Mechanics Modeling Language (MechML)

Version 1.0

June 2016

Scope

This specification defines the Mechanics Modeling Language (MechML), revision 1.

The objective of MechML is to provide systems engineers, system architects, mechanical engineers, and draughtsman with a modeling language with which design-related aspects of a technical system can be modeled in a SysML model. One of the primary goals of MechML is to support all engineering activities that are required to apply the FAS4M (Functional Architectures of Systems for Mechanical Engineers) method. MechML was developed as part of the research project FAS4M, which was funded by the German public sector (*Bundesministerium für Wirtschaft und Energie*).

Copyright

License

Disclaimer of Warranty

Trademarks

Related Documents

- OMG Systems Modeling Language™ (SysML®), Version 1.4. OMG document number: formal/2015-06-03. http://www.omg.org/spec/SysML/1.4/
- OMG Unified Modeling Language™ (UML®), Version 2.5. OMG document number: formal/15-03-01. http://www.omg.org/spec/UML/2.5/

Table of Contents

1 Language Architecture

MechML is a set of so-called stereotypes that are pooled in a special package, called a profile, and provides additional extensions needed to address requirements in the domain of mechanics and mechanical engineering, especially when applying the FAS4M method.

The MechML profile package imports the Systems Modeling Language (SysML) profile. Since SysML itself is defined as a profile on the Unified Modeling Language (UML), the SysML profile package imports the UML 2 metamodel (see Figure 1 - Package structure and dependencies).

The stereotype «FunctionalBlock» is designed to be compatible with the stereotype of the same name in the profile of the FAS -method[1]. In order to allow using MechML independent of the FAS-profile, the profile is not imported here.

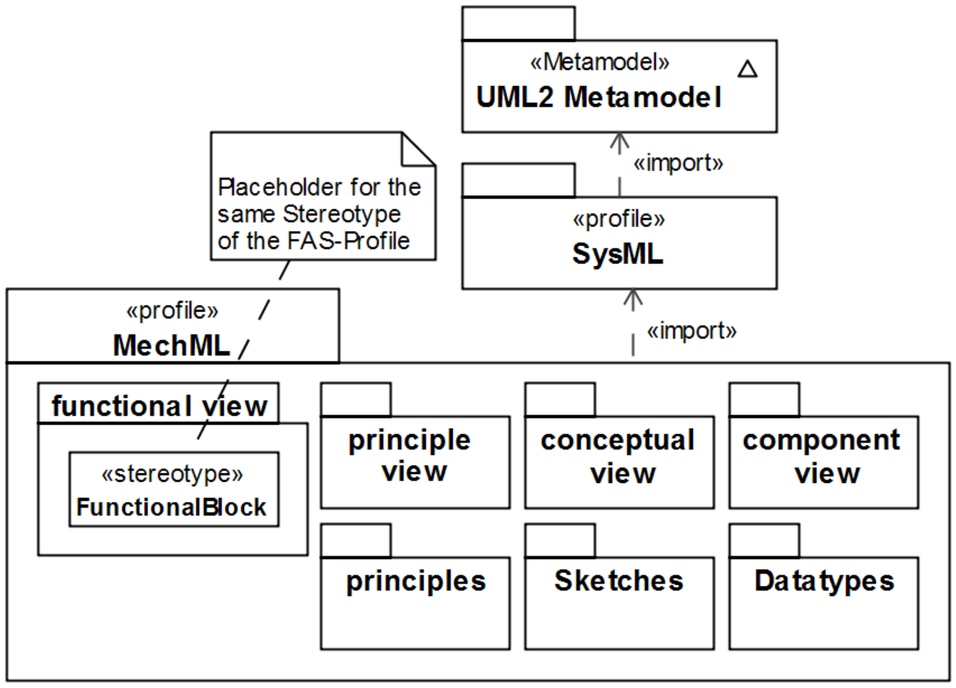

Figure 1 – Package structure and dependencies

The discussion of the languages SysML and UML are out of scope of this specification. These modeling languages are specified and discussed in detail in their respective OMG specifications (see section 0, *Related Documents*).

If not otherwise specified (namely for FreeSketch) the MechML-notation uses the standard visualization that UML defines for applied stereotypes and stereotype attributes.

[1] The FAS-Method is described in „Model-Based Systems Architecture“, Tim Weilkiens, Jesko G. Lamm, Stephan Roth, Markus Walker, Wiley, 2015
The profile is available under http://fas-method.org/content/fas-plugins/

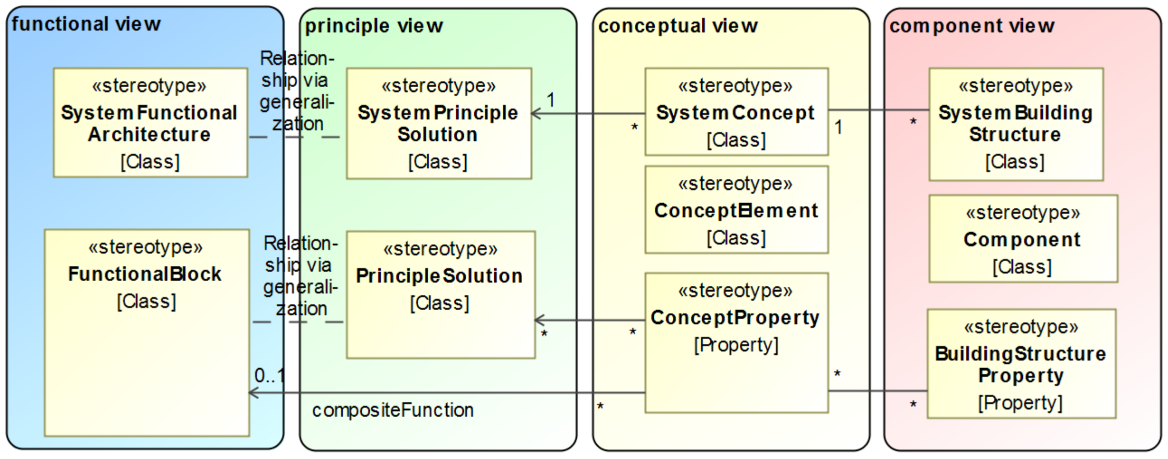
functional view
principle view
conceptual view
component view
«stereotype» System Functional Architecture [Class]
«stereotype» FunctionalBlock [Class]
Relation-ship via generali-zation
Relation-ship via generali-zation
«stereotype» System Principle Solution [Class]
«stereotype» PrincipleSolution [Class]
«stereotype» System Concept [Class]
«stereotype» ConceptElement [Class]
«stereotype» ConceptProperty [Property]
«stereotype» System Building Structure [Class]
«stereotype» Component [Class]
«stereotype» BuildingStructure Property [Property]
compositeFunction
0..1
1
*

