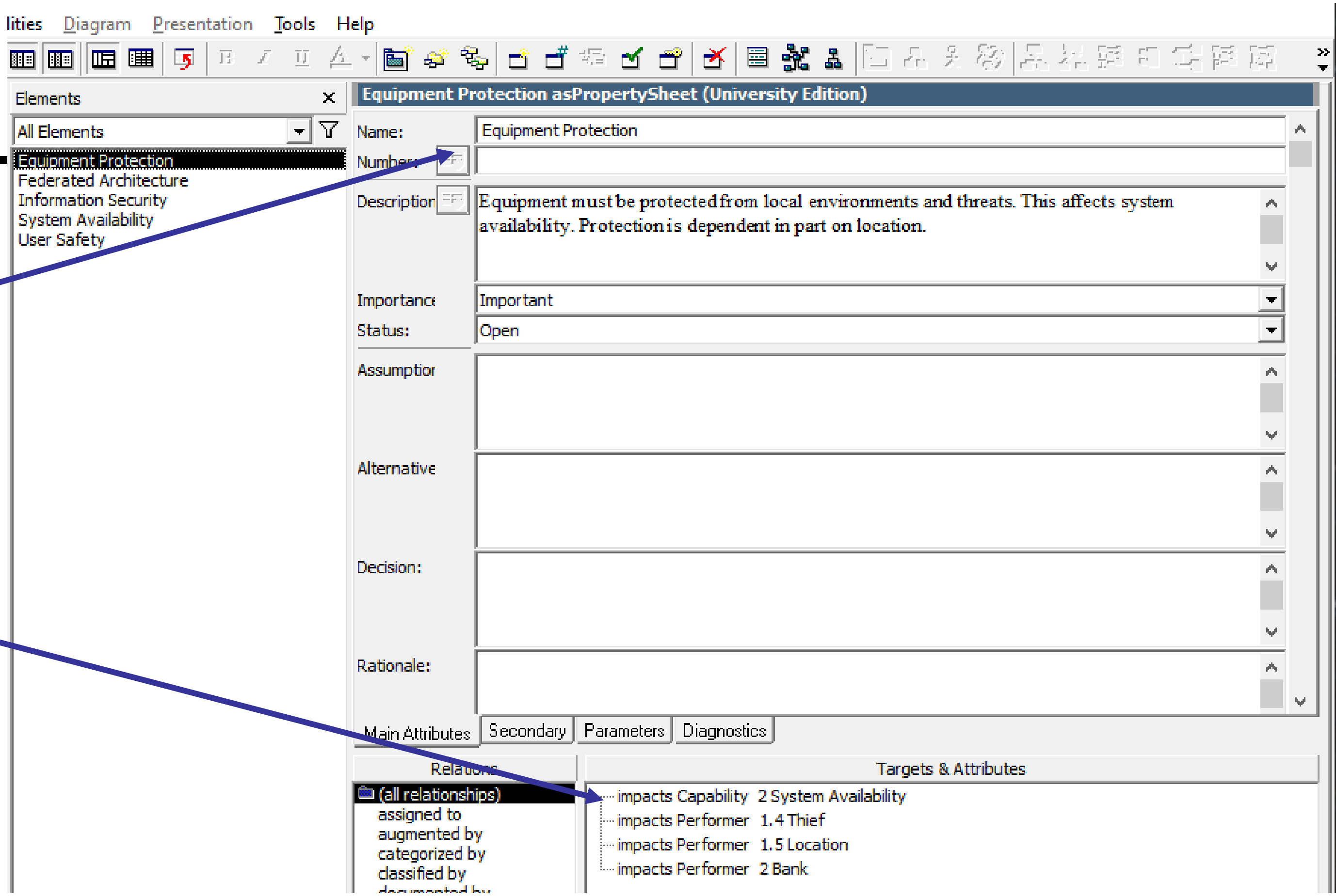


The AGENT shall exhibit CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION].

The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.

