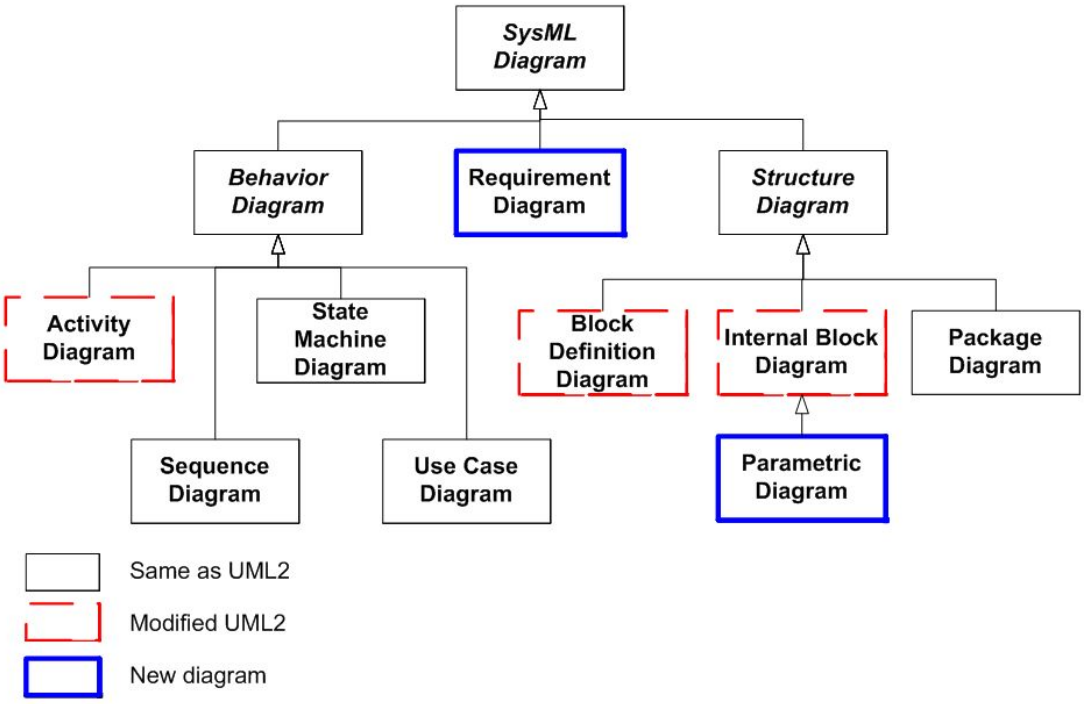
Street-Fighting Guide for Cameo Systems Modeler

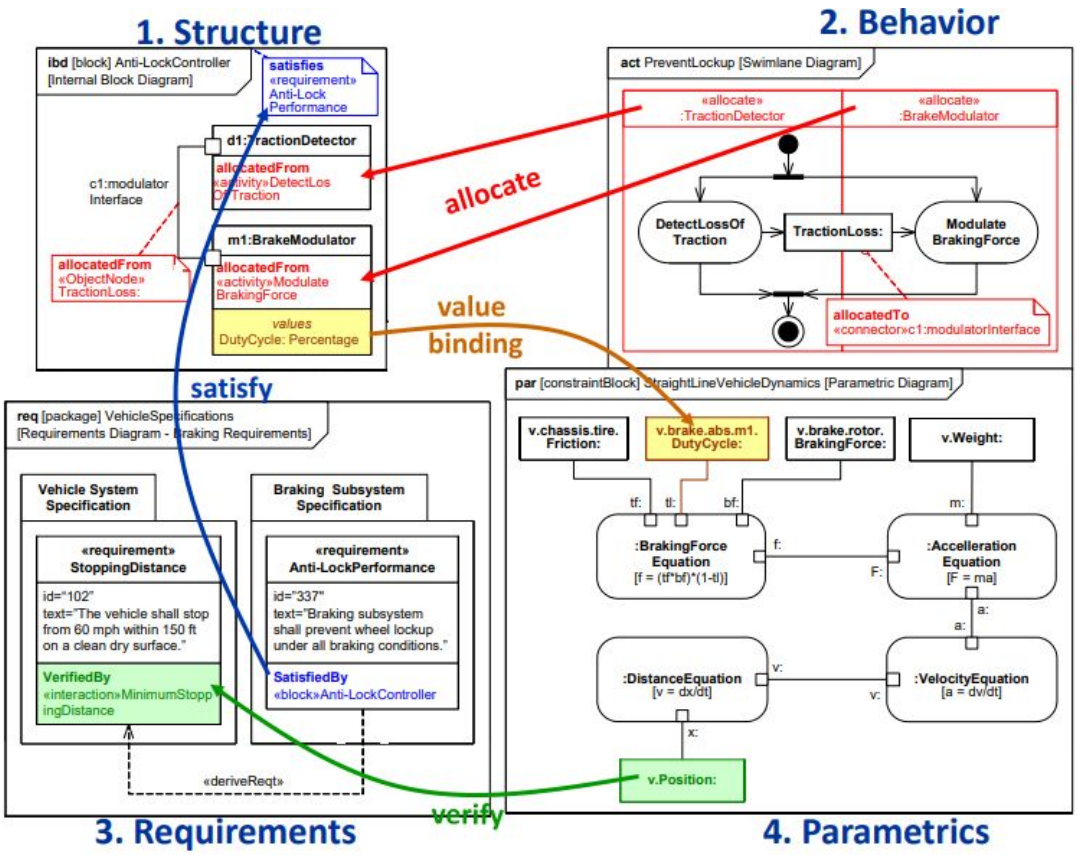
Introduction:

SysML is a general-purpose architecture modeling language for System Engineering applications. SysML supports the specification, analysis, design, verification of a board range of systems and systems-of-systems. The following figure illustrates the types of diagram that SysML consist of.



SysML Diagram Taxonomy

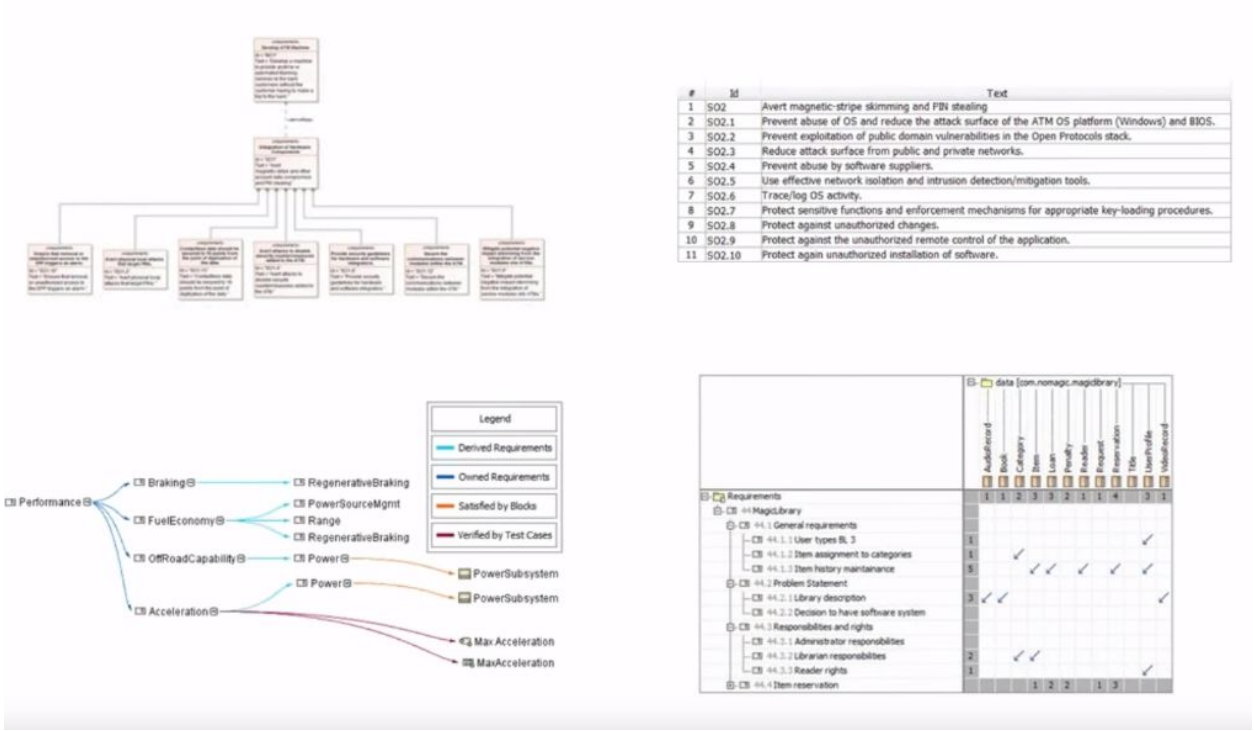
The 4 pillars of SysML are



Requirements Diagram:

The layout of the requirements can be broken down into 4 main categories in SysML; they are:

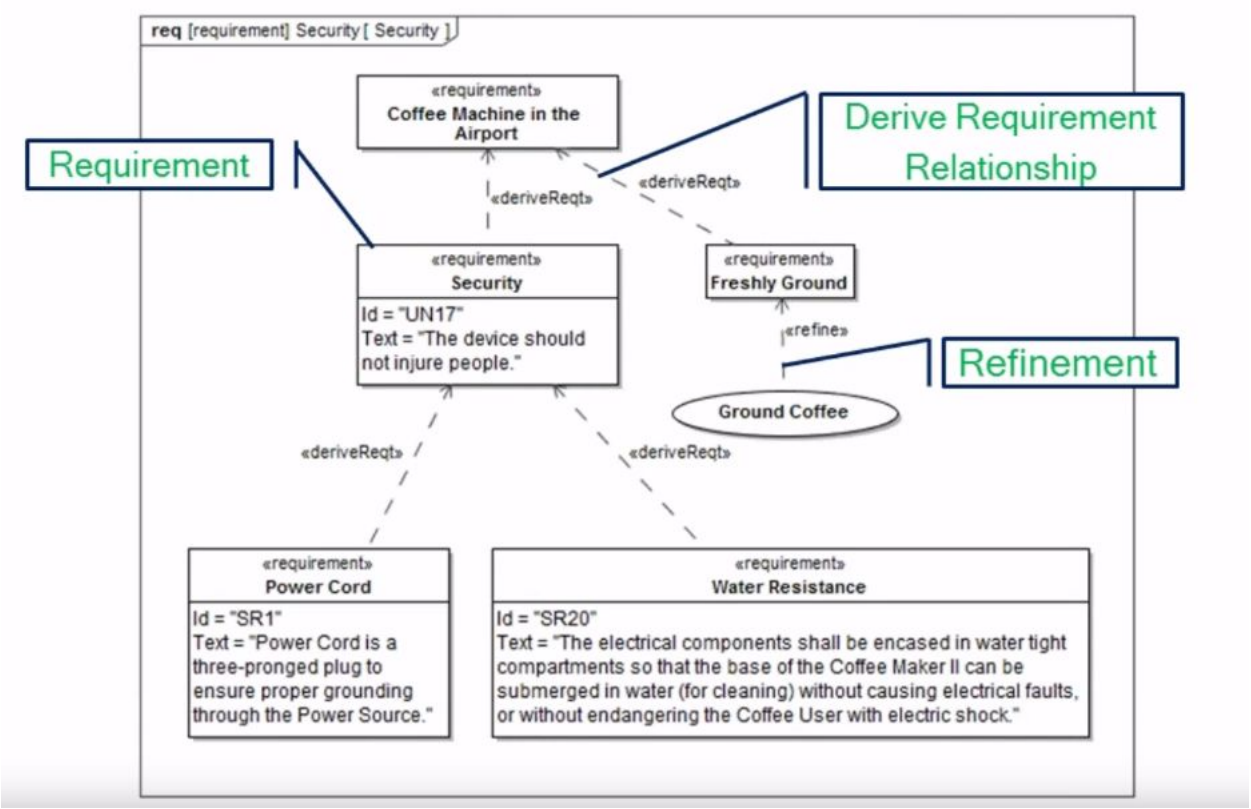
1) Requirement Diagram (block level illustration of the requirements)
2) Requirement Table (tabular view of the requirements)
3) Requirement Containment map (a tree-like illustration)
4) Requirement Matrices (Req2Req or Req2Struct)