Figure 2 – The four views of MechML

An overview of MechML’s most important language elements and their relationships can be found in Figure 2. The stereotypes in this figure are arranged from left-to-right by the methodological views, according to the approach as prescribed by the FAS4M method:

- Deriving a functional architecture from the system’s use cases (Functional View). Creating this view is usually done with the help of the FAS method.
- Opening up the solution space using a morphological box (Principle View).
- Developing an overall concept (Conceptual View) for the system of interest.
- Developing a physical building structure (Component View) of the system of interest.

In each of those four views, exactly one element exists that represents the system of interest according to the level of abstraction of this view.

2 Functional View

2.1 Summary

The Functional View is the view on the highest abstraction level and describes the result of a functional analysis using the FAS (Functional Architectures of Systems) method.

SysML Blocks are defined as modular units of system description. They provide a general-purpose capability to describe the architecture of a system, even from an abstract, technology-independent and pure functional point of view. MechML provides stereotyped Blocks that can be used to create the functional views that, for example, are required by the FAS4M-method[2].

[2] The FAS4M-method is described on http://www.fas4m.de

2.2 Abstract Syntax

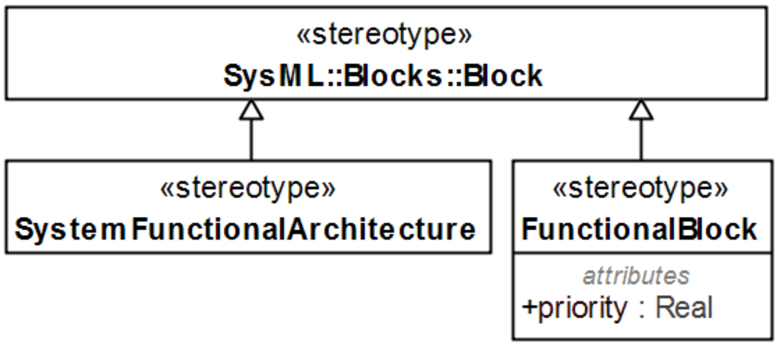

Figure 3 – Functional View

2.3 Definitions

2.3.1 SystemFunctionalArchitecture

The SystemFunctionalArchitecture is an element that represents the system of interest on a pure functional level. Its parts and connectors describe the functional architecture of the system. There can be many SystemPrincipleSolutions that will be a specialization of it.

Relations

- Generalization
  A SystemFunctionalArchitecture may be the target of a Generalization from a SystemPrincipleSolution.

Constraints

- FunctionalBlock as Parts
  {may only contain FunctionalBlocks}

2.3.2 FunctionalBlock

A Functional Block represents an element in a system model that supports a number of functions.

In systems engineering, a function is an element that takes inputs (material, energy, and information) and transforms them into outputs (material, energy, and information).

Note: MechML reuses the stereotype FunctionalBlock defined in the FAS-profile. It doesn't import this profile though, in order not to force people to use it. If both profiles are used, use the Stereotype of the FAS-profile.

As with every other Block, FunctionalBlocks are also defined in a block definition diagram (bdd) and used in an internal block diagram (ibd).

Attributes

- priority : Percent
  Specifies the importance of the function for the system.

Relations

- Generalization
  a FunctionalBlock may be the target of Generalizations from PrincipleSolutions

2.4 Notation

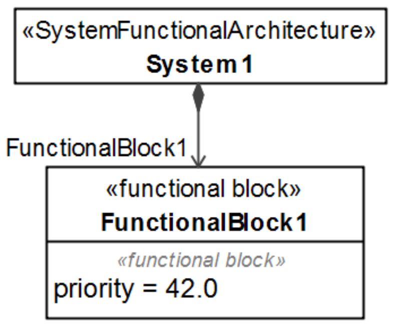

Figure 4 – Graphical nodes used in functional views

3 Principles

3.1 Summary

A principle in systems engineering is a fundamental concept that serves as the basis for a design decision, or for a chain of reasoning when system design decisions must be made. MechML provides two kinds of principles: the SolutionPrinciple and the OperatingPrinciple.

3.2 Abstract Syntax

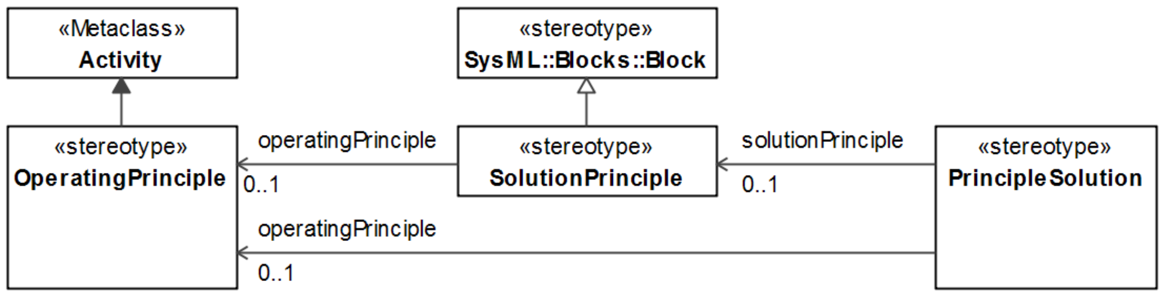

Figure 5 – Principles

3.3 Definitions

3.3.1 OperatingPrinciple

A physical effect together with geometrical and material properties can form an OperatingPrinciple that can be used to solve a construction problem. In the literature there exist collections of such OperatingPrinciples. It is expected that a number of libraries of such principles will eventually be available.

3.3.2 SolutionPrinciple

A SolutionPrinciple concretizes how an OperatingPrinciple will be used to solve the problem at hand. It can also be used without specifying the OperatingPrinciple explicitly. Later it will be the basis of PrincipleSolutions in a morphological box.

