Using MBSE with SysML Parametrics to Perform Requirements Analysis

Yvonne Bijan

Dr. Henson Graves

Dr. Junfang Yu

Dr. Jerrell Stracener

Dr. Timothy Woods

Introduction

- Poorly written requirements are a frequent cause of failure on programs
- Improving requirements can result in better performance on projects
- A cause of unverifiable requirements is the use of ambiguous terms (Hooks, 1993)
- As long as requirements are text based, they are likely to be vague and ambiguous
- In writing requirements, precision is required (Bittner, 2008)

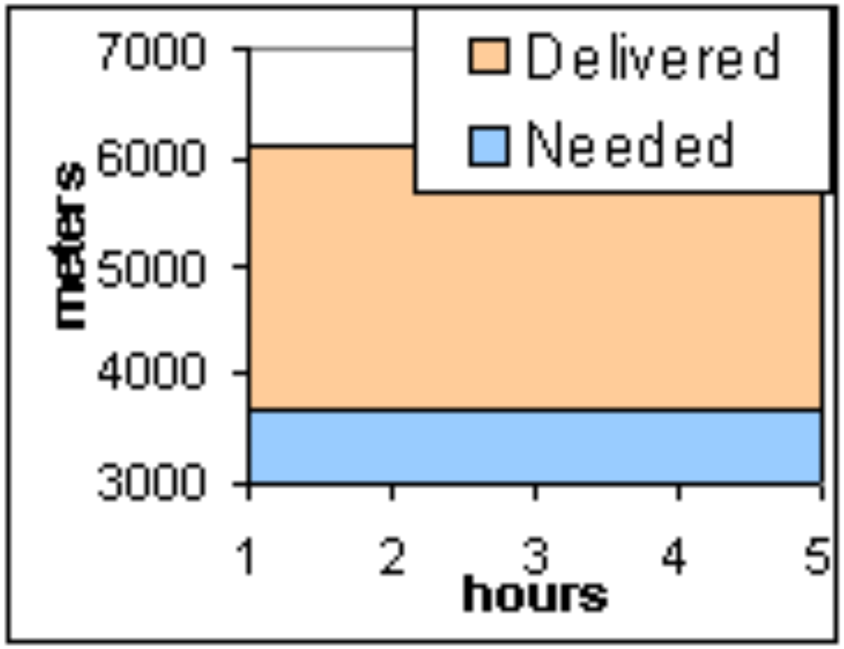
How can we make requirements more precise?

Scenario

- Customer
  - Needs an aircraft that can perform a holding pattern for five hours in order to perform a mission objective
- Contractor
  - Develops an aircraft that can do so while flying at an altitude of 6,000 meters
- Customer
  - Complains that during validation, the aircraft can only loiter for four hours
- An investigation ensues

Scenario Investigation

- Contractor determines that the customer is flying the aircraft at 3,600 meters, which has higher fuel consumption than at 6,000 meters
- Contractor notifies the customer that they need to fly at 6,000 meters (as designed)



Scenario

- Contractor
  - States customer never said they needed to fly it lower
- Customer
  - Needs to fly lower
  - Wants the aircraft fixed
  - Wants more fuel on the aircraft so they can fly longer
- Issues
  - Increased weight
  - Reduced cargo carrying capacity
  - Added complexity

What makes a Good Requirement?

- complete
- free of contradiction
- unambiguous
- feasible
- understandable
- verifiable
- identifiable
- atomic
- free of duplication
- correctly derived
- traceable to source (Hood et al. 2008)

Issues with Requirements

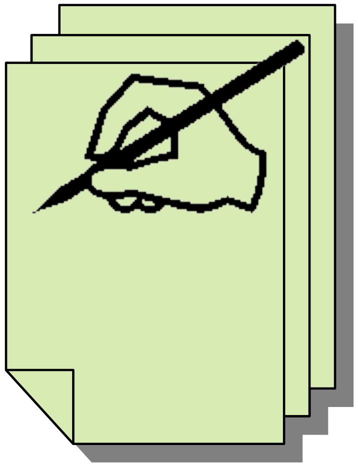
- Characteristics are not assessed with any consistency
- All characteristics are not included, especially unambiguous and completeness
- A major cause of unverifiable requirements is the use of ambiguous terms (Hooks, 1993)
- Vague and ambiguous when they are text based, even when they use relationships, attributes, and diagrams.
- Often incomplete because environmental constraints are not expressed as part of the context
- Much of the system development work should consist of understanding the intent of customer requirements and making the requirements precise with the conditions under which the requirements are valid

Questions that need to be answered

- We all think we know a good requirement when we see one, but:
  - How do we construct one?
  - How do we take a vague need/requirement handed to us from a customer and refine it into a complete requirement that addresses the conditions under which it is true?
  - How do we eliminate or reduce vagueness to increase the precision of requirements?
  - How do we express environmental constraints as part of the system context in order to have a more complete requirement?

Approaches

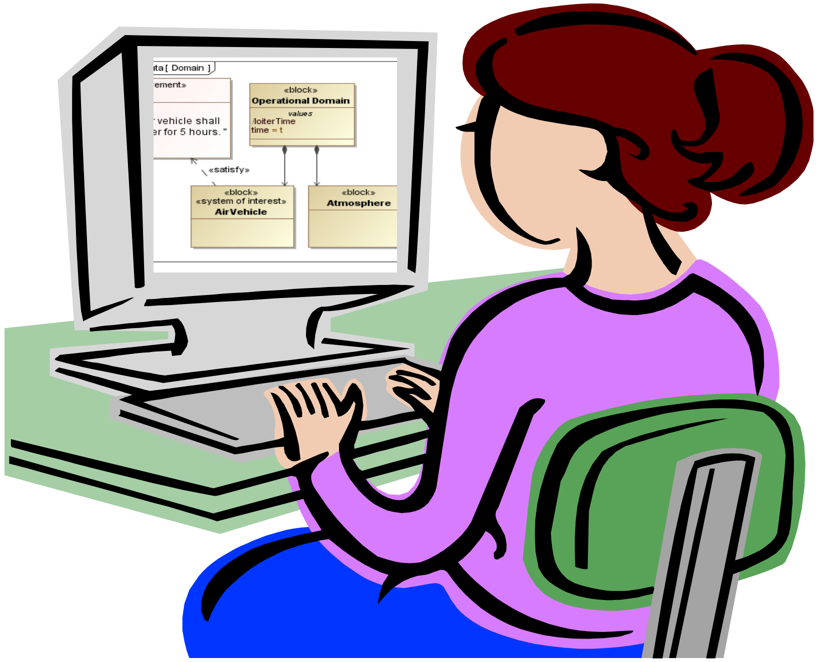
- Document-based approaches have fundamental limitations regarding the completeness, consistency and relationships between requirements
- The completeness, consistency, and relationships between requirements, design, engineering analysis, and test information are difficult to assess since this information is spread across several documents. (Friedenthal et al. 2008)



- Planguage captures detail about the requirements to clarify them (Gilb, 2010)
- Useful for considering all of the elements that should be included in a requirement
- But the requirements are still textual

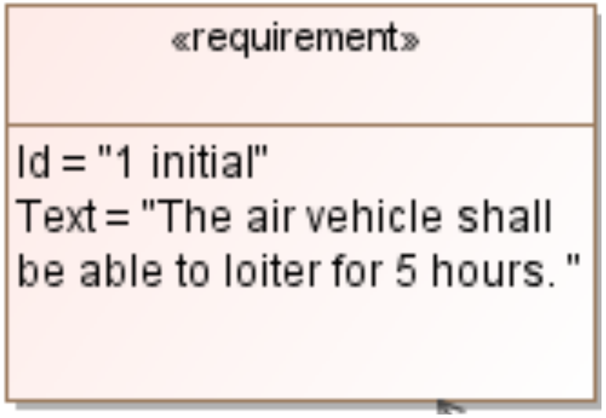
Approaches

- Model-Based Systems Engineering with SysML addresses relationships among requirements and traceability to parts of the system
- Requirements are still modelled as plain text statements within the requirement model element in SysML



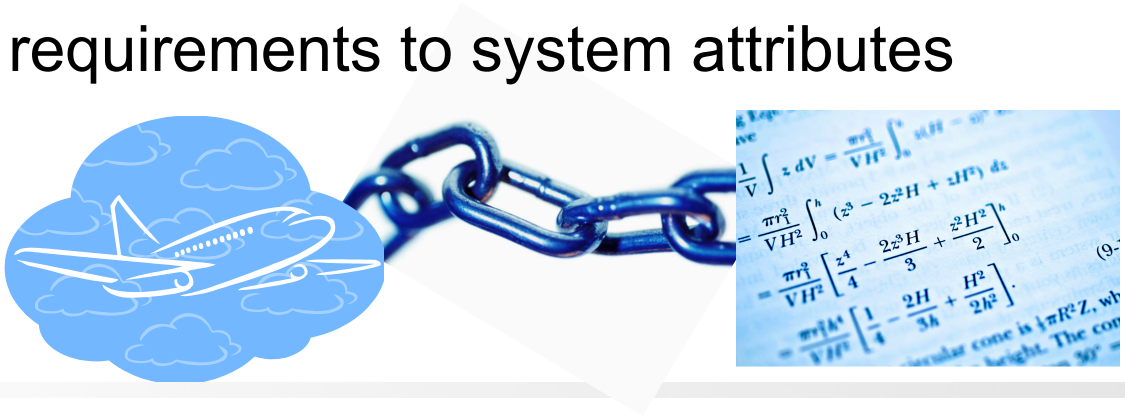
Model Based Systems Engineering

- By applying a model based approach, requirements and system interfaces become much more than simple text as they can be automatically validated and re-used (Karban et al. 2009)
- However requirements are still just imprecise textual statements
- SysML gives Systems Engineers the ability to define trace, verify, satisfy, derived, etc. relationships to other model elements
- What is lacking is the ability to truly integrate requirements in a model of the system within its operating context


«requirement»
Id = "1 initial"
Text = "The air vehicle shall be able to loiter for 5 hours. "

