Enterprise Architect Global Summit 2022

Traceability is a key to a valuable model

Takeshi Kouno

Sparx Systems Japan

tkouno@sparxsystems.jp

Agenda

1. Introduction
2. FAQs in Japan about modeling tools
3. Why we need modeling tools?
4. Traceability benefits and tips
5. A Problem in Traceability and its solution

- My session is announced as 60 minutes duration, but it will be approximately 40 minutes.
- I have uploaded this slide to the Teams channel.

Who am I

Takeshi Kouno

- CEO and Founder of Sparx Systems Japan (2003-)
  - Number of Enterprise Architect users in Japan:
    - 71 in 2003
    - 70,000+ in 2022
- My first Enterprise Architect was version 3.0

Frequently Asked Question

What are the differences between diagramming tools and modeling tools?

Is it enough to use a (free) diagramming tool or an Office tool like PowerPoint to describe a UML/SysML/BPMN/Archimate... model?

If you use Enterprise Architect to draw pictures, the answer of the second question might be Yes.

We should use Enterprise Architect as a modeling tool, that means we should receive the full benefits of Enterprise Architect.

But what and how?

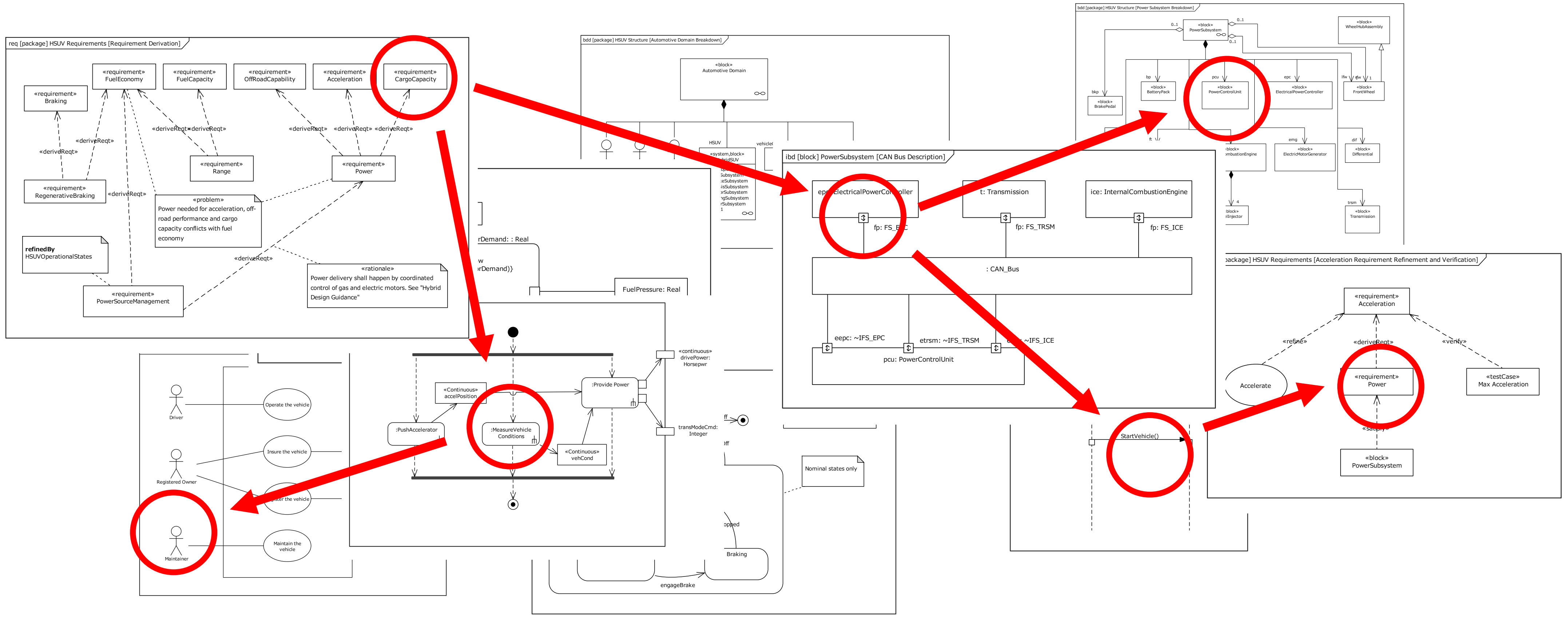
Main Benefits of Enterprise Architect

- Supporting lots of notations
- Generation source files and documents
- Simulation
- Project management
- Sharing model easily by ProCloud/WebEA/Prolaborate
- ...