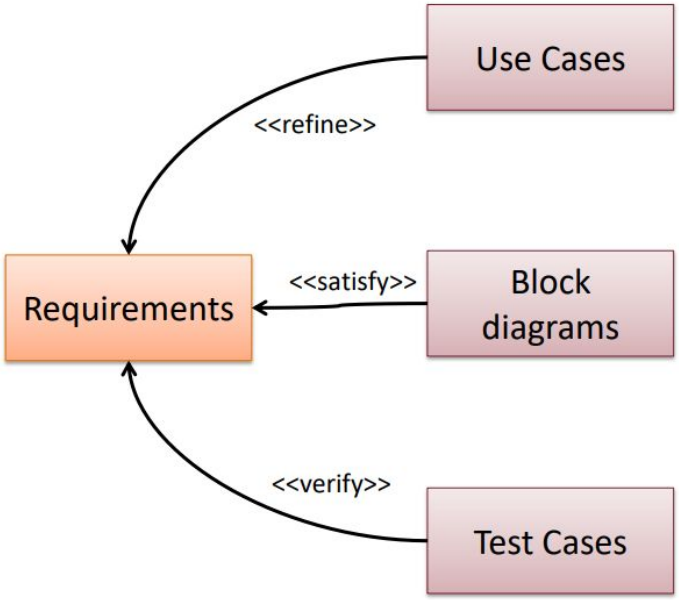
#	Id	Text
1	SO2	Avert magnetic-stripe skimming and PIN stealing
2	SO2.1	Prevent abuse of OS and reduce the attack surface of the ATM OS platform (Windows) and BIOS.
3	SO2.2	Prevent exploitation of public domain vulnerabilities in the Open Protocols stack.
4	SO2.3	Reduce attack surface from public and private networks.
5	SO2.4	Prevent abuse by software suppliers.
6	SO2.5	Use effective network isolation and intrusion detection/mitigation tools.
7	SO2.6	Trace/log OS activity.
8	SO2.7	Protect sensitive functions and enforcement mechanisms for appropriate key-loading procedures.
9	SO2.8	Protect against unauthorized changes.
10	SO2.9	Protect against the unauthorized remote control of the application.
11	SO2.10	Protect again unauthorized installation of software.

What is a requirement?

A basic requirement consists of the name, id, and description. A requirement can be a derive of another requirement.



A refinement clarifies a requirement with an “action” (functional objective) from the behavior pillar. The traceability of requirements in SysML models can be illustrated as follow:



Validation and Verification

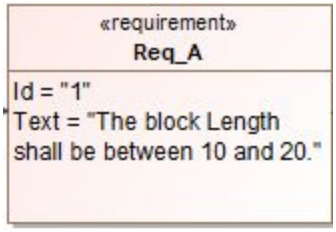
There are two forms of validation in SysML.

1) The Default Validation - Validate syntax and user-defined requirement constraints.
2) Test Cases

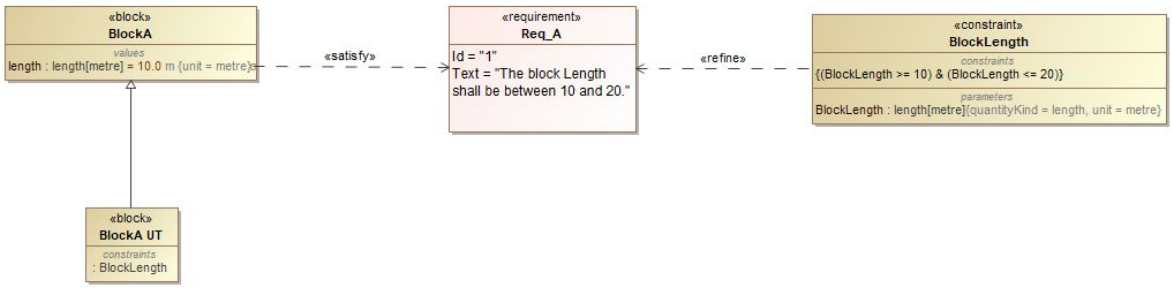
One of the powerful features of SysML is to build executable models. This allows users to better maintain and use of a business model. Traditionally, requirements are defined and are matched manually. As the complexity of the model grows, it becomes difficult to maintain.