Attributes

- operatingPrinciple : OperatingPrinciple
  The OperatingPrinciple on which the SolutionPrinciple is based.

3.4 Notation

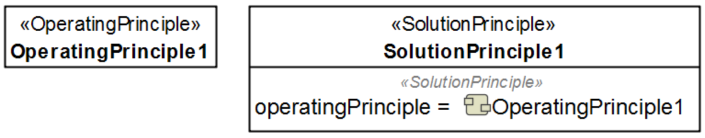

Figure 6 – Graphical nodes used for principles

4 Principle View

4.1 Summary

In mechanical engineering, the morphological analysis is a common and well-known method to cope with multi-dimensional and non-quantifiable problems. The Swiss astronomer Fritz Zwicky developed this approach to seemingly non-reducible complexity. The method, which was originally used for the classification of astrophysical objects, is now very widespread as a technique for fostering creativity, especially in the domain of mechanical engineering.

To support the engineer in morphological analysis, usually a tool called morphological box (also known as “Zwicky Box”) is used. MechML provides elements to create a morphological box in a system model.

4.2 Abstract Syntax

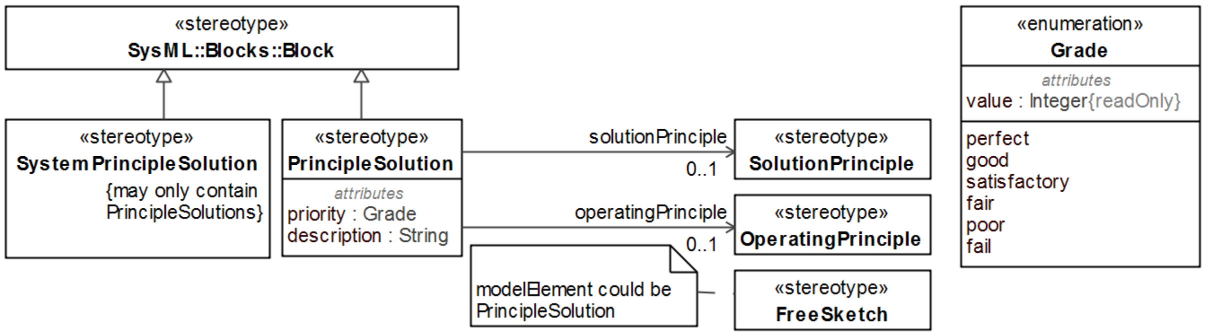

Figure 7 – Principle View

4.3 Definitions

4.3.1 SystemPrincipleSolution

The SystemPrincipleSolution represents the system of interest in the Principle View. It is a specialization of the SystemFunctionalArchitecture and inherits the functional architecture defined there. All the functional parts will later get redefined by PrincipleSolutions in order to select a set of Solutions that make up a potential system.

Relations

- Generalization
  The SystemFunctionalArchitecture for which this is a potential solution.

Constraints

- PrincipleSolutions as Parts
  {may only contain PrincipleSolutions}

4.3.2 PrincipleSolution

A PrincipleSolution describes how the functions of a FunctionalBlock could get supported in principle. It is related via a Generalization to the FunctionalBlock. It can get visualized in a FreeSketch.

Relations

- Generalization
  The FunctionalBlock for which this is a potential solution.

Attributes

- solutionPrinciple : SolutionPrinciple [0..1]
  The underlying SolutionPrinciple of this PrincipleSolution
- operatingPrinciple : OperatingPrinciple [0..1]
  The underlying OperatingPrinciple of this PrincipleSolution. This is used when no SolutionPrinciple was defined.
- priority : Grade
  How good does this solution support the intended functions?
- description : String
  A textual description of the solution.

4.4 Notation

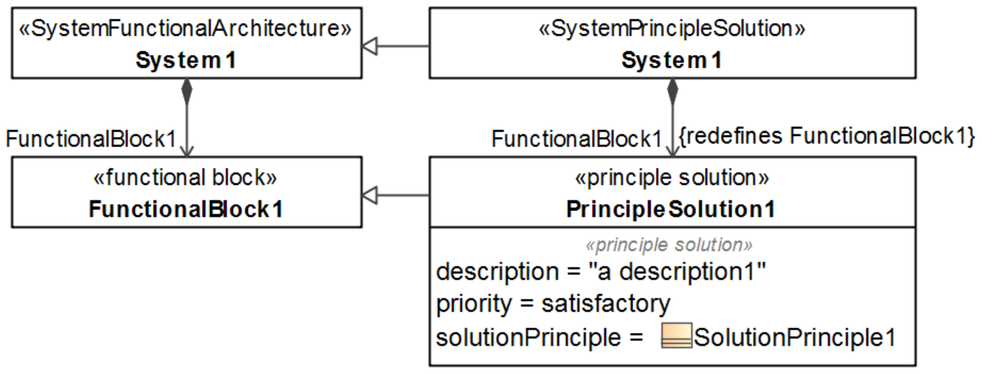

Figure 8 – Graphical nodes used for the principle view

The morphological box can be visualized in a block definition diagram, where the column on the left side shows all the functional blocks, followed to the right by the PrincipleSolutions (see following diagram). The selected combination of PrincipleSolutions (path through the morphological box) can be made visible by drawing Composite Relationships for all of them and bending them at appropriate points, so that all lines are graphically merged to one.

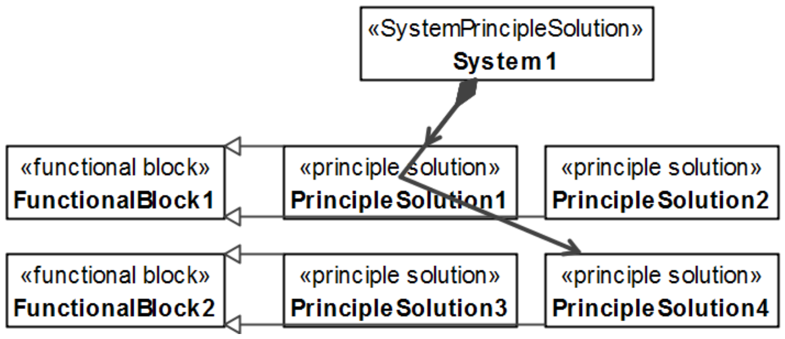

Figure 9 – Morphological box visualized with Composition Relationships

5 Conceptual View

5.1 Summary

A conceptual View is a view that describes an overall concept for the system of interest. Several concepts might get explored before the final decision is taken, which one to build.