However, the most basic use is to visualize the specification and design of the target system or software by using some notations.

Elements in Diagrams are Related

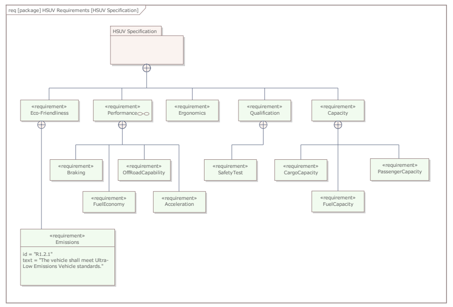
Elements on diagrams are related each other:

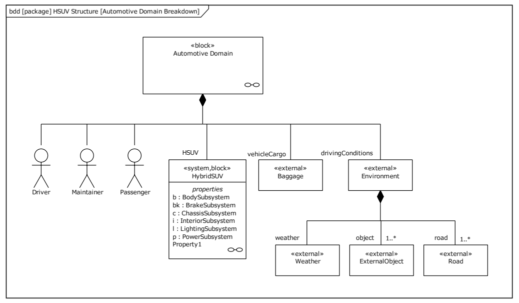


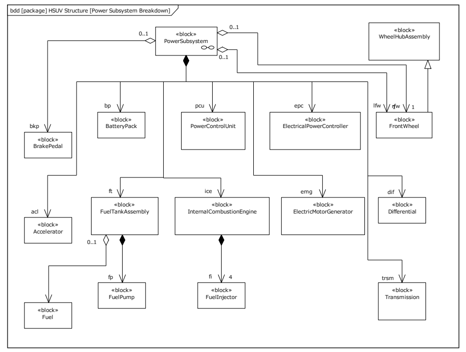
req [package] HSUV Requirements [Requirement Derivation]
«requirement» FuelEconomy
«requirement» FuelCapacity
«requirement» OffRoadCapability
«requirement» Acceleration
«requirement» CargoCapacity
«requirement» Braking
«requirement» Range
«requirement» Power
«requirement» RegenerativeBraking
«requirement» PowerSourceManagement
«deriveReqt»
«problem» Power needed for acceleration, off-road performance and cargo capacity conflicts with fuel economy
«rationale» Power delivery shall happen by coordinated control of gas and electric motors. See "Hybrid Design Guidance"
refinedBy HSUVOperationalStates
bdd [package] HSUV Structure [Automotive Domain Breakdown]
«block» Automotive Domain
ibd [block] PowerSubsystem [CAN Bus Description]
epc: ElectricalPowerController
t: Transmission
ice: InternalCombustionEngine
fp: FS_EPC
fp: FS_TRSM
fp: FS_ICE
: CAN_Bus
eepc: ~IFS_EPC
etrsm: ~IFS_TRSM
~IFS_ICE
pcu: PowerControlUnit
bdd [package] HSUV Structure [Power Subsystem Breakdown]
«block» PowerSubsystem
«block» WheelHubAssembly
«block» BatteryPack
«block» PowerControlUnit
«block» ElectricPowerController
«block» FrontWheel
«block» BrakePedal
«block» ElectricMotorGenerator
«block» Differential
«block» Transmission
package] HSUV Requirements [Acceleration Requirement Refinement and Verification]
«requirement» Acceleration
«refine»
«deriveReqt»
«verify»
Accelerate
«requirement» Power
«testCase» Max Acceleration
«satisfy»
«block» PowerSubsystem
StartVehicle()
Operate the vehicle
Insure the vehicle
Maintain the vehicle
Driver
Registered Owner
Maintainer
:PushAccelerator
«Continuous» accelPosition
:MeasureVehicle Conditions
«Continuous» vehCond
:Provide Power
«continuous» drivePower: Horsepwr
transModeCmd: Integer
FuelPressure: Real
Nominal states only
Braking
engageBrake