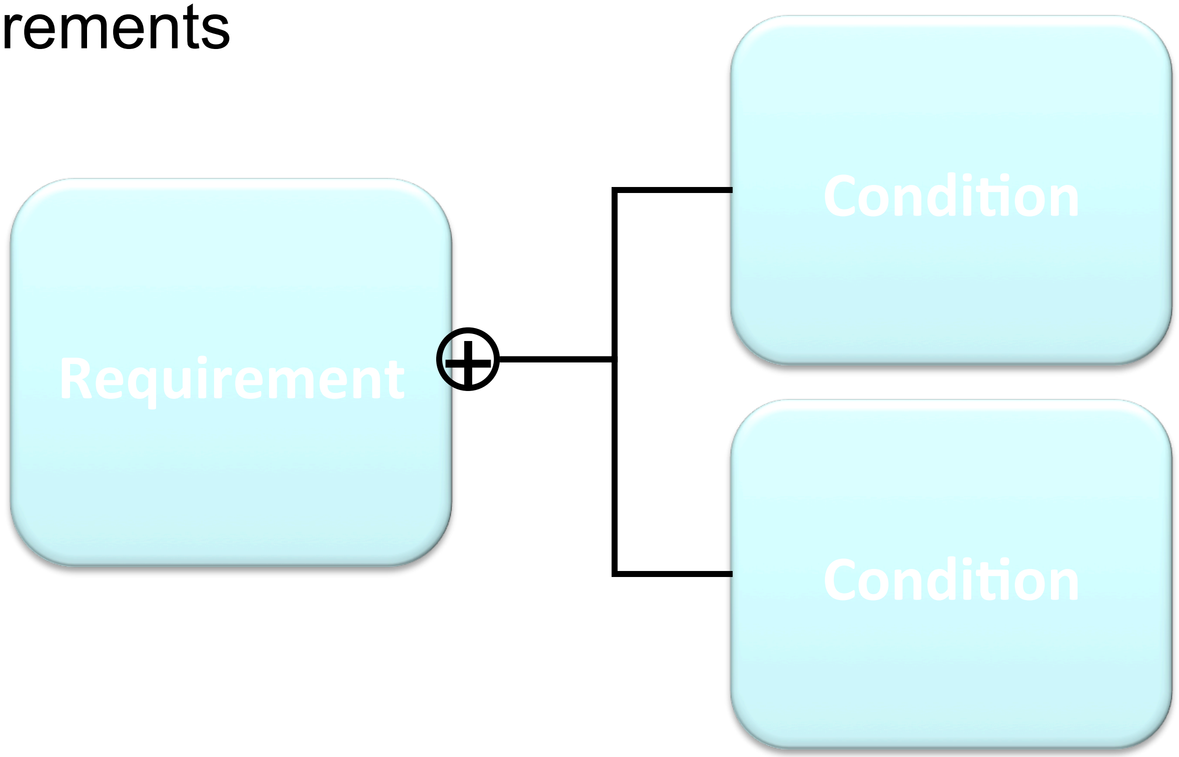
SysML Support

- SysML parametrics provides the ability to
  - Define equations for analysis
  - Define system constraints
  - Define interactions among system components and the environment
  - Link mathematical models to system attributes
- SysML cannot
  - Link requirements to constraints
  - Link requirements to system attributes



Solution

- Use Model-Based Systems Engineering with SysML parametrics and some extensions to develop requirements that are unambiguous, consistent, and complete with traceability to the system satisfying the requirements



Model Objectives

- Include environment
- Formalize performance requirements using parametrics
- Make the requirements and constraints more precise
- Use the domain model to describe the conditions under which requirements are to be verified
  - This also documents intended usage which can be validated by the customer
- Enable consistency checks
- Enable completeness checks
- Enable flow down to lower tiers as the system is decomposed into component subsystems and constraints are allocated to subsystems

Steps

- Define a domain model that includes
  - System of interest
  - Operating context for the system (environment)
- Add the initial customer needs/requirements as requirements blocks
- Flush out the value properties of the system of interest and environment
- Define a constraint block that will contain the constraints and parameters that are needed for analysis
- Define a mathematical function that represents the relationships the value properties of the system
- Update the requirement to capture the conditions
- Define test cases for verification

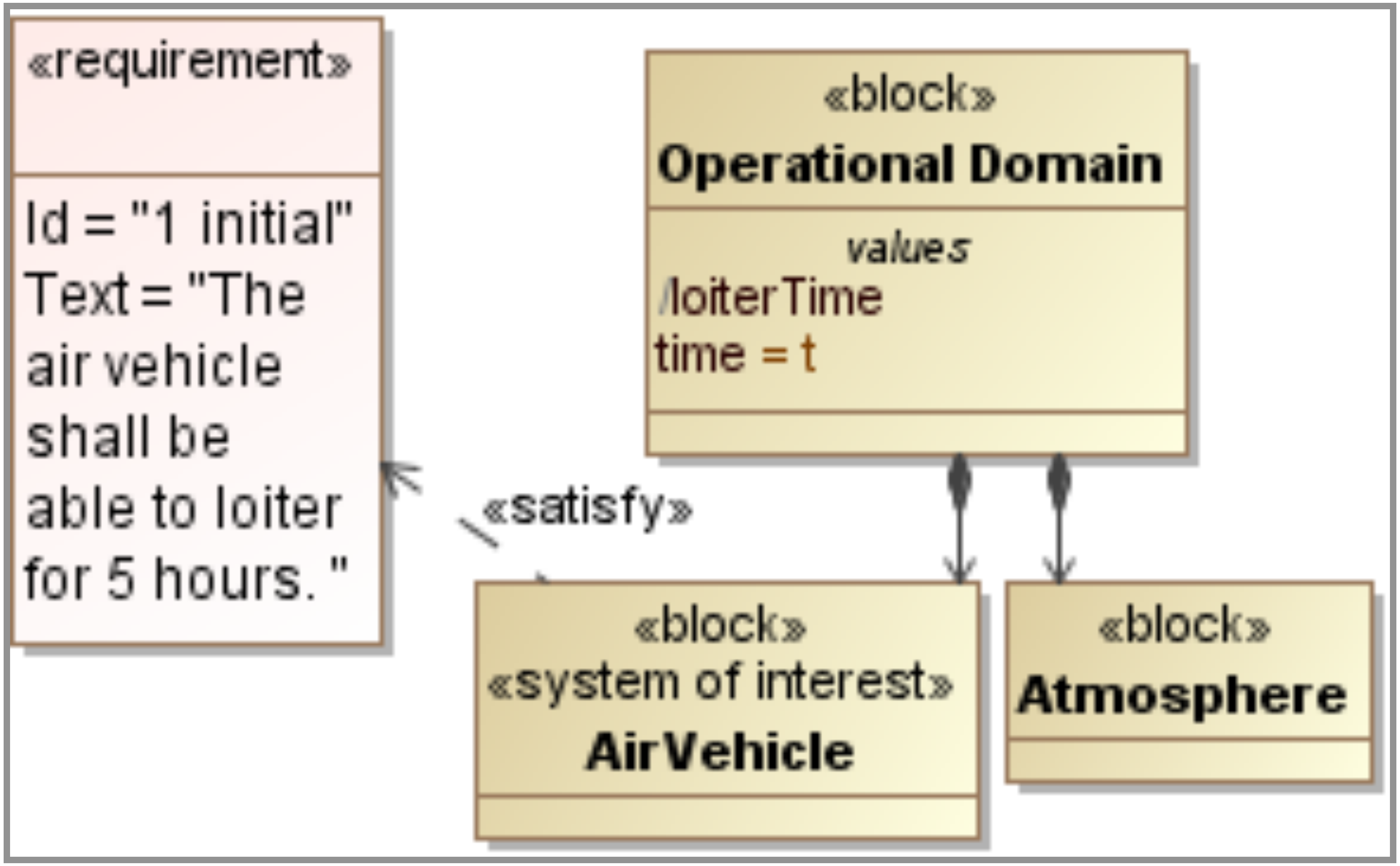
Customer Requirement

- Requirement starts out as
  - The air vehicle shall be able to loiter for at least five hours.
- The problem so far is that the requirement is too vague and the customer does not know enough to bound it.
- An analysis should be performed to understand what impacts the requirement and the relationship between this requirement and others.

Construct Domain Model

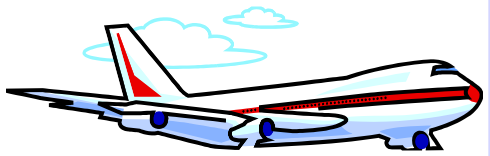
- Start with a model that contains the system and its operating context
- Treat the system of interest as a black box to avoid delving into the details
  - Engineers can decompose the system and apply the same type of modelling and analysis at progressively deeper levels after the top level system is understood.
- The operating context includes anything outside the system boundary that is necessary for modelling the system interactions
  - Friendly systems, threats, natural environment, etc
  - Only include what is necessary for the analysis -e.g. the atmosphere