Note: There are two possibilities to trace abstract PrincipleSolutions to more concrete ConceptProperties and to even more concrete BuildingStructureProperties: Stereotype Attributes and Allocation Relationships. In this specification both are described. The standard SysML way to connect different levels of abstraction is the Allocation relationship. The disadvantage of this relationship is, that it is allowed between any elements. Therefore the tool cannot support the modeler by allowing only valid connections. With stereotype properties the type is specified, so the tool will only allow elements of the correct type. A model should always use the method consistently. Choose the method that is best supported by the tool of your choice.

5.2 Abstract Syntax

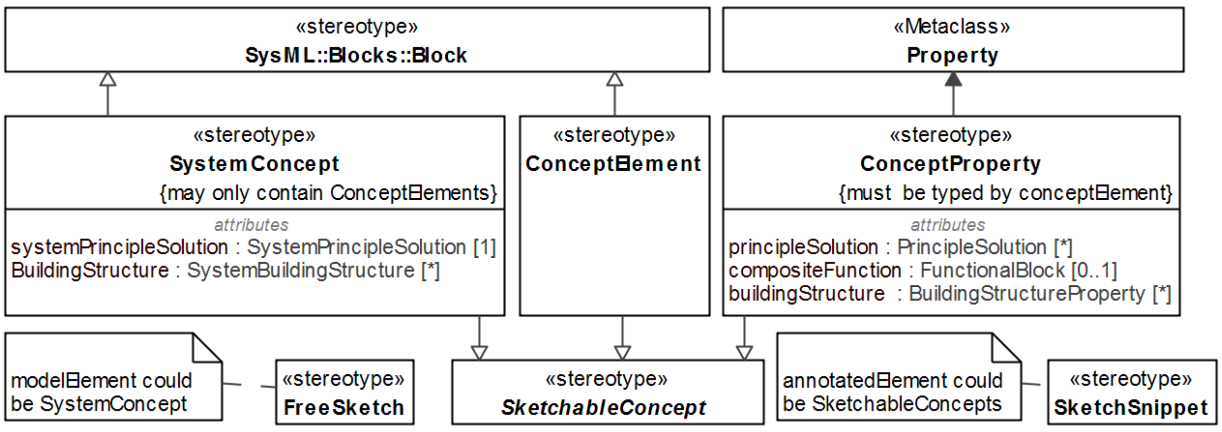
«stereotype»
SysML::Blocks::Block
«Metaclass»
Property
«stereotype»
SystemConcept
{may only contain ConceptElements}
attributes
systemPrincipleSolution : SystemPrincipleSolution [1]
BuildingStructure : SystemBuildingStructure [*]
«stereotype»
ConceptElement
«stereotype»
ConceptProperty
{must be typed by conceptElement}
attributes
principleSolution : PrincipleSolution [*]
compositeFunction : FunctionalBlock [0..1]
buildingStructure : BuildingStructureProperty [*]
modelElement could be SystemConcept
«stereotype»
FreeSketch
«stereotype»
SketchableConcept
annotatedElement could be SketchableConcepts
«stereotype»
SketchSnippet

Figure 10 – Conceptual View

5.3 Definitions

5.3.1 SystemConcept

The SystemConcept represents the system in the conceptual view. It references the SystemPrincipleSolution that is concretized by it and the SystemBuildingStructure that shows the BuildingStructure that concretizes it. It can be depicted in a FreeSketch or be a part of a Sketch defined by a SketchSnippet.

Relations

- Allocation from SystemPrincipleSolution
  The source is the SystemPrincipleSolution, that is concretized by this SystemConcept.
- Allocation to SystemBuildingStructure
  The target is the SystemBuildingStructure, that concretizes this SystemConcept.

Attributes

- systemPrincipleSolution : SystemPrincipleSolution [1]
  The SystemPrincipleSolution that is concretized by the SystemConcept. Alternative to the Allocation (see above).
- BuildingStructure : SystemBuildingStructure [*]
  The SystemBuildingStructures that concretizes this SystemConcept. Alternative to the Allocation (see above).

Constraints

- ConceptElements as Parts
  {may only contain ConceptElements}

5.3.2 ConceptElement

A ConceptElement is a part of a SystemConcept. It could be shown as a SketchSnippet in a FreeSketch. It can define ports that are to be connected in the conceptual architecture. Port Compatibility rules (as defined by SysML or additional own rules) can then be used to verify the architecture.

Note: It is preferable not to attach Allocations here (this would be Allocation of definition). They should instead be attached to the parts that are typed by ConceptElements (Allocation of usage: only the usage of an element is allocated. Different usages could have different Allocations).

5.3.3 ConceptProperty

This is an auxiliary stereotype, used to mark part properties of a SystemConcept that are typed by ConceptElements. It carries some attributes, that allow traceability to the other views. Also it can be the cause for a new FunctionalBlock (compositeFunction), that is needed because of the decision for this ConceptElement. This new FunctionalBlock makes it necessary to find new PrincipleSolutions and a new morphological sub box.

Allocations

- Allocation from PrincipleSolution
  The source is the PrincipleSolution, that is concretized by this ConceptProperty.
  Note: This is mixed allocation from definition to usage. When the conceptual view is already very detailed, it could make sense to use the properties of the SystemPrincipleSolution as sources of the allocation.
- Allocation to BuildingStructureProperty
  The target is the BuildingStructureProperty, that concretizes this ConceptProperty.

Attributes

- compositeFunction : FunctionalBlock [0..1]
  A new FunctionalBlock that becomes necessary because of the introduction of this ConceptElement. It will later be the root for a new sub morphological box.
- principleSolution : PrincipleSolution [*]
  The PrincipleSolutions that are concretized by the ConceptProperty. Alternative to the Allocation (see above).
- buildingStructure : BuildingStructureProperty [*]
  The SystemBuildingStructures that concretizes this SystemConcept. Alternative to the Allocation (see above).