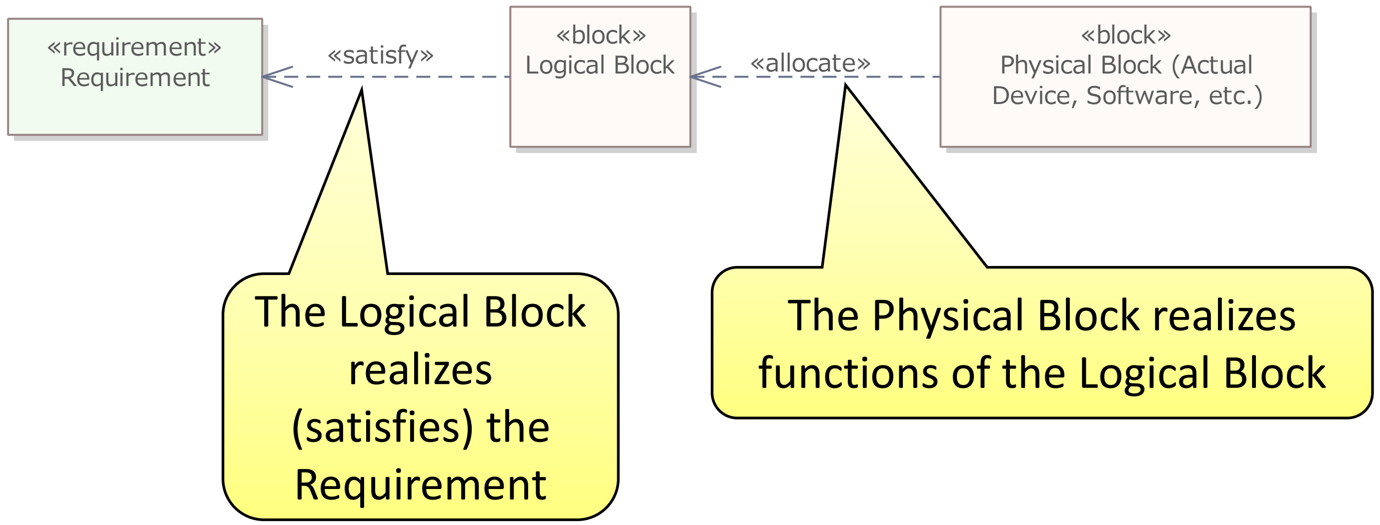
Simple Example

Requirement - Logical Block - Physical Block





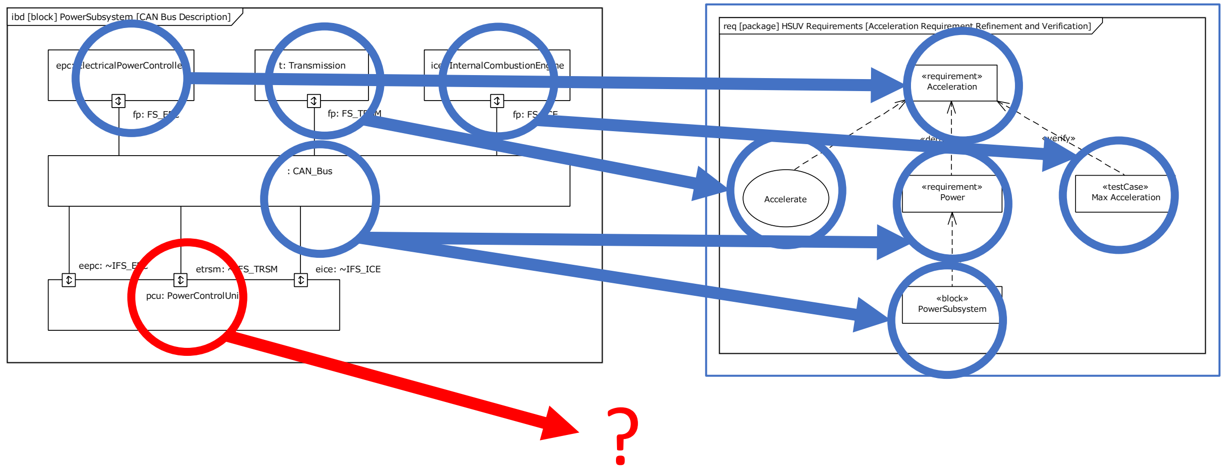



«requirement»
Requirement
«satisfy»
«block»
Logical Block
«allocate»
«block»
Physical Block (Actual Device, Software, etc.)
The Logical Block realizes (satisfies) the Requirement
The Physical Block realizes functions of the Logical Block