3. Design Constraint – 1

- Design characteristic for Design type constrains solution space based on
  - Derivation from Stakeholder Concerns and MOEs
  - Lessons learned – shortcut for other attributes based on Suitability, Environments
  - Allocation from Functional decomposition, e.g., fuel capacity, vehicle weight
- Parts, materials, processes should be derived from Stakeholder MOEs (e.g., availability in intended environments)



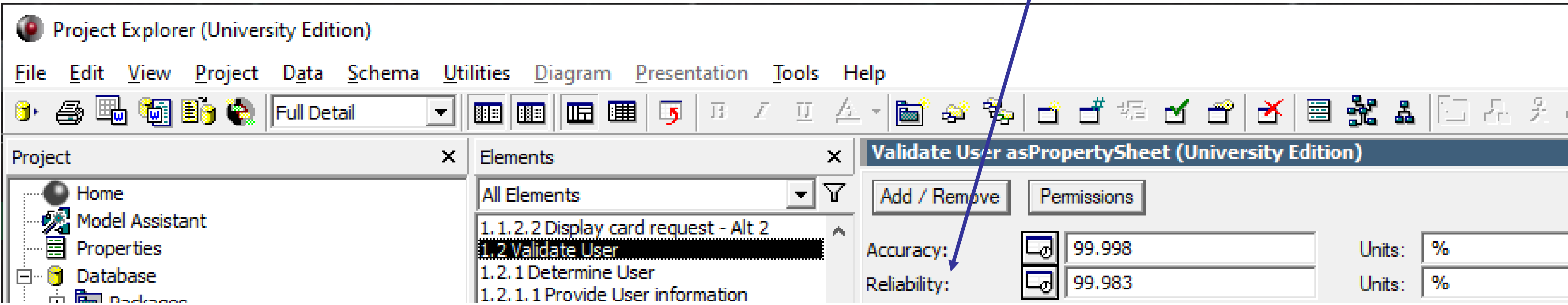
The AGENT shall exhibit DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION].

Availability MOE Traceability

3. Design Constraint – 2

- Design characteristic for Design type
  - Budgeted/allocated design attribute (space, weight, power, cooling, reliability)
- Derived from Parametric Diagram (e.g. fuel capacity) (F/P requirement decomposition)
- "Interface requirements" defined in F/P requirements (e.g., reference ICD)