Constraints

- typed by ConceptElement
  {must be typed by ConceptElement}

5.4 Notation

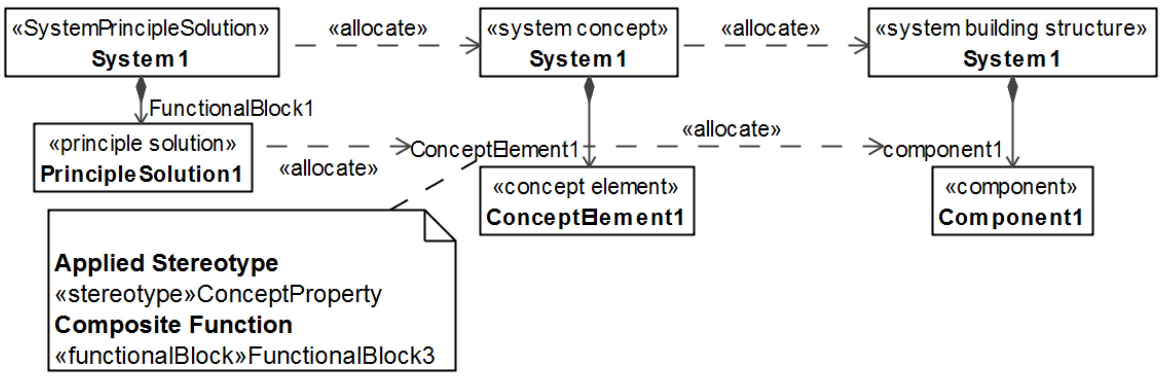

Figure 11 – Graphical nodes used for conceptual views in bdds (Allocation)

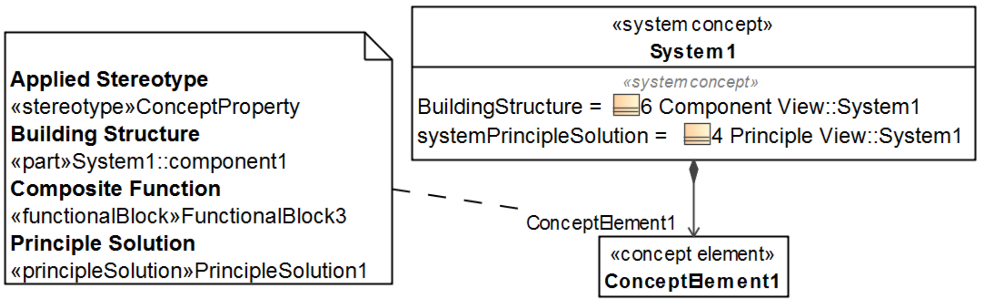
«system concept»
System1
«system concept»
BuildingStructure = 6 Component View::System1
systemPrincipleSolution = 4 Principle View::System1
Applied Stereotype
«stereotype»ConceptProperty
Building Structure
«part»System1::component1
Composite Function
«functionalBlock»FunctionalBlock3
Principle Solution
«principleSolution»PrincipleSolution1
ConceptElement1
«concept element»
ConceptElement1

Figure 12 – The same model as in Figure 11 with properties instead of allocations

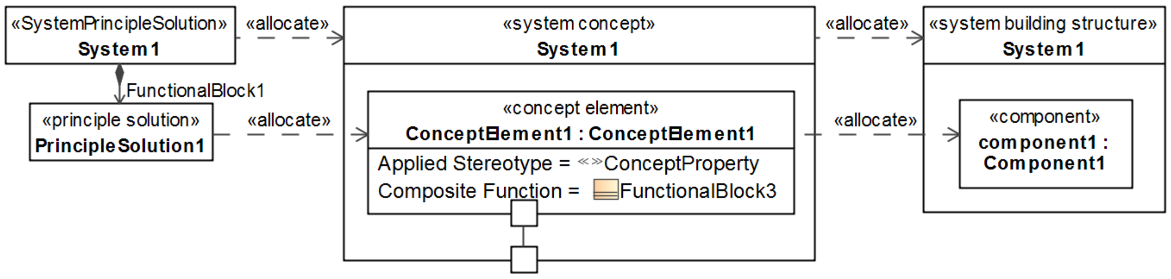

Figure 13 – The same model as in Figure 11 with ibd

An ibd can show the ports of the ConceptElements and how they are connected.

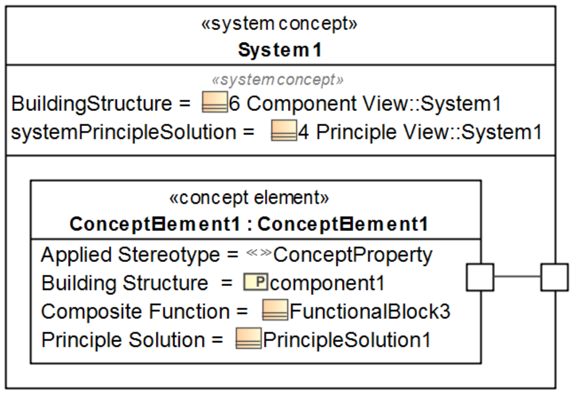
«system concept»
System1
«system concept»
BuildingStructure = 6 Component View::System1
systemPrincipleSolution = 4 Principle View::System1
«concept element»
ConceptElement1 : ConceptElement1
Applied Stereotype = «»ConceptProperty
Building Structure = component1
Composite Function = FunctionalBlock3
Principle Solution = PrincipleSolution1

Figure 14 – The same model as in Figure 12 with ibd

6 Component View

6.1 Summary

The Component View is the most concrete of all four views. It shows a physical building structure of the system of interest. It is the bridge into the world of Computer Aided Design (CAD). From this view an empty CAD-structure can get generated that will be used as a basis for the 3D-construction (a prototype for this generator has been part of the FAS4M project). The MechML model will then hold CAD-IDs to reference the corresponding elements of the CAD-structure. Vice versa the CAD-Model will hold references to the MechML-model. This way a traceability from a concrete CAD-part to the MechML-component back to the concept, principle, function and requirement is established. The common gap between the interdisciplinary systems engineering level and the mechanical design is thus bridged.