SPARX SYSTEMS JAPAN

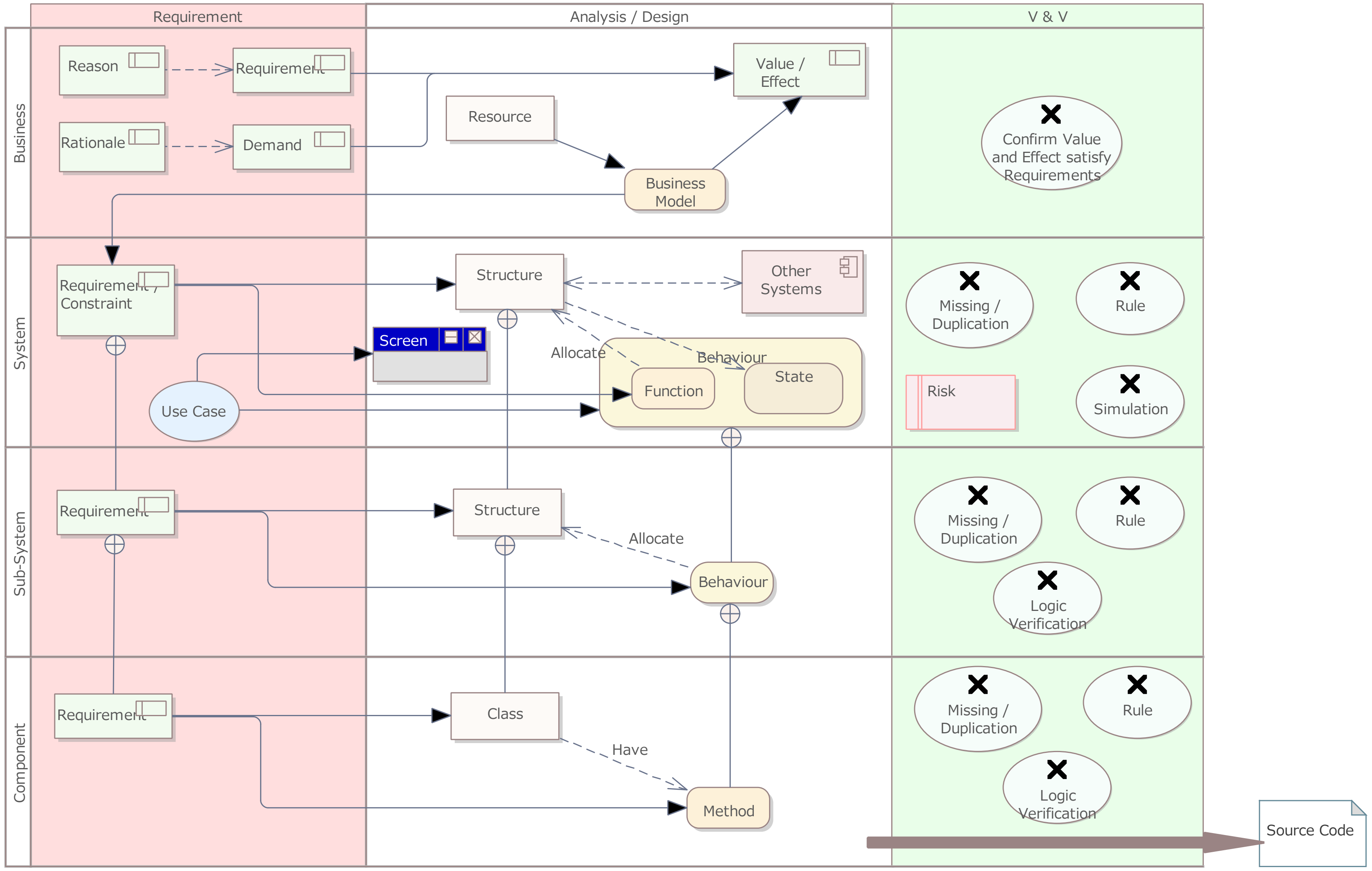
More Elements and Diagrams, More Complex

It is difficult to find unnecessary or isolated elements, inconsistent relationships among elements for large model - too complex!


ibd [block] PowerSubsystem [CAN Bus Description]
epc: ElectricalPowerControlle
t: Transmission
ice: InternalCombustionEngine
fp: FS_EPC
fp: FS_TRSM
fp: FS_ICE
: CAN_Bus
eepc: ~IFS_EPC
etrsm: ~IFS_TRSM
eice: ~IFS_ICE
pcu: PowerControlUni
req [package] HSUV Requirements [Acceleration Requirement Refinement and Verification]
«requirement» Acceleration
«deriveReqt»
«verify»
Accelerate
«requirement» Power
«testCase» Max Acceleration
«block» PowerSubsystem
?