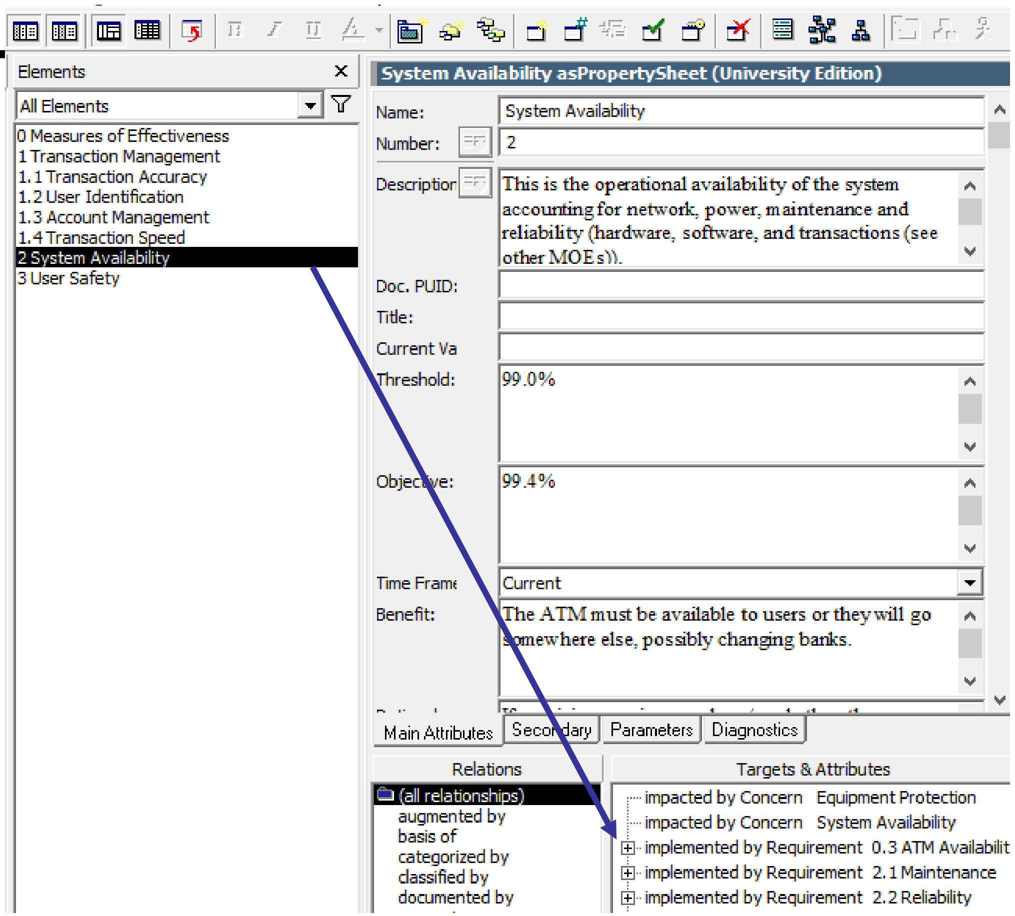
Reliability Allocations



The AGENT shall exhibit DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION].

3. “Characteristic” – Suitability