6.2 Abstract Syntax

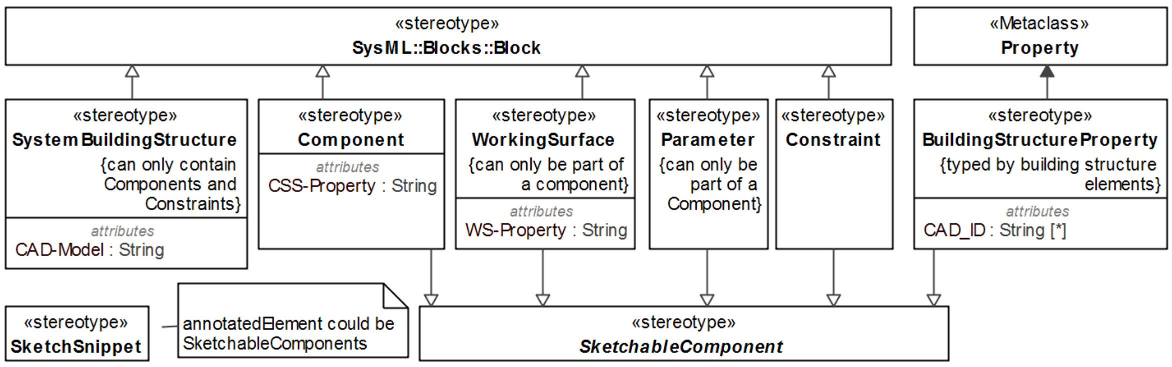

Figure 15 – Component View

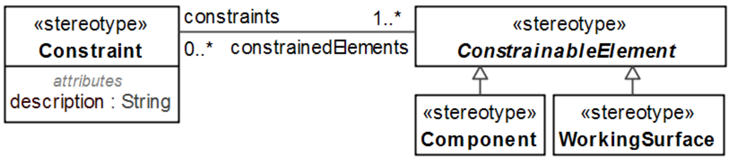

Figure 16 – Constraint

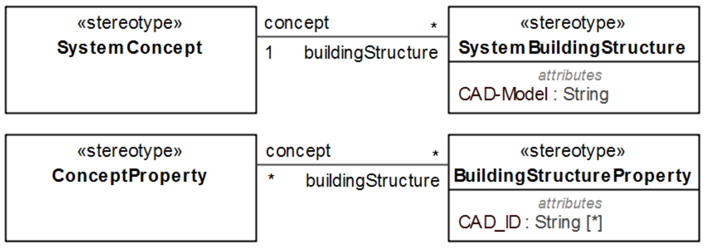

Figure 17 – Traceability

6.3 Definitions

6.3.1 SystemBuildingStructure

The SystemBuildingStructure represents the system in the component view. It references the SystemConcept that is concretized by it and the CAD-model that shows the concrete 3D-design of the system.

Relations

- Allocation from SystemConcept
  The source is the SystemConcept, that is concretized by this SystemBuildingStructure.

Attributes

- concept : SystemConcept [1]
  The SystemConcept that is concretized by this SystemBuildingStructure. Alternative to the Allocation (see above).
- CAD-Model : String
  A reference (URL, filename,…) to the CAD-model that shows the detailed 3-D construction for this system.

Constraints

- Components and Constraints as Parts
  {can only contain Components and Constraints}

6.3.2 Component

A Component is a part of a SystemBuildingStructure of another Component. It could be shown as a SketchSnippet in a FreeSketch. It can define ports that are to be connected in the physical architecture. Port Compatibility rules (as defined by SysML or additional own rules) can then be used to verify the architecture.

It can be depicted as part of a FreeSketch defined by a SketchSnippet.

Note: It is preferable not to attach Allocations here (this would be Allocation of definition). They should instead be attached to the parts that are typed by Components (Allocation of usage: only the usage of an element is allocated. Different usages could have different Allocations).

Attributes

- CSS-Property : String
  Channel and support structure. Specifies the properties of the component that follow from its use as such a structure.

6.3.3 WorkingSurface

Two Components interact, when they are connected via a working surface pair. One half of this pair is modeled with the WorkingSurface.

Attributes

- WS-Property : String
  Working surface. Specifies the properties of the WorkingSuface that follow from its use as such a surface.

Constraints

- part of Component
  {can only be part of a component}

6.3.4 Parameter

A Parameter is a value relevant for the 3D-construction, that is already known while modeling the SystemBuildingStructure. Examples are angles, dimensions or material. It can refer to a SketchSnippet of a FreeSketch, where the Parameter is used.

Constraints

- part of Component
  {can only be part of a component}

6.3.5 Constraint

A Constraint describes aspects of the design like geometrical position, measurement, form or position tolerance, surface quality, global orientation or fitting. It constrains one or more elements, which can be Components or WorkingSurfaces. It can refer to a SketchSnippet of a FreeSketch, where the constrained elements are shown.

Attributes

- description : String
  textual description of the Constraint.

6.3.6 ConstrainableElement {abstract}

Abstract base element for elements that can get constrained by a Constraint. There are two specializations defined: Component and WorkingSurface.

6.3.7 BuildingStructureProperty

All parts of a SystemBuildingStructure or of a Component are BuildingStructureProperties. They can get depicted as part of a FreeSketch referenced by a SketchSnippet.

Relations

- Allocation from ConceptProperty
  The source is the ConceptProperty, that is concretized by this BuildingStructureProperty.

Attributes

- concept : ConceptProperty [0..*]
  The ConceptProperty that is concretized by this BuildingStructureProperty. Alternative to the Allocation (see above).
- CAD_ID : String[0..*]
  An ID known in the CAD-model. Together with the CAD-Model property of the SystemBuildingStructure this ID references a unique element in the CAD-model. There can be many CAD_IDs, since a BuildingStructureProperty can have a multiplicity greater than 1. Then the CAD-model will contain multiple CAD-elements with unique IDs that correspond to one BuildingStructureProperty.