How to manage Complexity among many diagrams and elements?

Complex Example - Hierarchical Model



Key to Manage Complexity: Traceability

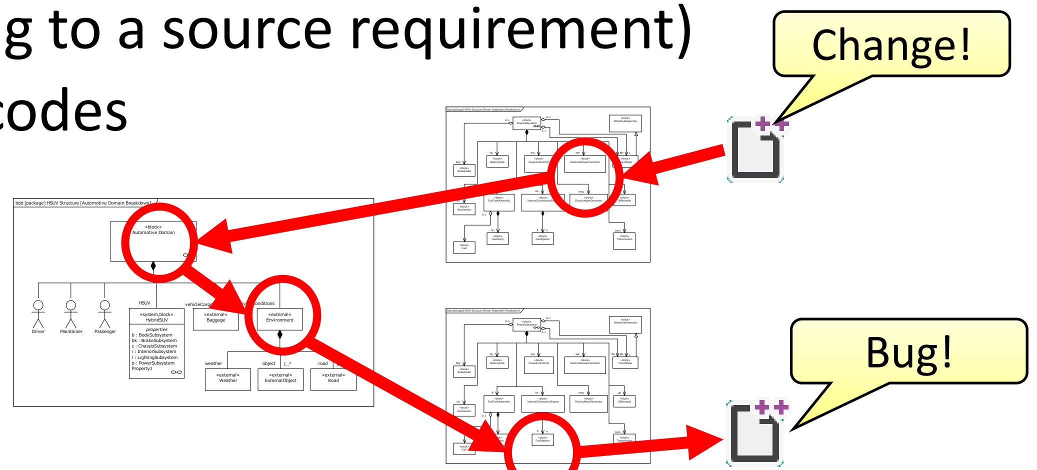
Traceability is information how to relate model elements.

From Wikipedia:

In systems and software development, the term traceability refers to the ability to link product requirements back to stakeholders' rationales and forward to corresponding design artifacts, code, and test cases. Traceability supports numerous software engineering activities such as change impact analysis, compliance verification or traceback of code, regression test selection, and requirements validation.

Merits of Traceability

- We can confirm:
  - All requirements are realized by some elements
  - There is no isolated elements i.e. unnecessary elements
- We can know:
  - Why elements are necessary (by tracing to a source requirement)
  - Change impact **before** change source codes



Traceability is necessary for describing Specification and Design!

Traceability is an Important Key

Diagrams made by drawing tools is just pictures and have no information about relationships to other pictures.

Specification and Design contain various kind of relationships (i.e. Traceability), but we cannot describe the relationships by a bunch of pictures.

That is why we use modeling tools for describing specifications and designs, and managing Traceability in them.

(I know, there are other reasons to use modeling tools, but today I will talk about Traceability.)

Sorts of Traceability in Enterprise Architect

1. Connector
   - In UML, usually Trace or Abstraction types are used
   - Some notations have special types for Traceability (e.g. SysML)
2. Classifier/Type
   - Class - Object/Part/Port/Partition/...
   - Attribute - Type, Operation - Return Type, Parameter - Type
   - Trigger - Signal
   - Action - Signal
3. Placement in a diagram
4. Hierarchy in the Browser
5. Tagged Value - RefGUID type

Facilities for Traceability in Enterprise Architect

- Relationship Matrix
  - Design Ribbon -> Package/Matrix
- Diagram Matrix View
  - Diagram context menu -> Switch View -> Switch to Relation Matrix View
- Traceability Window
  - Design Ribbon -> Trace
- Find in all Diagrams (Element Usage)
  - Element context menu -> (Find ->) Find in all Diagrams

Sparx Japan offers the Traceability Suite Add-in. In some demo, I use the Add-in.

https://www.sparxsystems.jp/en/trace/

Why There are Many Facilities

Because purposes are different:

- Matrix:
  - to confirm all elements are related, but direct relationships only
  - to find missing and/or isolated elements
- Traceability Window, Traceability Map*:
  - to check hierarchical relationships (multi-level relationships)
- Usage in Diagrams, ElementUsageEx*:
  - to help understanding change impact

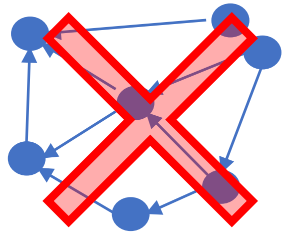
We need to use all facilities for Traceability management.