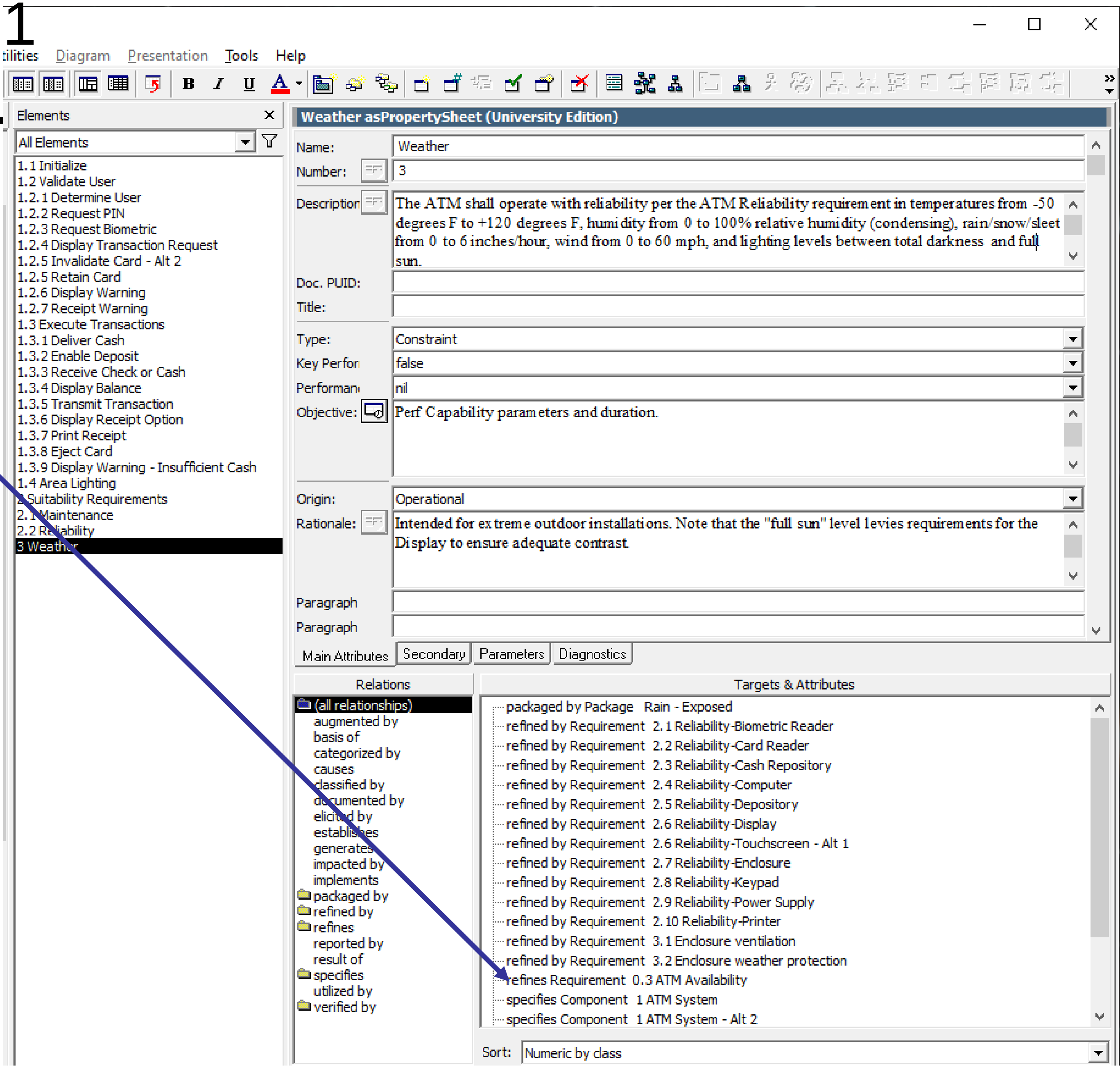
- Required attribute (“is”, “has” or “does”)
- Based on MOE
  - The system shall have availability > 99.99% while exposed to environments per section 3.2.6.