Domain Model

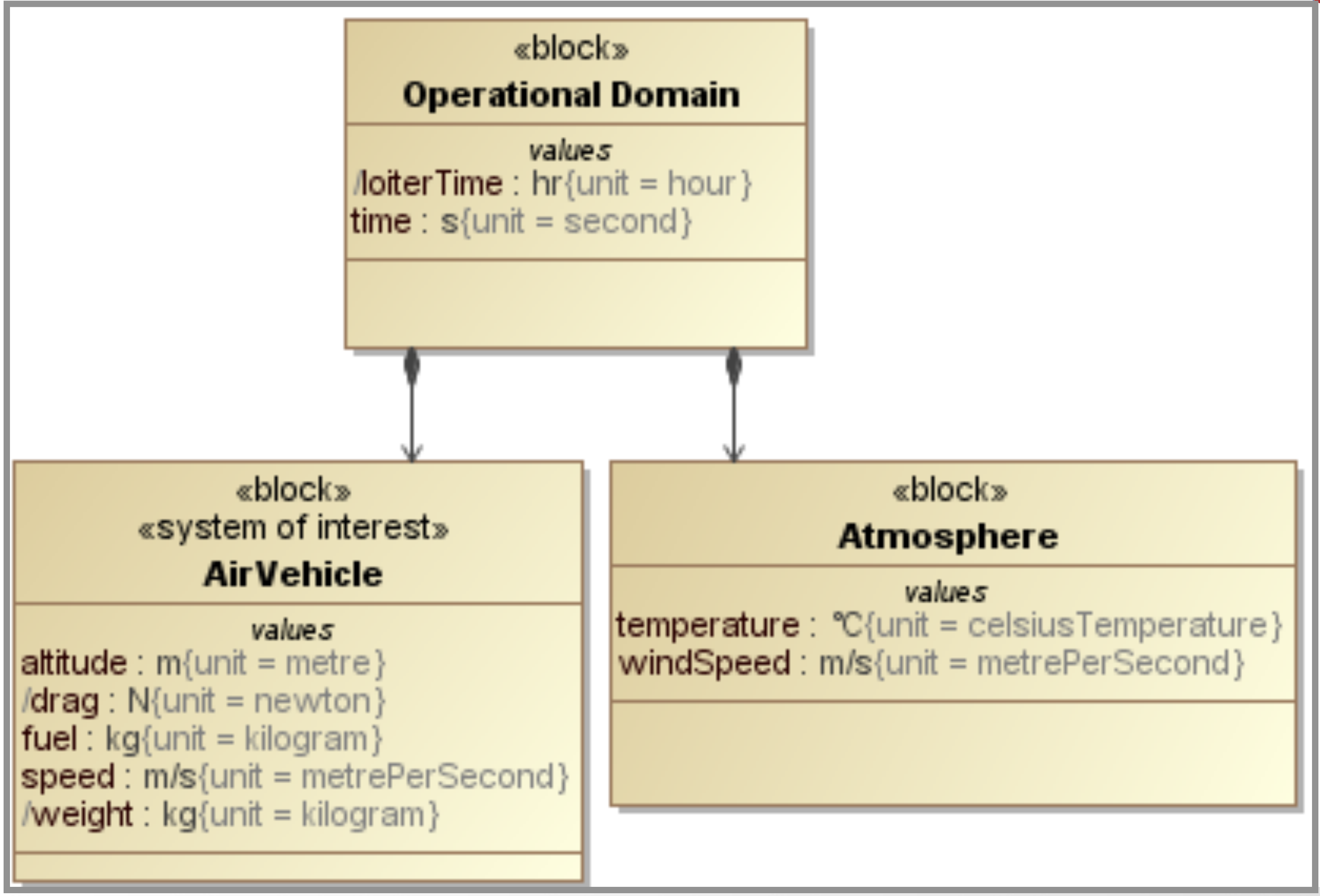


Blocks and Value Properties

- Research on the duration of an aircraft flying a racetrack pattern reveals that fuel consumption important
- Drag, weight, altitude, speed, wind, and air temperature are major impacts to fuel consumption rates
- Air vehicle includes altitude, drag, speed, weight and fuel
- Loiter time is a value property of the Operational Domain
  - derived from values in the air vehicle and operating context
- Operational Domain includes time
  - many of the value properties in the model depend on time
    - weight of the air vehicle which changes as fuel is burned
- Not only is the loiter duration affected by characteristics of the air vehicle, it is also affected by the environment
- Our systems do not operate in a vacuum

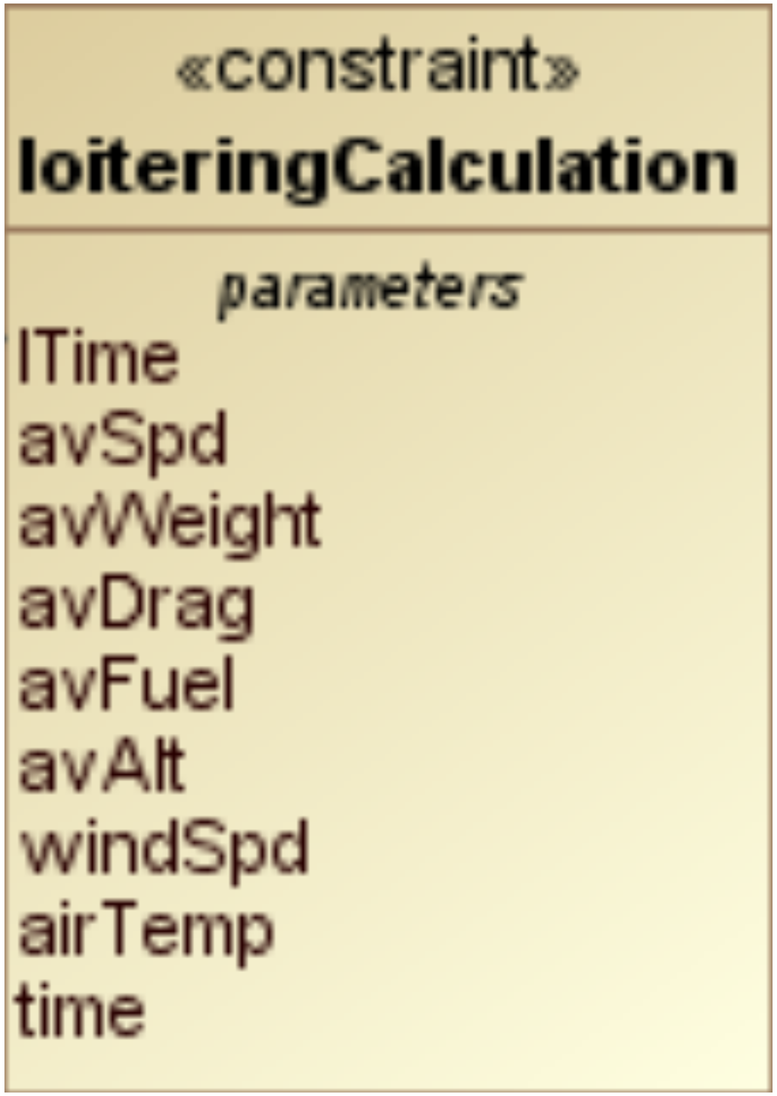


Add System Characteristics to model


«block»
Operational Domain
values
/loiterTime : hr{unit = hour}
time : s{unit = second}
«block»
«system of interest»
AirVehicle
values
altitude : m{unit = metre}
/drag : N{unit = newton}
fuel : kg{unit = kilogram}
speed : m/s{unit = metrePerSecond}
/weight : kg{unit = kilogram}
«block»
Atmosphere
values
temperature : °C{unit = celsiusTemperature}
windSpeed : m/s{unit = metrePerSecond}

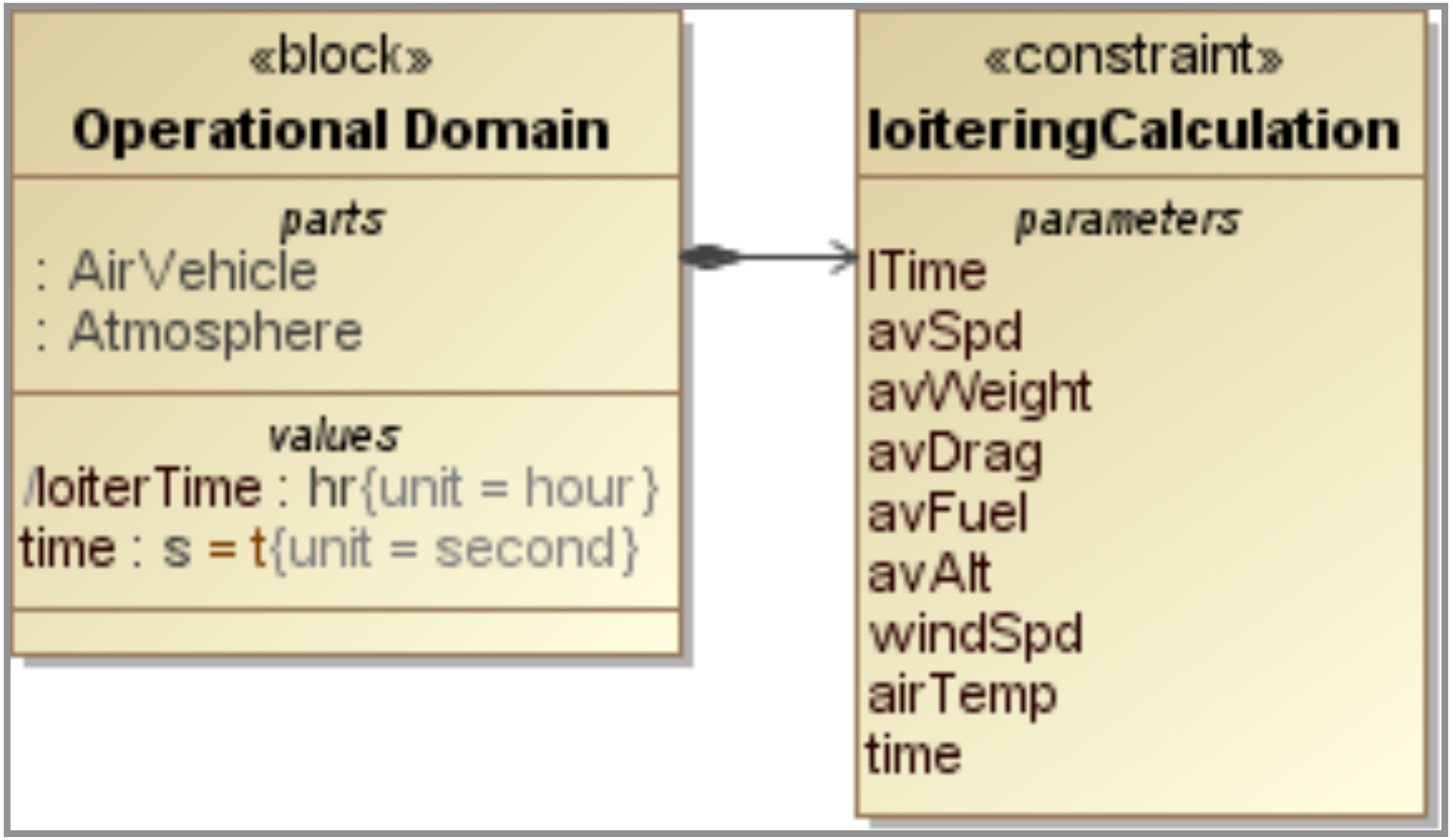
Define Constraint Block

- Define a constraint block that is a part of the operational domain
- Include the constraint parameters that are needed for calculating the duration of the loiter time
- The constraint parameters correlate to the system attributes and will be bound to them in the next step