*: The TraceSuite Add-in features

Tips about Traceability

Defining and maintaining Traceability information costs a lot, so

- At first, consider objective why you need Traceability - then only related elements are target to define Traceability
  - Do not define Traceability for all elements
- If a model is temporary, Traceability is not necessary.
  - Multi-layer models (e.g. System - Subsystem - Component) and/or long-term development are the best for Traceability
    - We can receive benefits from Traceability more than cost paying for Traceability
- Do not define complex Traceability
  - It costs a lot to keep latest/correct

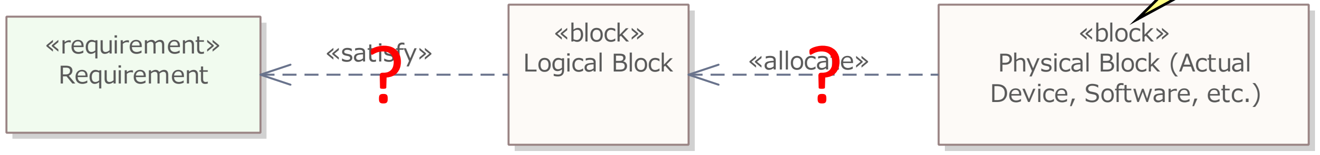


Problem in Traceability

We can see relationships in the model, and we can find missing and/or isolated elements by Traceability but...

We cannot find inconsistency in the model by using (single-path) Traceability. The model with perfect Traceability does not mean consistent and validated.

Example:


«requirement» Requirement
«satisfy»
?
«block» Logical Block
«allocate»
?
«block» Physical Block (Actual Device, Software, etc.)

The Physical Block is traced from a Requirement correctly (i.e. not isolated), but are these relationships are correct?