Constraints

- typed by building structure elements
  { must be typed by Component, WorkingSurface, Constraint or Parameter}

6.4 Notation

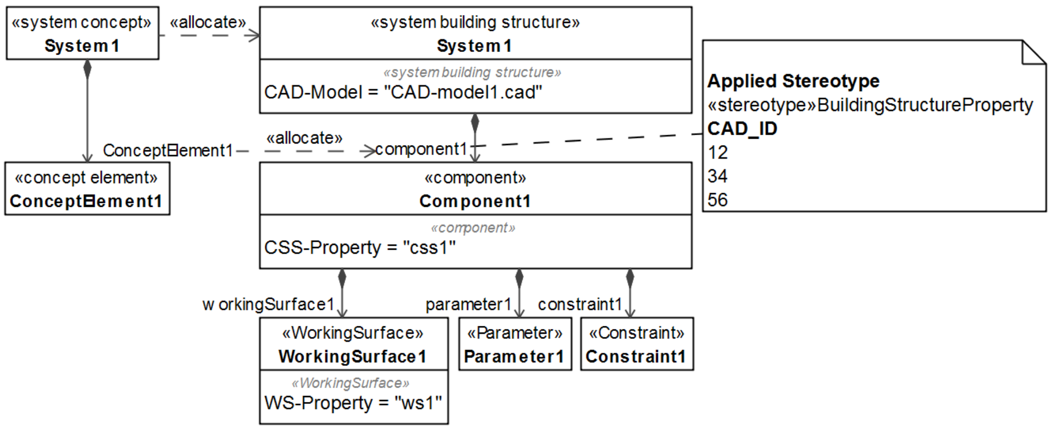

Figure 18 – Graphical nodes used for component views in bdds (Allocation)

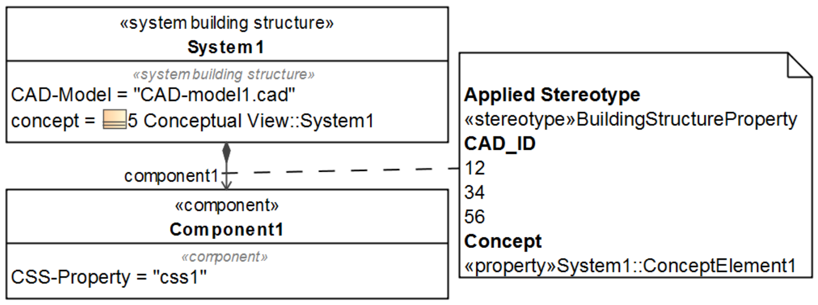
«system building structure»
System1
«system building structure»
CAD-Model = "CAD-model1.cad"
concept = 5 Conceptual View::System1
component1
«component»
Component1
«component»
CSS-Property = "css1"
Applied Stereotype
«stereotype»BuildingStructureProperty
CAD_ID
12
34
56
Concept
«property»System1::ConceptElement1

Figure 19 – Part of the same model as in Figure 18 with properties instead of allocations

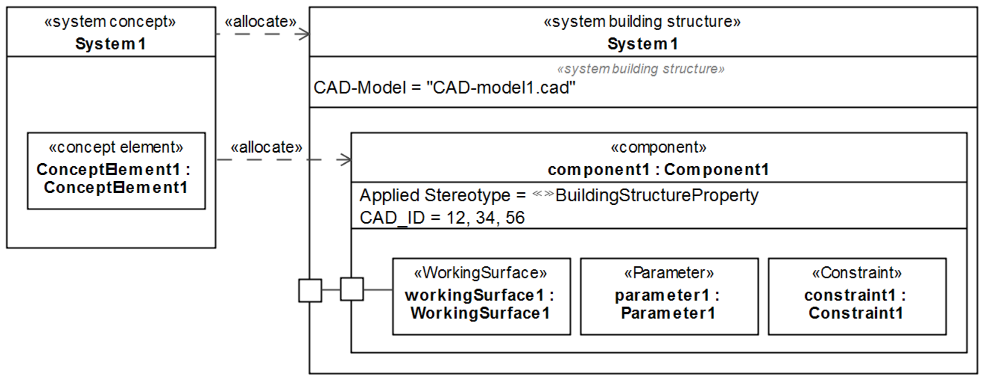
«system concept»
System1
«concept element»
ConceptElement1 : ConceptElement1
«allocate»
«allocate»
«system building structure»
System1
«system building structure»
CAD-Model = "CAD-model1.cad"
«component»
component1 : Component1
Applied Stereotype = «»BuildingStructureProperty
CAD_ID = 12, 34, 56
«WorkingSurface»
workingSurface1 : WorkingSurface1
«Parameter»
parameter1 : Parameter1
«Constraint»
constraint1 : Constraint1

Figure 20 – The same model as in Figure 18 with ibd

An ibd can show the ports of the Components and how they are connected.

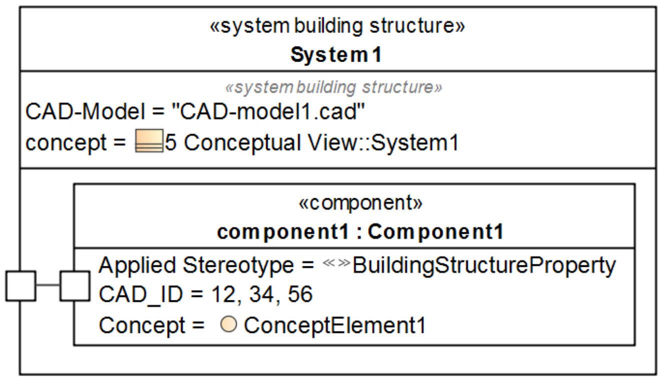
«system building structure»
System1
«system building structure»
CAD-Model = "CAD-model1.cad"
concept = 5 Conceptual View::System1
«component»
component1 : Component1
Applied Stereotype = «»BuildingStructureProperty
CAD_ID = 12, 34, 56
Concept = ConceptElement1

Figure 21 – Part of the same model as in Figure 19 with ibd

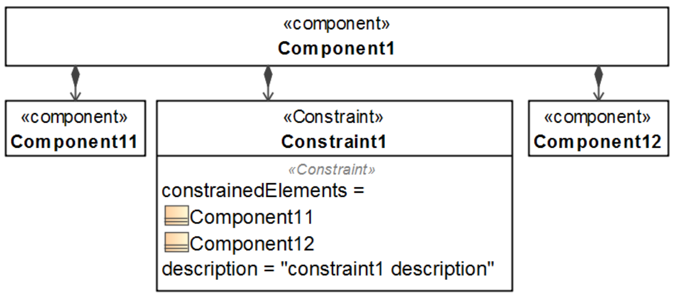

Figure 22 – Constraint

7 Sketches

7.1 Summary

Developing and discussing ideas and concepts using free, usually hand-drawn sketches, e.g. on a flip chart, a white board, or a napkin, is a widespread approach in many engineering disciplines, especially in mechanical engineering. Furthermore, due to its high level of abstraction, SysML and its model elements are unsuitable to convey precise information about the shape and the detailed design (e.g. forms, distances, etc.) of a mechanical component. For example, the spatial arrangement of SysML elements on a diagram's canvas is meaningless and says nothing about the spatial relationships of the described system or component.