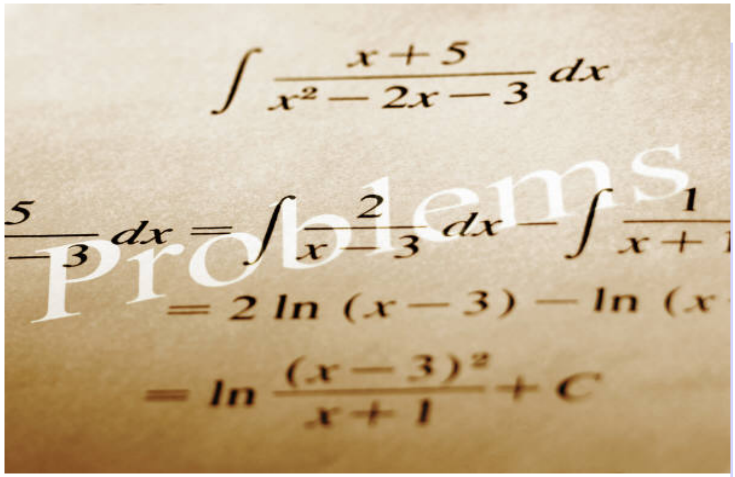
Define Constraint Block

- Constraint block belongs to the operational domain

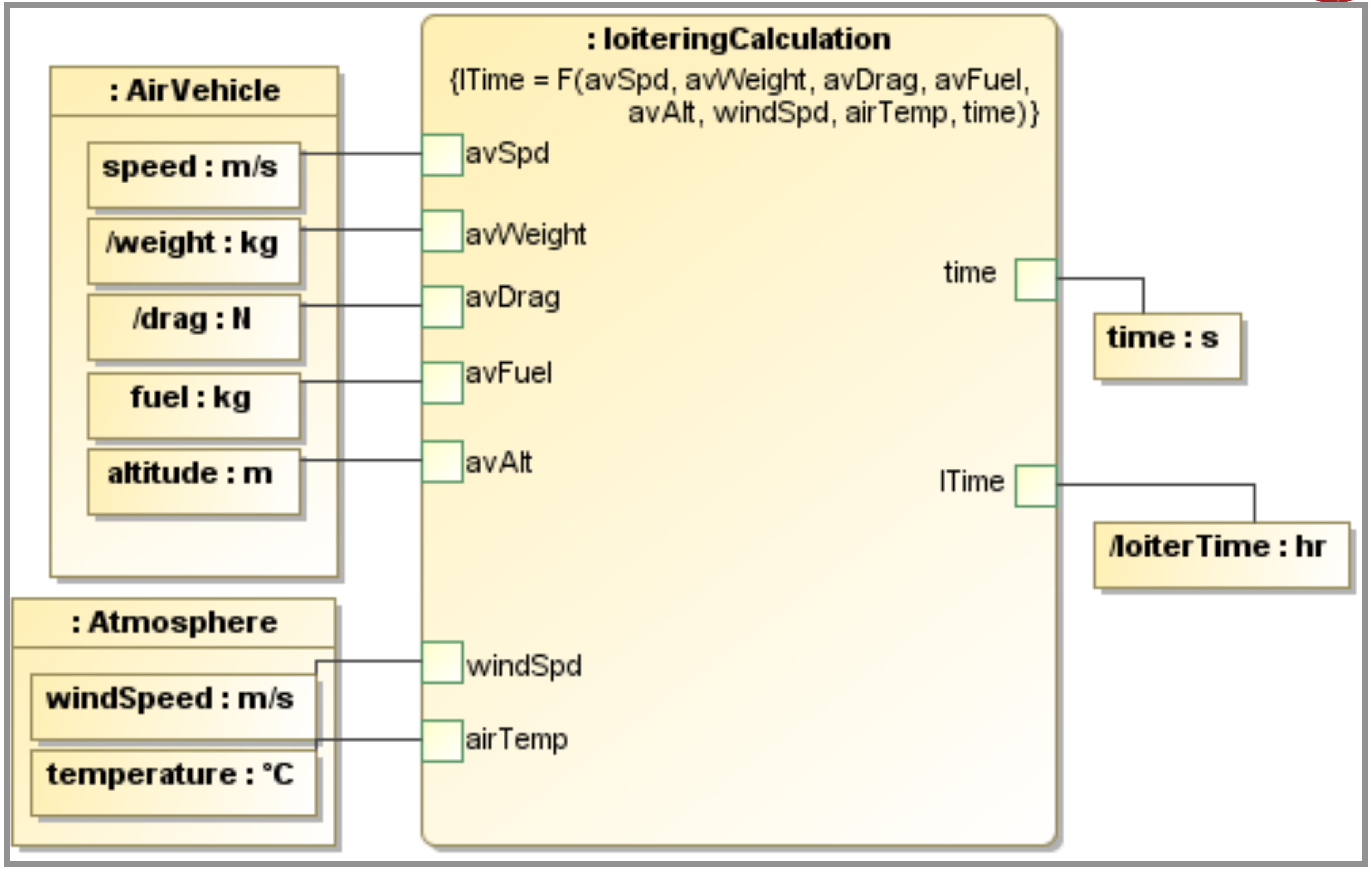


Develop the Parametric Diagram

- Define a mathematical function to compute the duration of flight
- Construct a parametric diagram
- Set the duration to five hours
- Set whatever inputs are known
  - altitude and speed
- If there are any degrees of freedom after setting all of the known variables, optimize for the best values that meet all of the constraints
- If there are no degrees of freedom
  - Solve the equation and determine if there are any feasible solutions
- Value properties can have distributions for a sensitivity analysis
- Perform trade-offs if any of the variables have margin



Parametric Diagram



Update Requirement

- Once all of the values for the constraint parameters are bounded, update the requirement
- Include all of the conditions determined in the parametric diagram in the text of the refined requirement to constrain the initial loiter statement
  - maximum speed and minimum altitude
- Include environmental conditions if needed
  - Weather conditions, wind speed, air temperature

Update Requirement

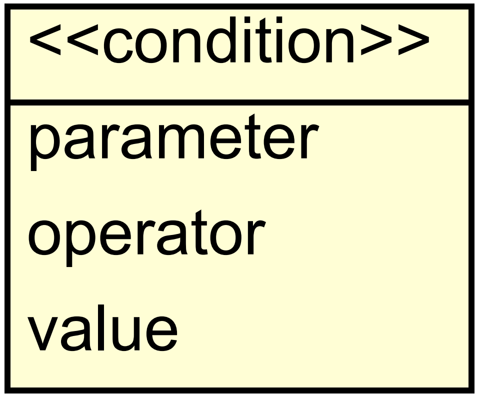
Current Requirement

"The air vehicle shall be capable of loitering for five hours given a minimum speed of x, minimum altitude of y, maximum weight of z, air temperature between w and v, and wind speed less than u."

- Include all of the pieces in the requirement together
- The air vehicle has to be able to loiter under all of the stated conditions simultaneously to meet the customer need.
- Splitting the requirement is a mistake because it allows engineers to verify the pieces individually and pass the requirements if the aircraft can fly at the required speeds one day, the required temperature another day, fly for five hours at one speed, but pass verification for flying at higher speeds another day for fewer than five hours.

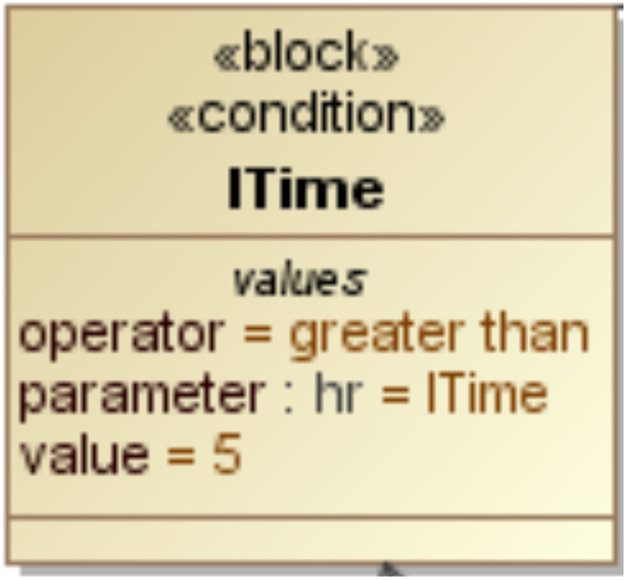
New Stereotype

- Instead of putting the conditions in the text of the requirement block, capture them as model elements that can be integrated with the rest of the model
- Adds precision
- Ensures consistency
- We define a new stereotype called “condition”
  - Includes properties that must be met in order for the system to successfully pass the requirement
  - Value properties are:
    - Operator
    - Parameter
    - value