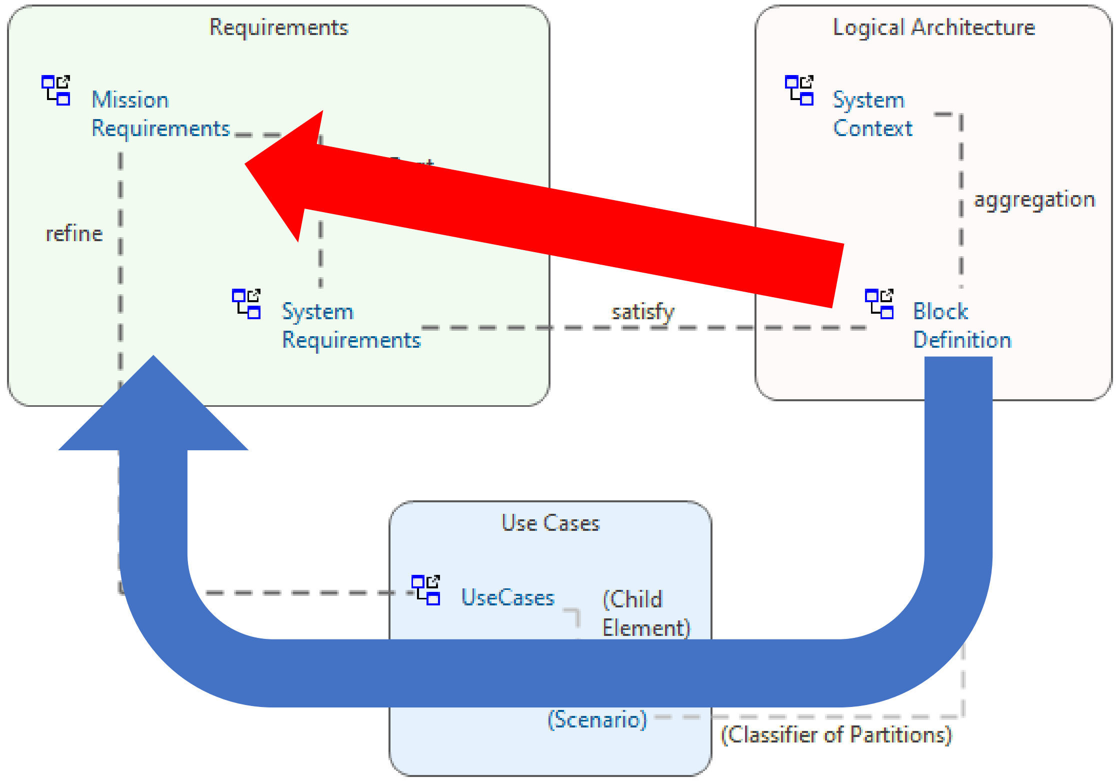
Solution

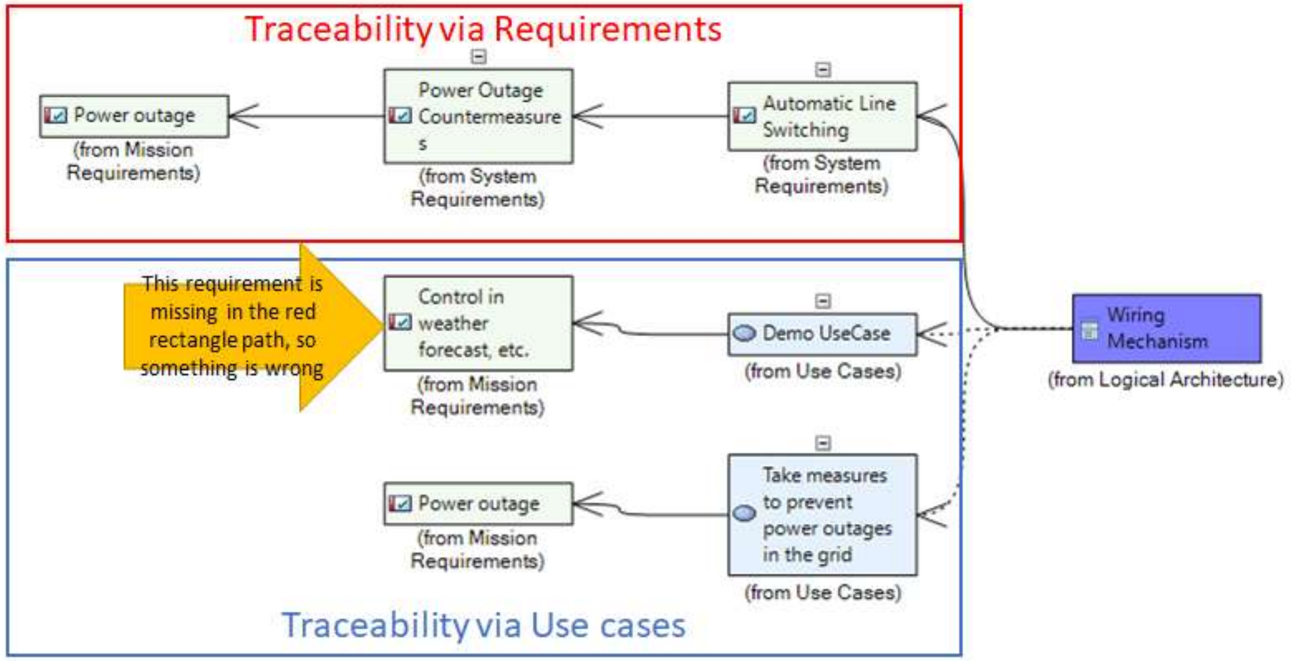
2-paths Traceability can solve the problem

- We can confirm the correctness of model and Traceability information automatically

Example Model of 2-Paths

In my MinimumMBSE document:




Traceability via Requirements
Power outage
(from Mission Requirements)
Power Outage Countermeasures
(from System Requirements)
Automatic Line Switching
(from System Requirements)
This requirement is missing in the red rectangle path, so something is wrong
Control in weather forecast, etc.
(from Mission Requirements)
Demo UseCase
(from Use Cases)
Wiring Mechanism
(from Logical Architecture)
Power outage
(from Mission Requirements)
Take measures to prevent power outages in the grid
(from Use Cases)
Traceability via Use cases

https://www.sparxsystems.jp/en/trace/minimumMBSE.htm

Summary

- Traceability is one of the keys to distinguish modeling tools from drawing tools
- We need to use suitable facilities to know/investigate models
- Traceability sometimes makes model too complex and we must pay cost to manage Traceability - try to find your best Traceability level
- Single-path Traceability cannot find inconsistency - consider if you can make 2-paths Traceability

Thank You!

I hope this session could help your better Enterprise Architect life.

- I have uploaded this slide to the Sparx Systems Japan Teams channel.
- If you have any questions, comments and/or ideas, Please drop a post in the Teams channel or email to tkouno@sparxsystems.jp.