Model Compliance Verification

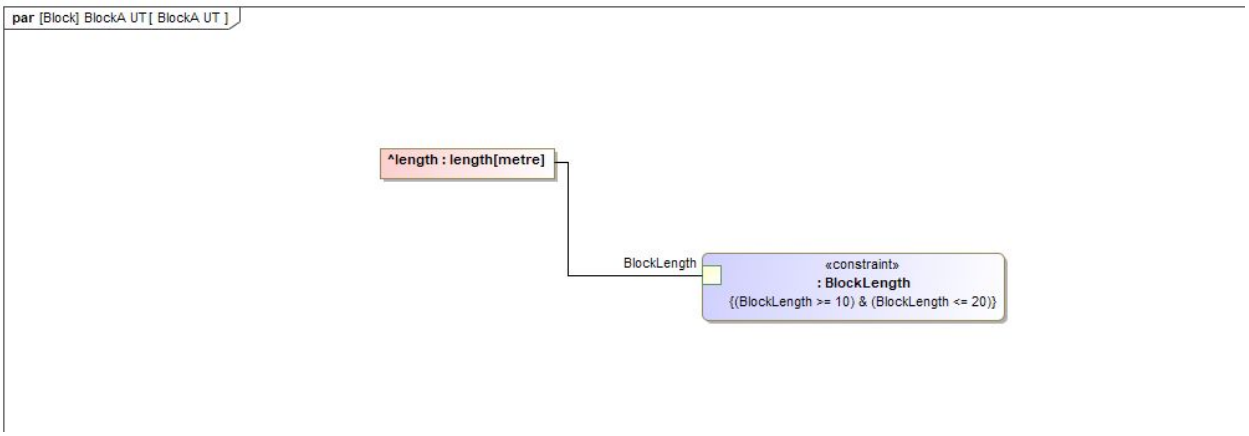
In SysML, a requirement is defined in the context of high-level human language.



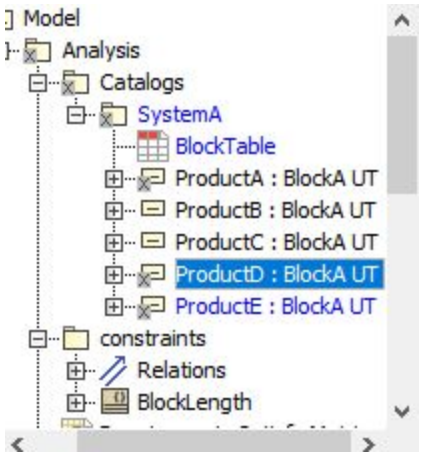
In the below diagram, the requirement block is “refined” by an actual constraint. This constraint contains a parameter and logic. A block defines the Scafell of the design. The BlockA UT (Under Test) is a realization of the Scafell.



This realization is defined inside of an “analysis” folder. Next, within the Block UT, add a parametric diagram and bind the constraint like this.



Under the analysis folder, create the “actual” product such as the following.



These realized products will be the actual block being tested. Finally, create a instance table and drag the instances to the table. Add the constraint column to the table.

Criteria

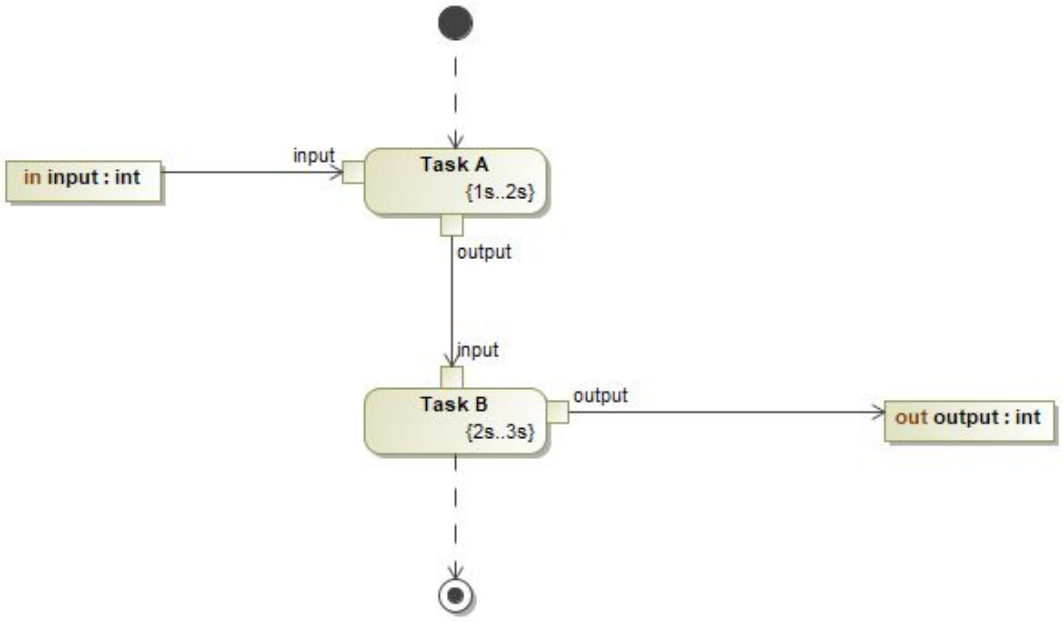
Classifier: BlockA UT ... Scope (optional): BlockLength ... Filter:

#	Name	length : length[metre]	: BlockLength
1	ProductA	9 m	fail
2	ProductD	1 m	fail
3	ProductE	30 m	fail
4	ProductB	12 m	pass
5	ProductC	10 m	pass

As one can see, for products with block length outside of the defined range will fail the test. This easily allows one to verify the compliance of the model.

Duration Analysis

This analysis allows users to investigate the duration of various activities. Activity diagram allows the user to define duration constraints. This powerful capability enables better capture of time-critical tasks.



Unit Test

This analysis allows users to verify and test activities. Unit tests do not seem to be matching JUnit Test in Java. It seems that it is just another contextual element that used for matching the requirement.

Behavior Pillar

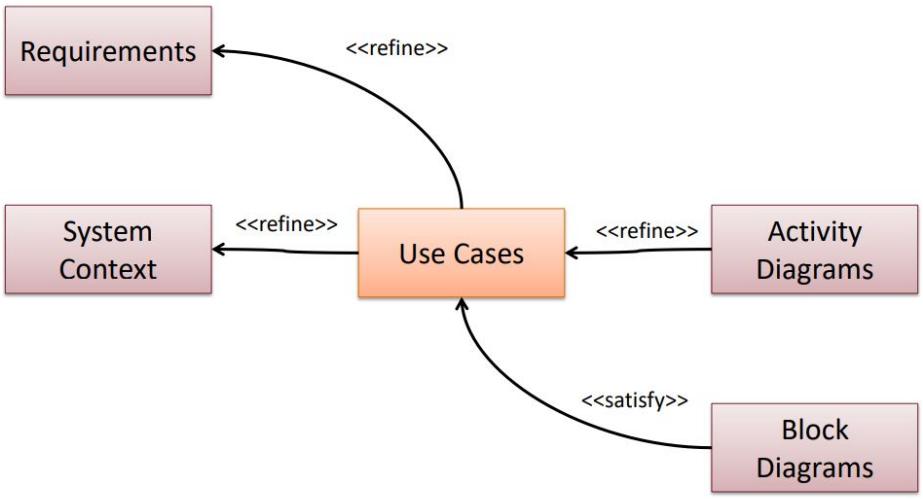
The behavior pillar consists of a few types of diagrams. They are the sequence diagram, the activity diagram, and the finite-state machine.

Note: Whenever a state machine transacts from one state to another, it requires the use of a “signal” element.

Prototyping - TODO coming soon.

This allows users to prototype each idea.

Traceability of Use Cases



Note: In order to use Unit Test, users are required to install a Testing Plugin.

Reports

There are existing report templates in Cameo. However, if one needs to customize a template, users are required to build his/her own template. Cameo uses Velocity Template Language (VTL). Please consult the reference for syntax. To incorporate the customized report, click Report Wizard then “new” and set the template. Then, add new variables.

References

1) https://inf.mit.bme.hu/en/edu/courses/remo-en/materials
2) https://docs.nomagic.com/display/CSTTWRT/Using+simulation+command+line+and+showing+test+results+through+Jenkins
3) http://velocity.apache.org/engine/1.7/vtl-reference.html
4)