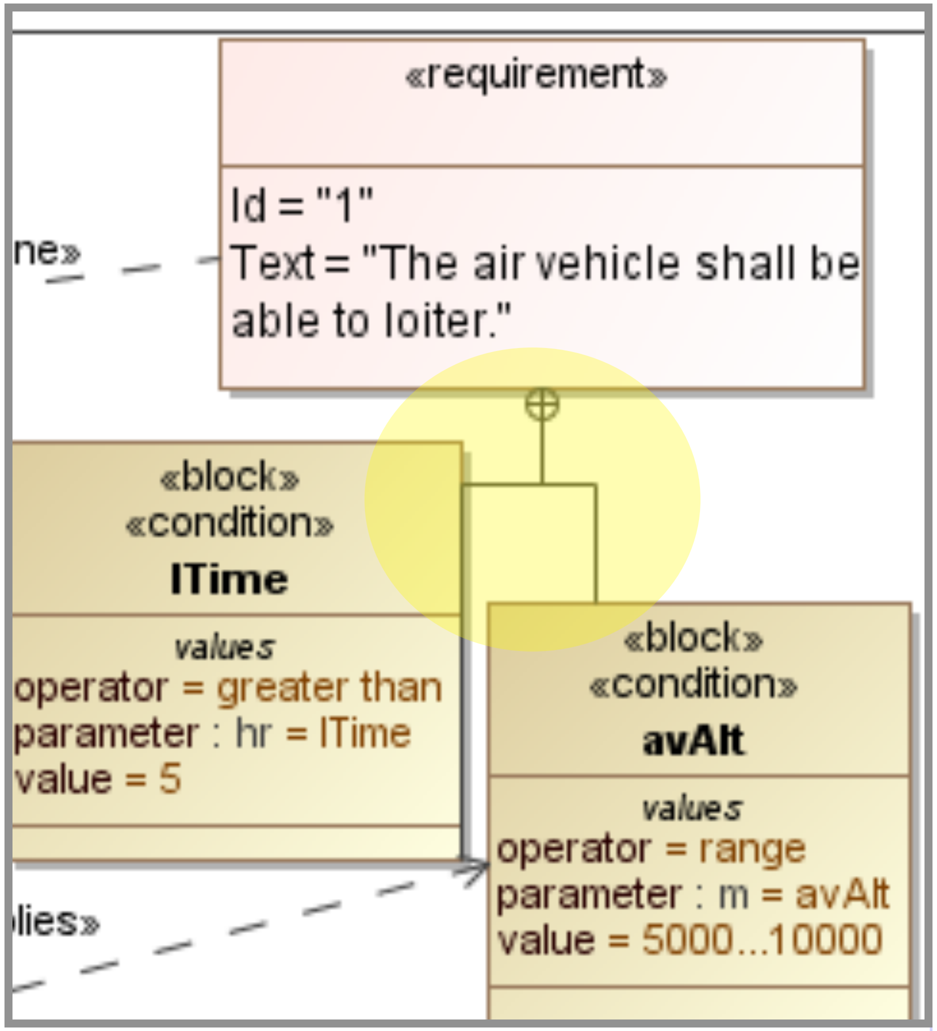
Properties of Condition

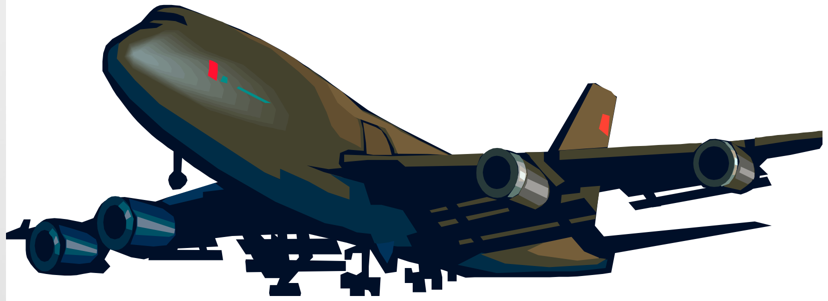
Name	Definition
parameter	A value property of condition that is bound to a constraint parameter. It has units and is constrained to match a value property of a block within the Operational Domain.
operator	The comparison operation between parameter and value.
value	The value the condition parameter must meet.



Relationships to Condition

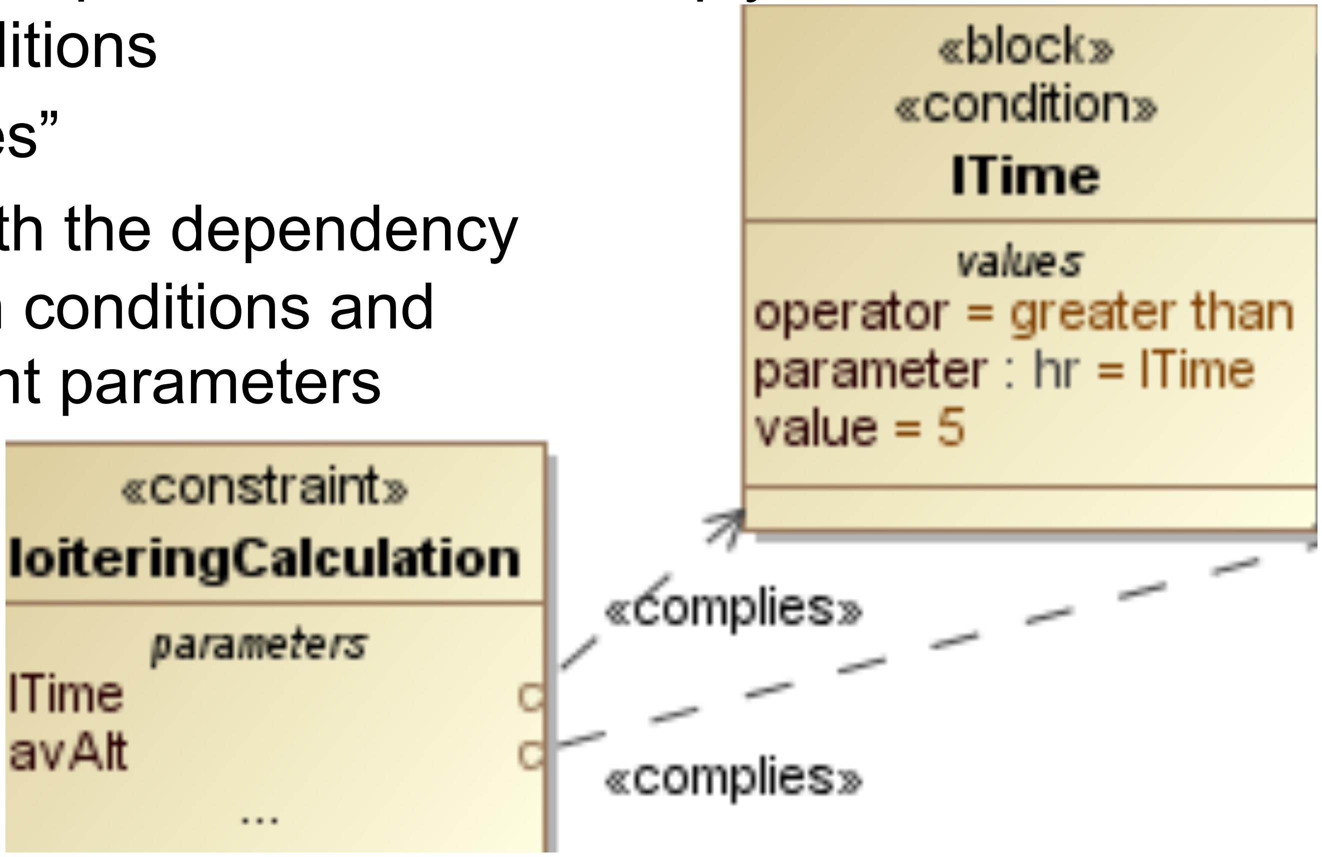
- Relationship between the condition and requirement is “contain”





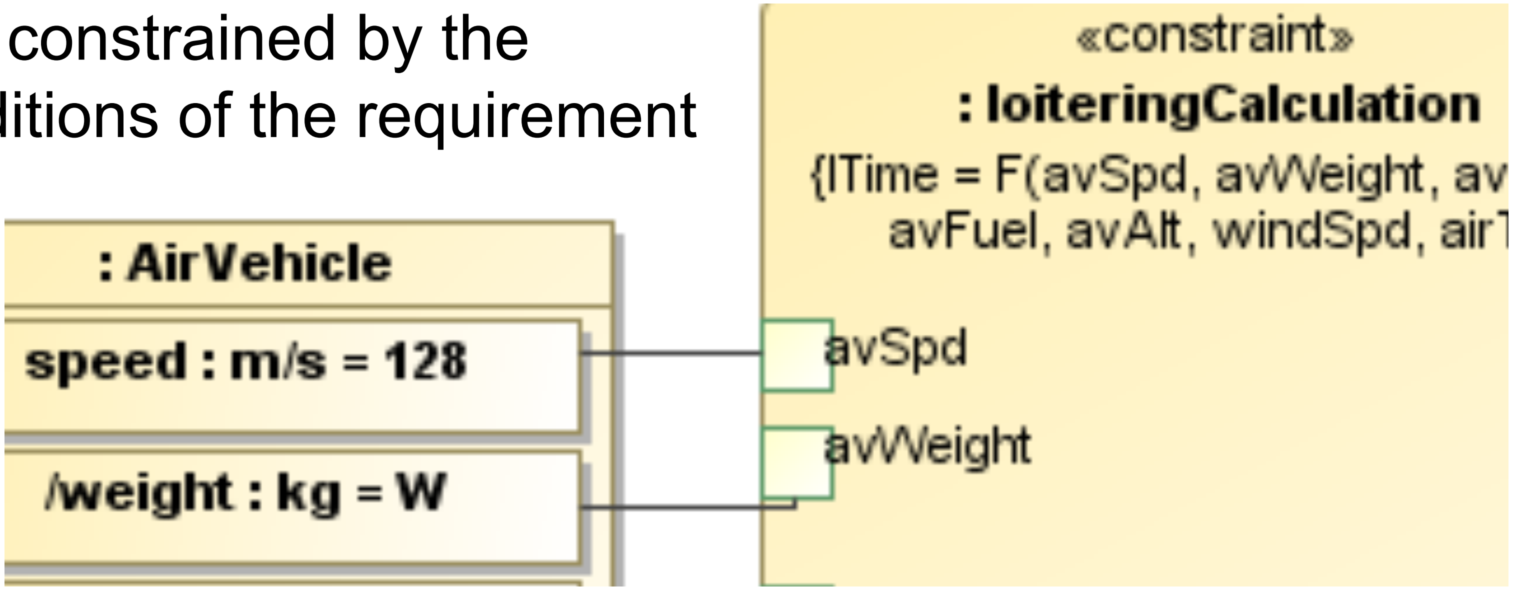
Relationships to Condition

- Need to tie conditions and constraint parameters together
  - Constraint parameters must comply with the values in the conditions
  - "complies"
  - Used with the dependency between conditions and constraint parameters