The AGENT shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION].

3. "Characteristic" – Environment – 1

- Based on Suitability
- The ATM shall operate with reliability per the ATM Reliability requirement in temperatures from -50 degrees F to +120 degrees F, humidity from 0 to 100% relative humidity (condensing), rain/snow/sleet from 0 to 6 inches/hour, wind from 0 to 60 mph, and lighting levels between total darkness and full sun.

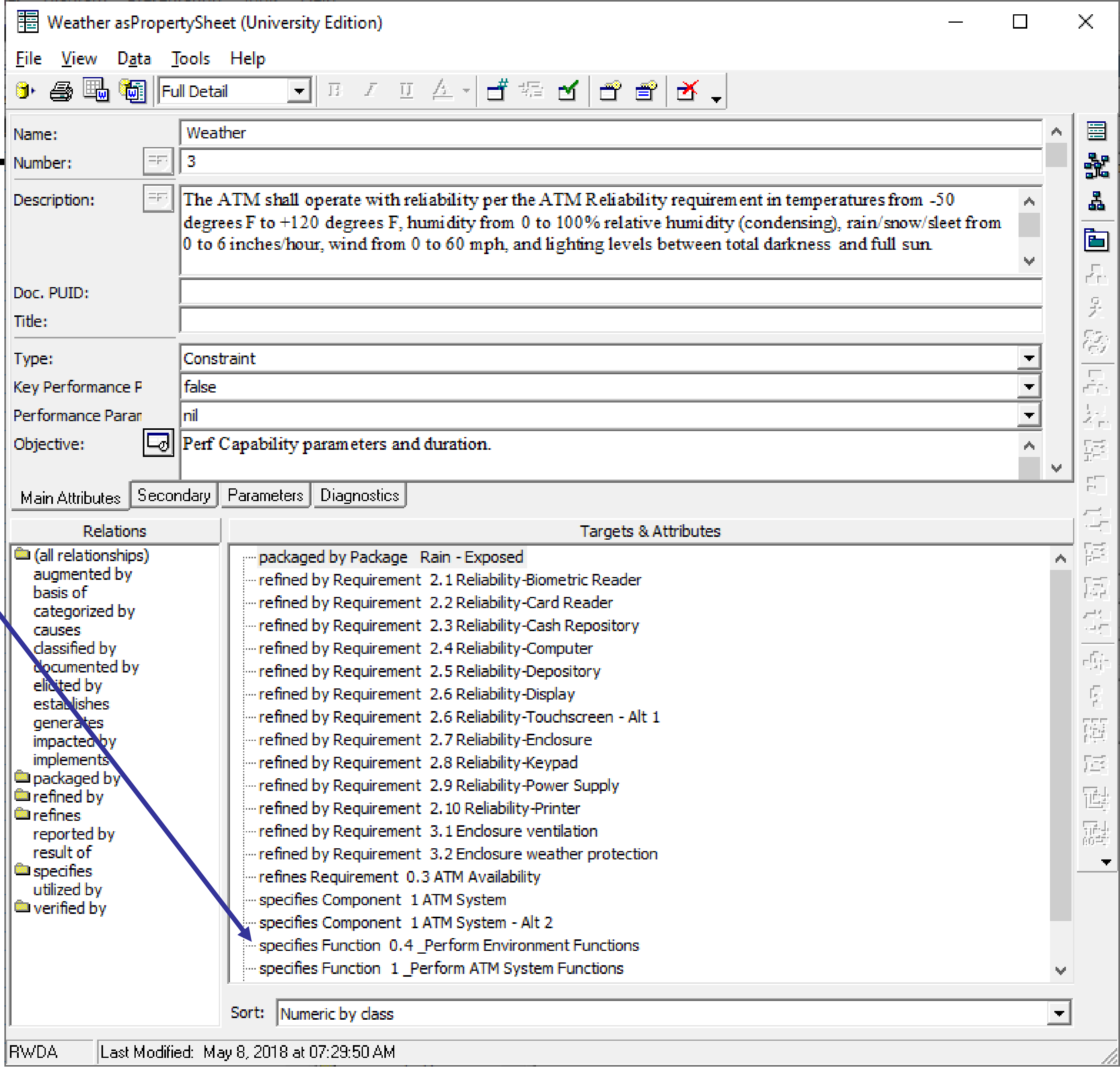


The AGENT shall exhibit CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION].

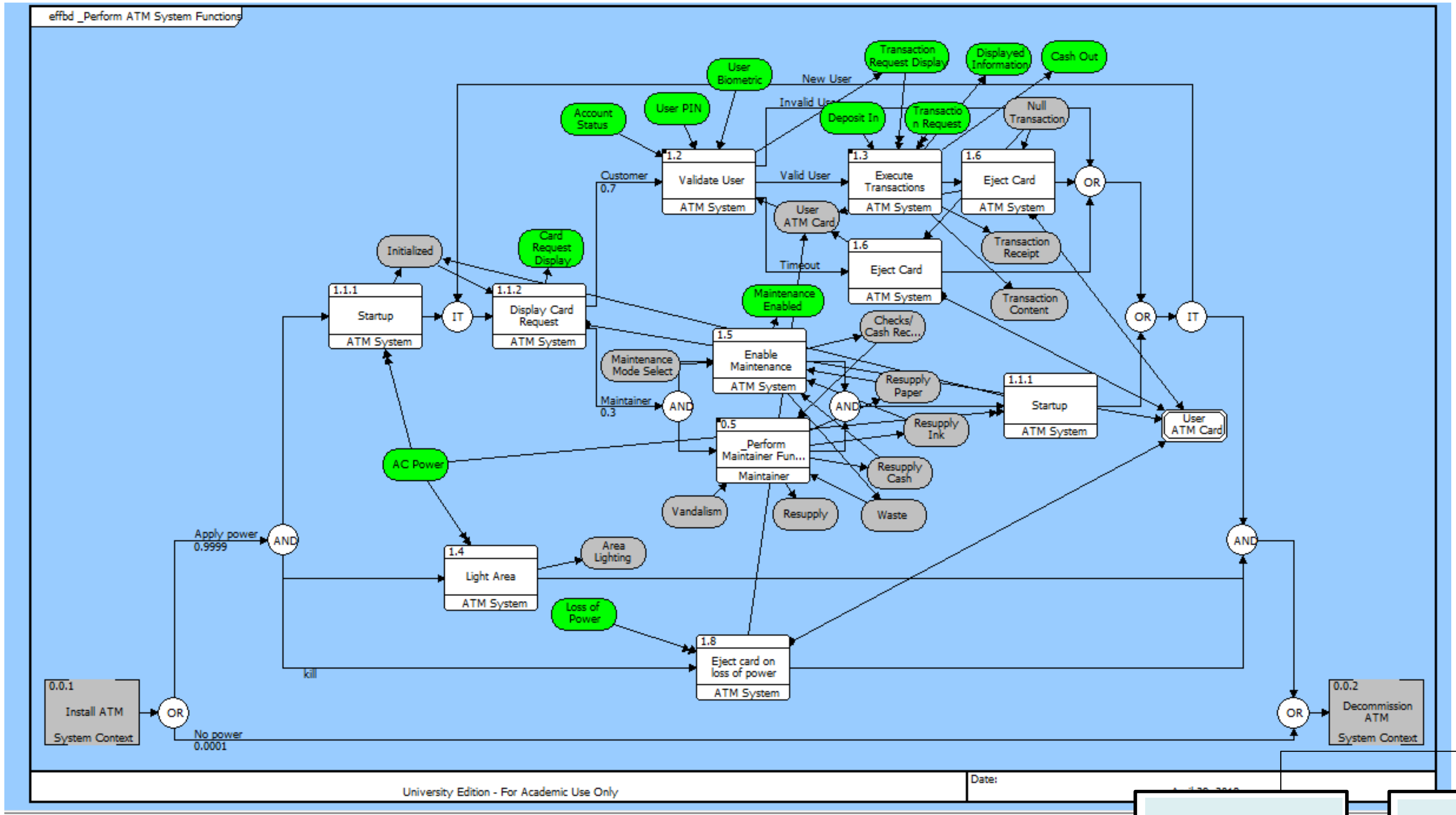
3. “Characteristic” – Environment – 2

- Based on Function
- The system shall satisfy all functional requirements per section 1 while exposed to operational environments per section 3.

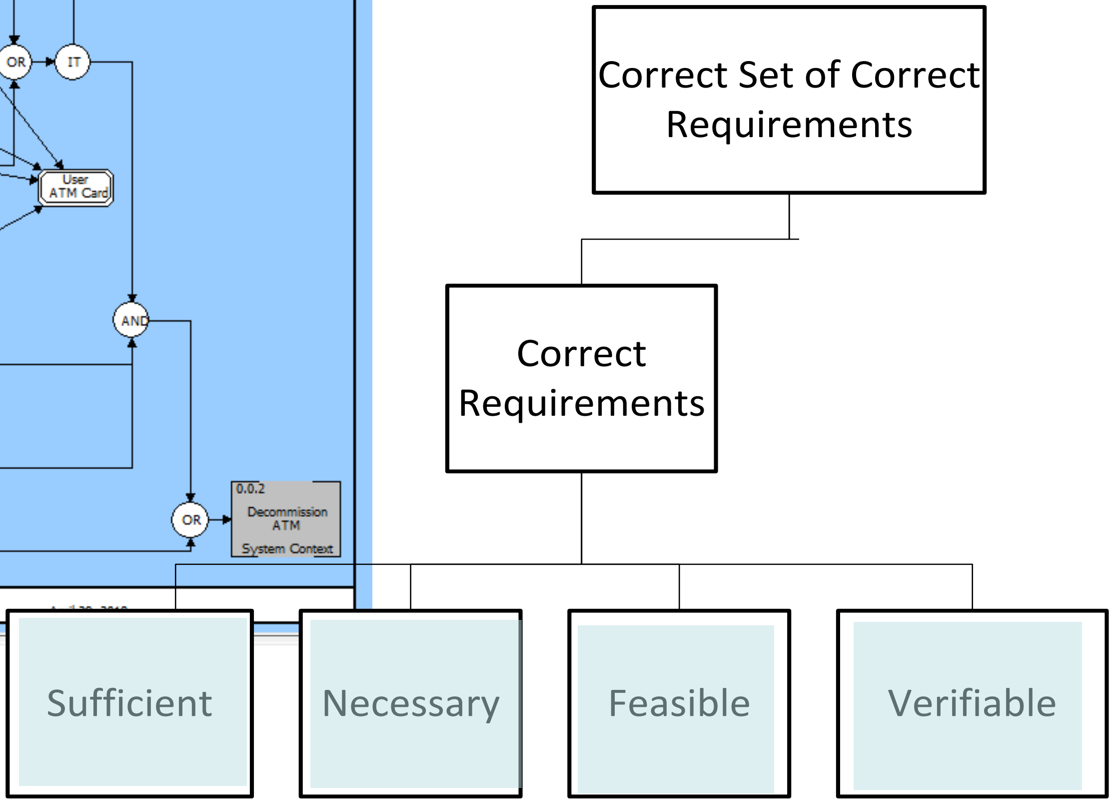
The AGENT shall exhibit CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION].