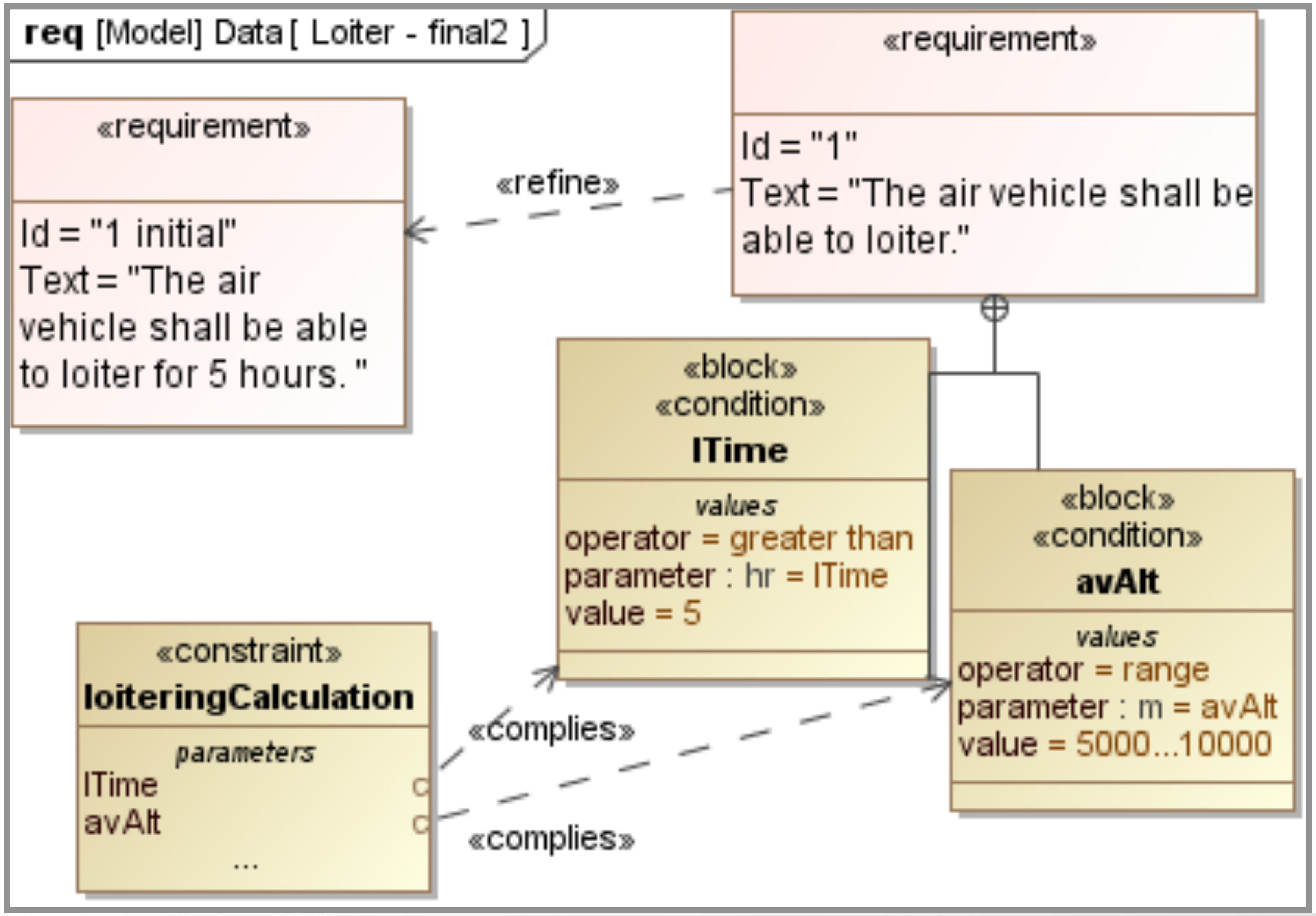
New Requirement Diagram

- Value properties of the system are bound via binding connectors in the parametric diagram and so they are also constrained by the conditions of the requirement

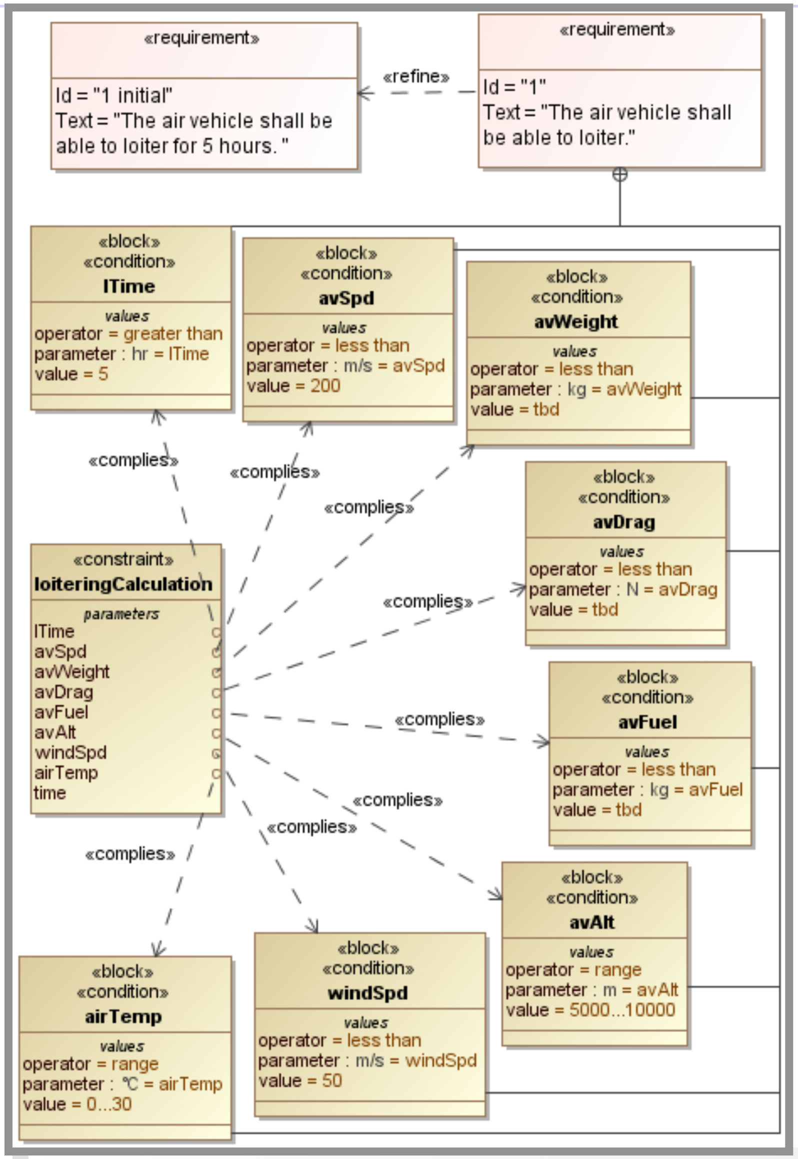


- The next slide shows a requirement that is unambiguous and consistent with the rest of the model.
- Depicts new requirement as a refinement of the original requirement given to the engineers by the customer

New Requirement Diagram (Zoomed in)



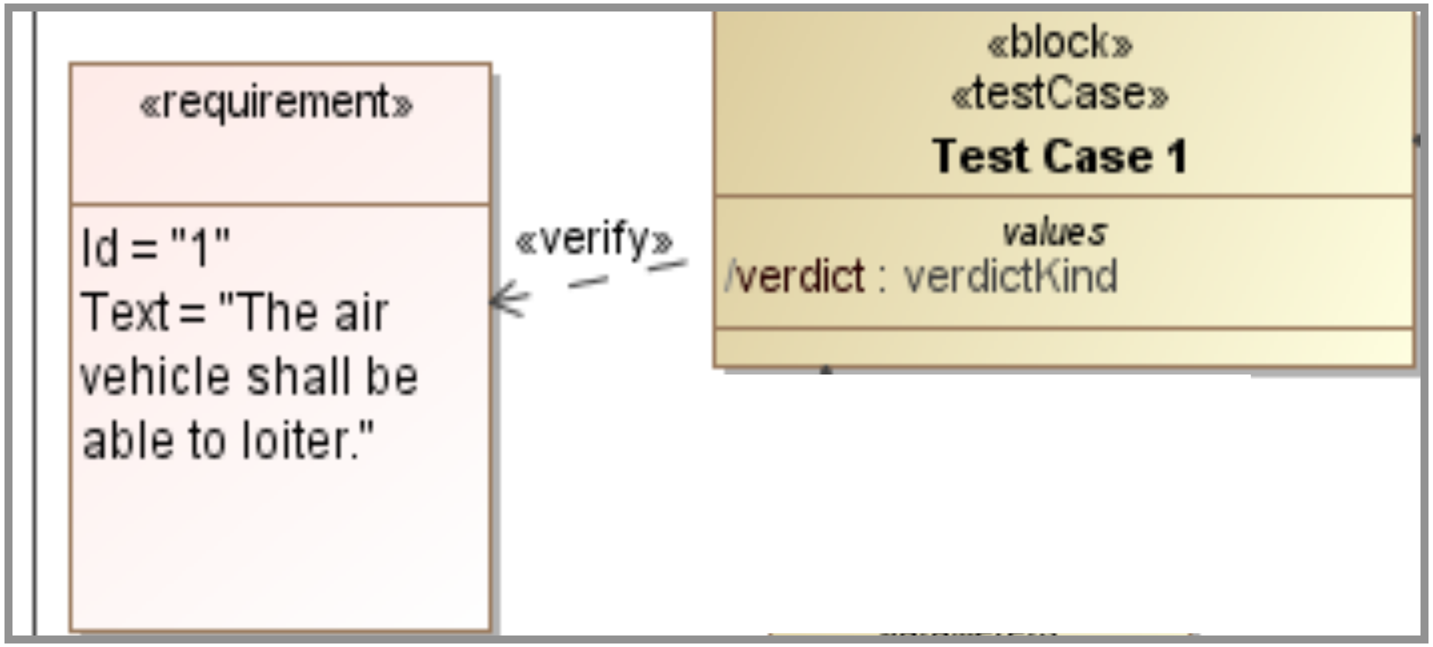
Full Requirements Diagram



- If the constraint equation cannot be solved, verification will fail.
- Another benefit is that a script can be written that follows the satisfy relationship to the system and the value properties of that system to the constraint parameters to which they are bound. It then could follow the containment relationship to the conditions to the constraint parameters, and check that the system meets the conditions to satisfy the requirements.
- Requirement statements can be auto generated

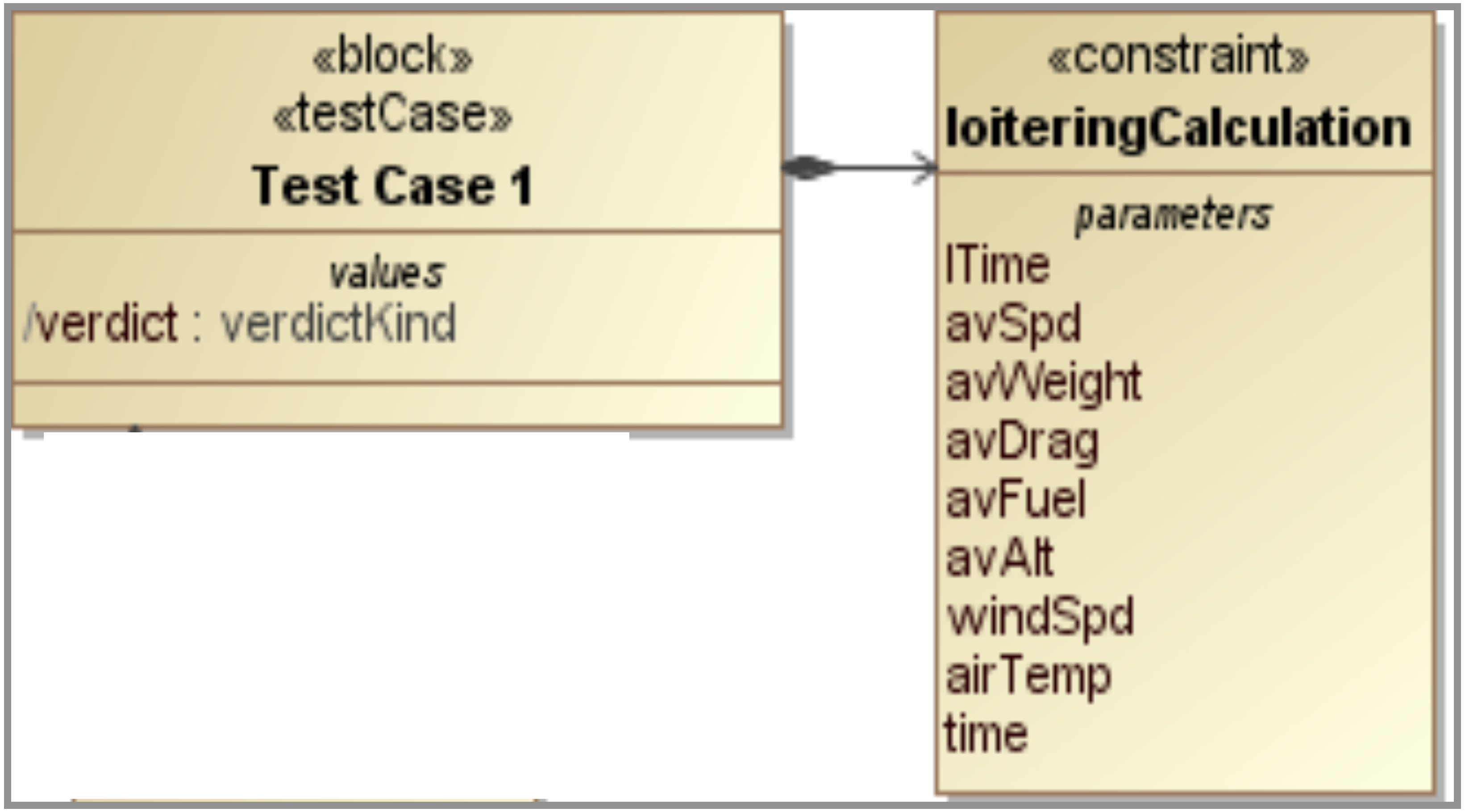
Verification

- Include the verification of requirements in model
- Model has conditions that need to be exercised
- Construct a requirement diagram with a new block representing a test case that verifies the requirement



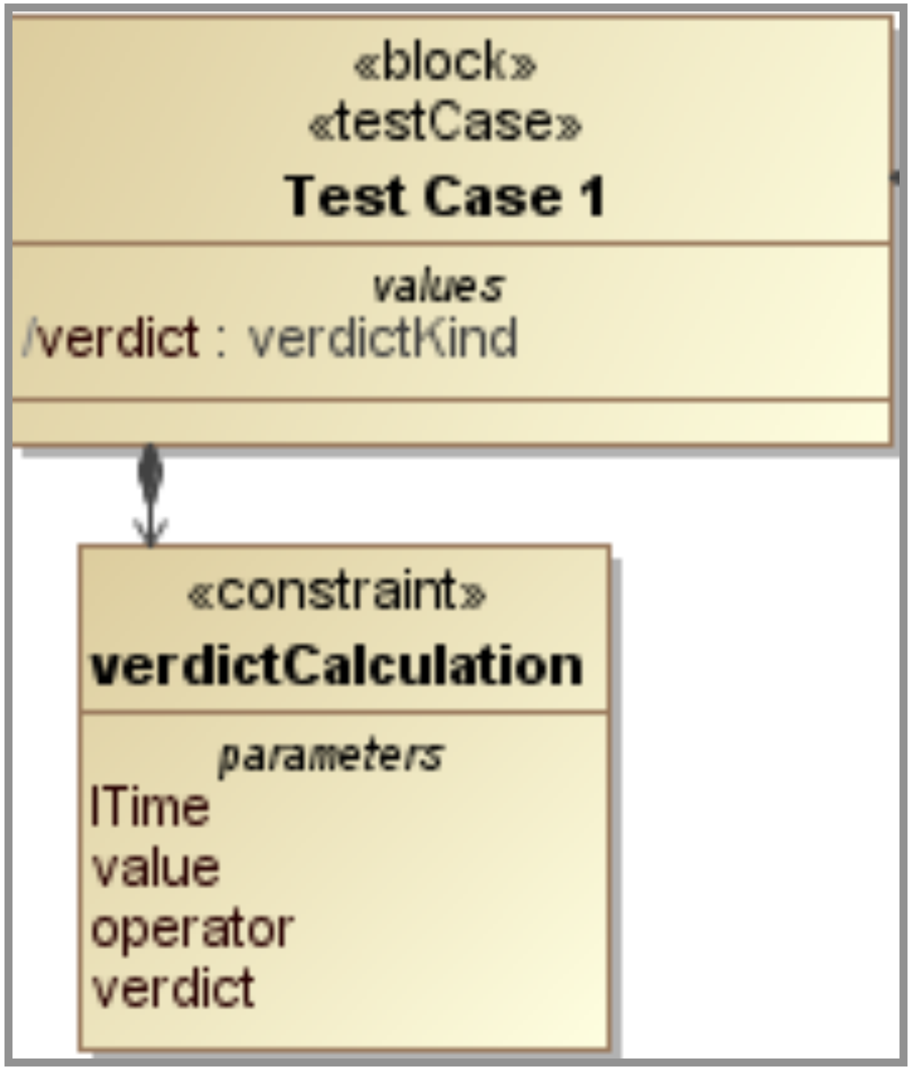
Verification

- Test case uses the same loitering calculation that was defined during the design process

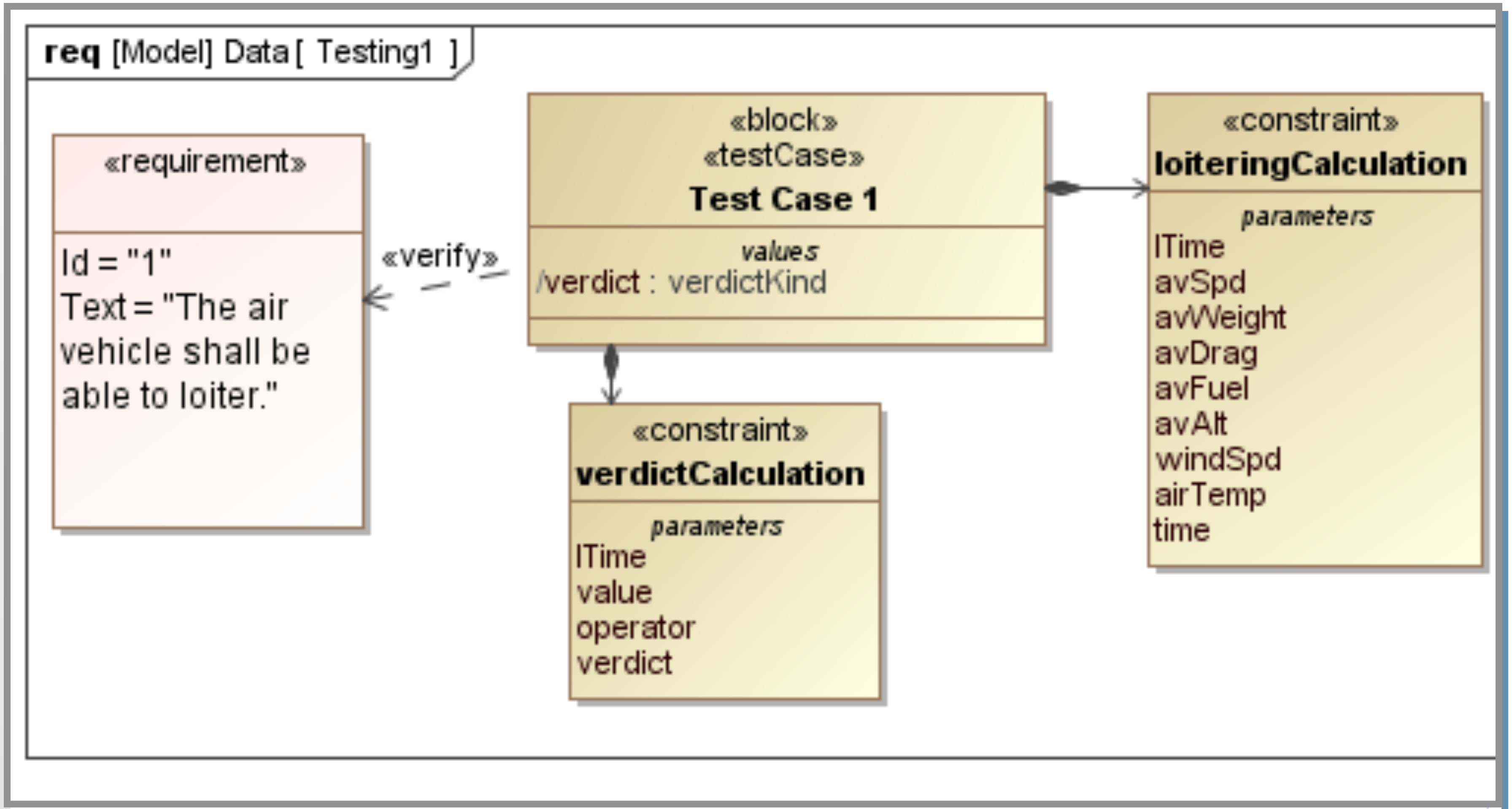


Verification

- Test Case has a constraint that sets the verdict of the test case based on the output of the loitering calculation
- Create as many test cases as needed
- Use the model for verification by analysis using solvers or for verification by test by using the model to define the conditions of the test and to record the results