4. Using Architecture to Validate Requirements – Simulation

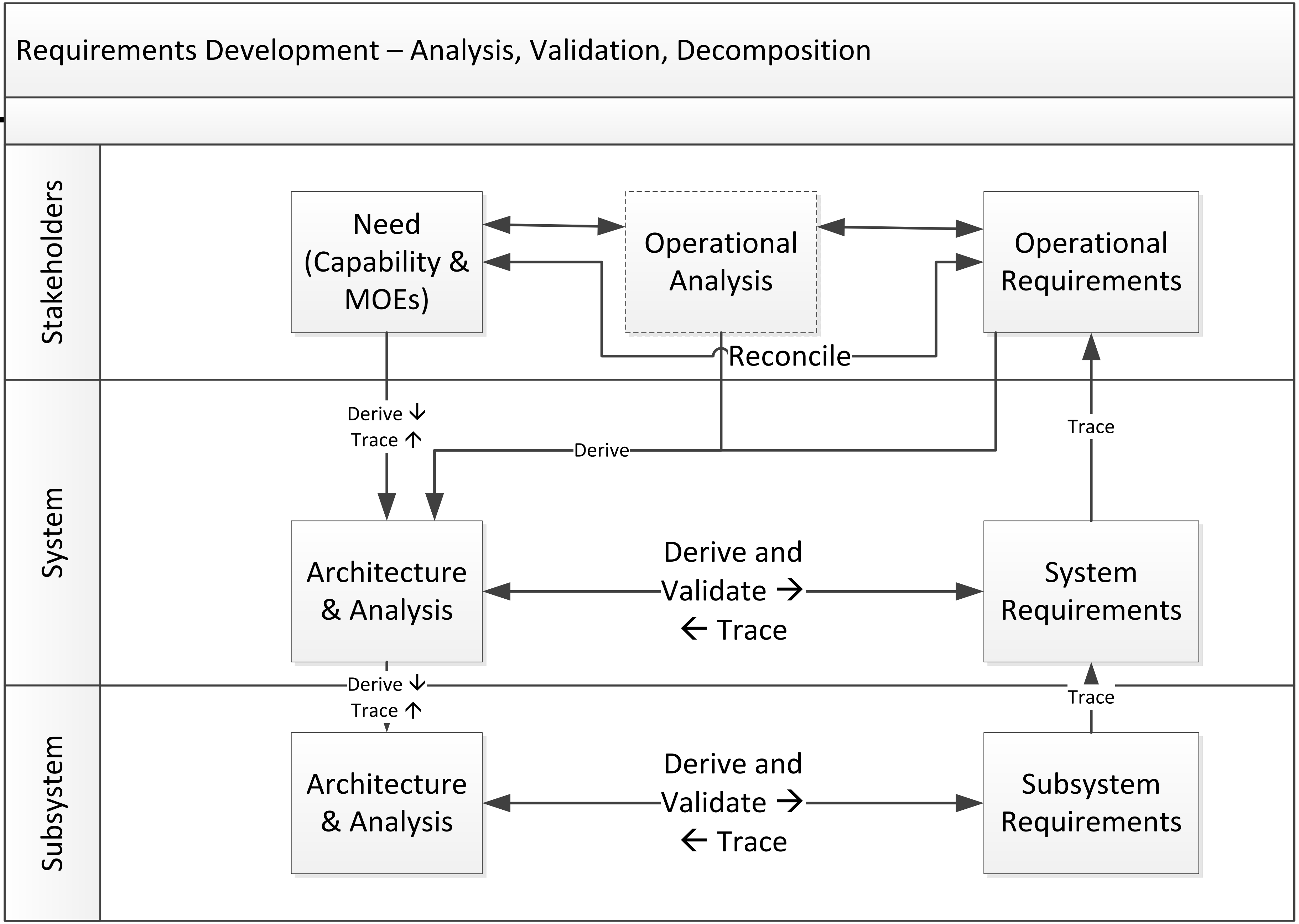


Carson and Noel, INCOSE 2018



5. Process First

- Analysis → requirements
- Next-lower-level architecture (design) constrains further analysis
  - Interfaces
  - Derivation and allocations of functions
  - Derivation of requirements
- Recursion to lowest levels
- Traceability is inherent in this process



6. Summary

- Process first: define structured requirements (types and structure by type) for your organization
- Select and adapt tool
  - Define representation of requirements elements in the tool data model
  - Define process for establishing *relationships* among model data elements
- Develop requirements *from* Concerns, MOEs, ConOps, Missions, Functions, Inputs/Outputs, Performance
  - Relate (trace) requirement elements to source data – MBSE architecture elements