Verification all put together

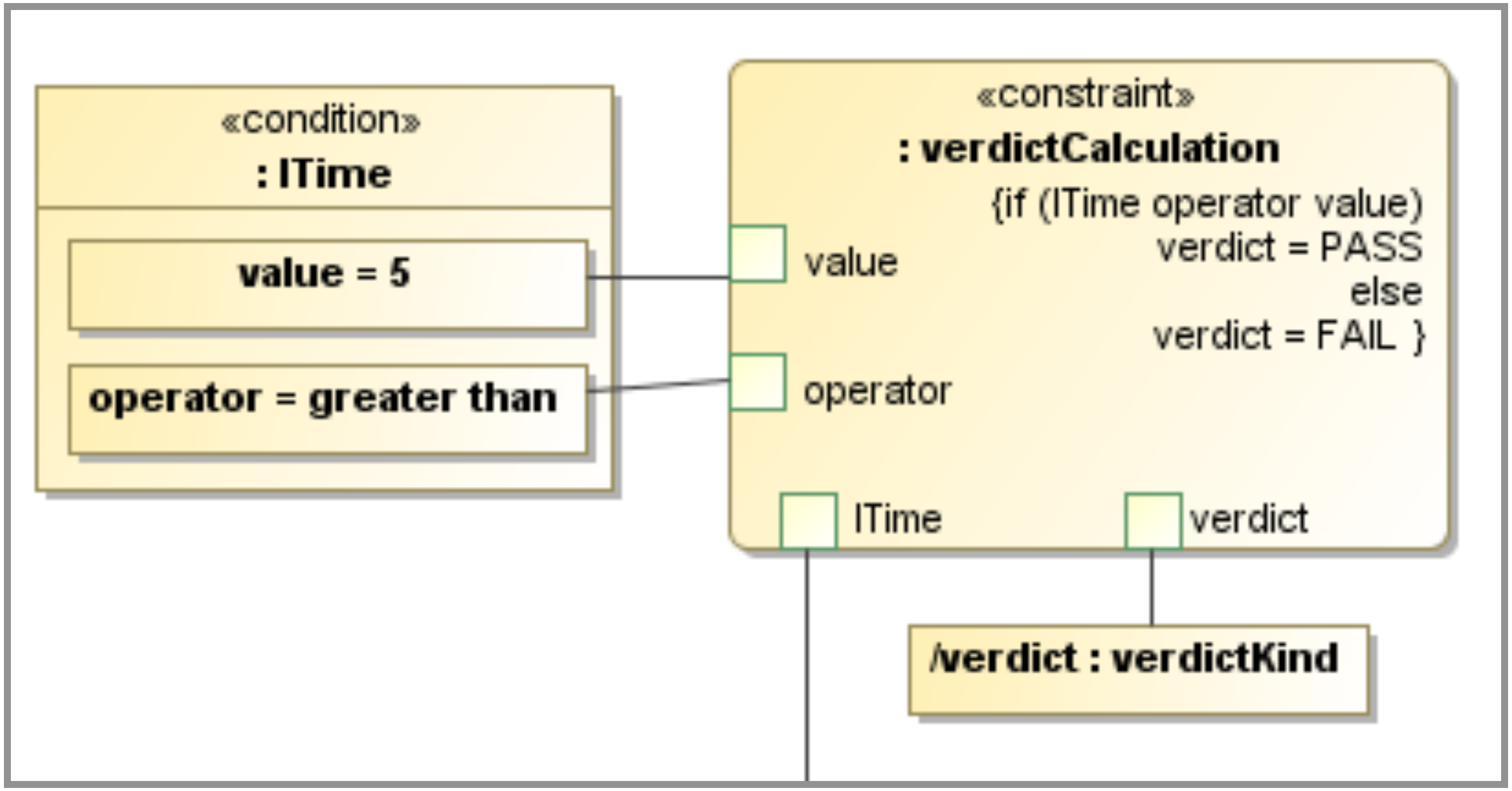


Verification Parametric

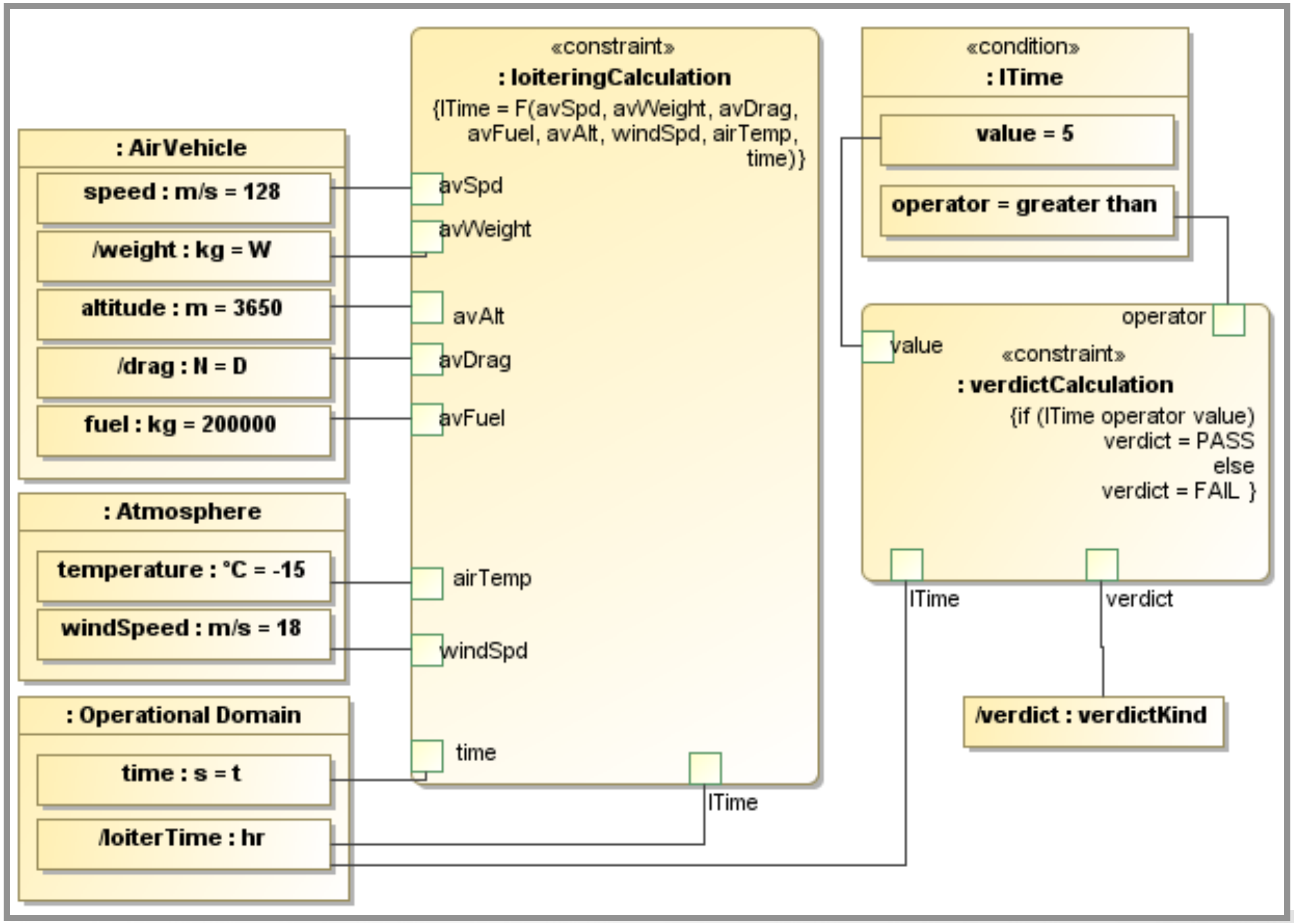
- Test case has a parametric diagram that shows the relationships among the value properties of the system, the environment, the operational domain, and verdict
- The duration of the loiter time is measured during test or calculated during from simulations
- The constraint inside the verdict calculation is formed from the condition that is contained by the requirement
- Value properties are set or measured and the constraint is used to solve for the loiter time
- Inputs are test conditions that are a part of the test set up
- Only unknowns are loiter time which is the output of the loitering calculation and the test case verdict which is the output of the verdict calculation

Verification Parametric (Zoomed in)

- If the output of the loiter time meets the condition, the verdict is set to PASS.



Verification Parametric



Summary

- Demonstrated a technique to develop requirements that are unambiguous, consistent, and complete with traceability to the system satisfying the requirements
- Introduced two new stereotypes to extend SysML in order to capture the textual parts of the requirement within the model to help with ensuring consistency
- Refined a vague requirement into a complete requirement that addresses the conditions under which it can be met
- Enabled consistency checks via scripts
- SysML Parametrics can be used with Model-Based Systems Engineering processes to develop more precise requirements and perform requirements analysis

References

- Bittner, K. 2008. When Requirements Go Bad. Ivar Jacobson Consulting Whitepaper.
- Estefan, Jeff. 2008. *Survey of Model-Based Systems Engineering (MBSE) Methodologies*. International Council on Systems Engineering.
- Fisher, G. 2002. Model-Based Systems Engineering of Automotive Systems. Digital Avionics Systems Conference, 1998. Proceedings, 17th DASC.
- Friedenthal, S. Moore, A. Steiner, R. 2008*. A practical guide to SysML: Systems Model Language*. Morgan Kaufmann Publishers.
- Gilb, Tom. 2010. *What's Wrong with Requirements Specification? An Analysis of the Fundamental Failings of Conventional Thinking about Software Requirements, and Some Suggestions for Getting it Right*. Journal Software Engineering & Applications.

References

- Hoffmann, H. 2010. Systems Engineering Best Practices with the Rational Workbench for Systems and Software Engineering Harmony Deskbook Release 3.1.1, IBM.
- Hood, C. Wiedemann, S. Fichtinger, S. Pautz, U. 2008. Requirements Management, The Interfaces Between Requirements Development and All Other Systems Engineering Processes. Springer-Verlag Berlin Heidelberg.
- Hooks, Ivy. 1993. *Writing Good Requirements*. Proceedings of the Third International Symposium of the NCOSE – Volume 2.
- Karban, R. Andolfato, L. Zamparelli, A. 2009. Toward Model Re-usability for the Development of Telescope Control Systems. ICALEPCS 2009. http://www.omgsysml.org/#Publications

References

- OMG Systems Modeling Language version 1.2. 2010. http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf
- Weilkiens, T. 2007. *Systems Engineering with SysML/ UML: Modeling, Analysis, Design*. Elsevier Science and Technology Books, Inc.