For this reason, MechML introduces special elements to incorporate drawings and sketches into the model.

7.2 Abstract Syntax

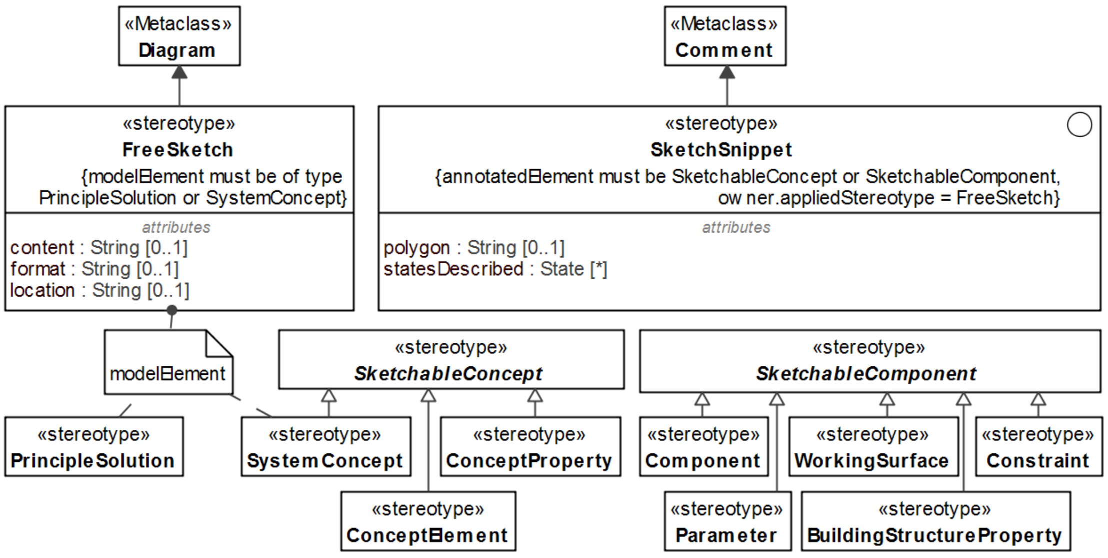
«Metaclass» Diagram
«Metaclass» Comment
«stereotype» FreeSketch {modelElement must be of type PrincipleSolution or SystemConcept}
attributes
content : String [0..1]
format : String [0..1]
location : String [0..1]
«stereotype» SketchSnippet {annotatedElement must be SketchableConcept or SketchableComponent, owner.appliedStereotype = FreeSketch}
attributes
polygon : String [0..1]
statesDescribed : State [*]
modelElement
«stereotype» SketchableConcept
«stereotype» SketchableComponent
«stereotype» PrincipleSolution
«stereotype» SystemConcept
«stereotype» ConceptProperty
«stereotype» Component
«stereotype» WorkingSurface
«stereotype» Constraint
«stereotype» ConceptElement
«stereotype» Parameter
«stereotype» BuildingStructureProperty

Figure 23 – Sketches

7.3 Definitions

7.3.1 FreeSketch

A FreeSketch can be used to incorporate a non-formal drawing (sketch) in the system model. An element of type FreeSketch can hold or reference a drawing that typically has been digitized using a digital camera or scanner.

FreeSketch is a stereotype defined on the UML metaclass Diagram.

Attributes

- content : String [0..1]
  A representation of the image.
- format : String [0..1]
  The format of the image (jpg, png or the like).
- location : String [0..1]
  External location of the image file, if the image is not embedded in the model.
- ^modelElement : Element
  The model element this Diagram refers to (attribute inherited from Metaclass Diagram). FreeSketches could refer to PrincipleSolutions or SystemConcept.

Constraints

- PrincipleSolution or SystemConcept
  { modelElement must be of type PrincipleSolution or SystemConcept}

7.3.2 SketchableConcept {abstract}

Abstract base element for elements that can get annotated by a SketchSnippet. Following Specializations are defined: SystemConcept, ConceptElement, and ConceptProperty.

7.3.3 SketchableComponent {abstract}

Abstract base element for elements that can get annotated by a SketchSnippet. Following Specializations are defined: Component, WorkingSurface, Parameter, Constraint, and BuildingStructureProperty.

7.3.4 SketchSnippet

A SketchSnippet is a model element to mark snippets, i.e. areas or sections in a FreeSketch. Thus, certain areas of a drawing can be referenced, and relationships to other model elements can be created. It is based on the Metaclass Comment, which allows it to annotate Elements and carry a text.

Attributes

- Polygon : String [0..1]
  The Polygon enclosing a part of the Sketch.
- statesDescribed : State[0..*]
  The states that are shown in the Snippet. For this purpose the SystemConcept needs to have a StateMachine. The States in this StateMachine can get referenced here.
- ^body : String [0..1]
  The property inherited from Comment, that can be used to describe the SketchSnippet.
- ^annotatedElement : Element [0,..*]
  The property inherited from Comment, that lists the Elements that are shown in this SketchSnippet

Constraints

- owned by Sketch
  {owner.appliedStereotype = FreeSketch}
- annotates elements of concepts and building structures
  {annotatedElement must be SketchableConcept or SketchableComponent}

7.4 Notation

The concrete syntax of the FreeSketch is a representation of the sketch itself, i.e. the drawing will be shown.

The concrete syntax of a SketchSnippet is a polygon, i.e. a finite chain of straight line segments closing in a loop to form a closed chain, that encloses the area of the FreeSketch that should be referenced.

Note: There is no UML defined possibility to show images or polygons on a diagram. Here we are leaving standard UML notation. Most tools however allow to include an image on any diagram. It is expected that a tool fully supporting MechML will include some kind of graphical editing capabilities like adjusting the image, creating polygons, detecting shapes on the Sketch, locking Snippets to protect them against accidental moving, highlighting Snippets that are selected, choosing the color of snippets and showing references to other model elements when the mouse is hovering above it.

If you don’t have the tool support, you can still use the standard notation of Comments. If it is possible to make them translucent, they could be used to mark rectangular regions on the diagram. Clearly this is only a workaround, but it allows you to use the concepts described here with any SysML-tool.

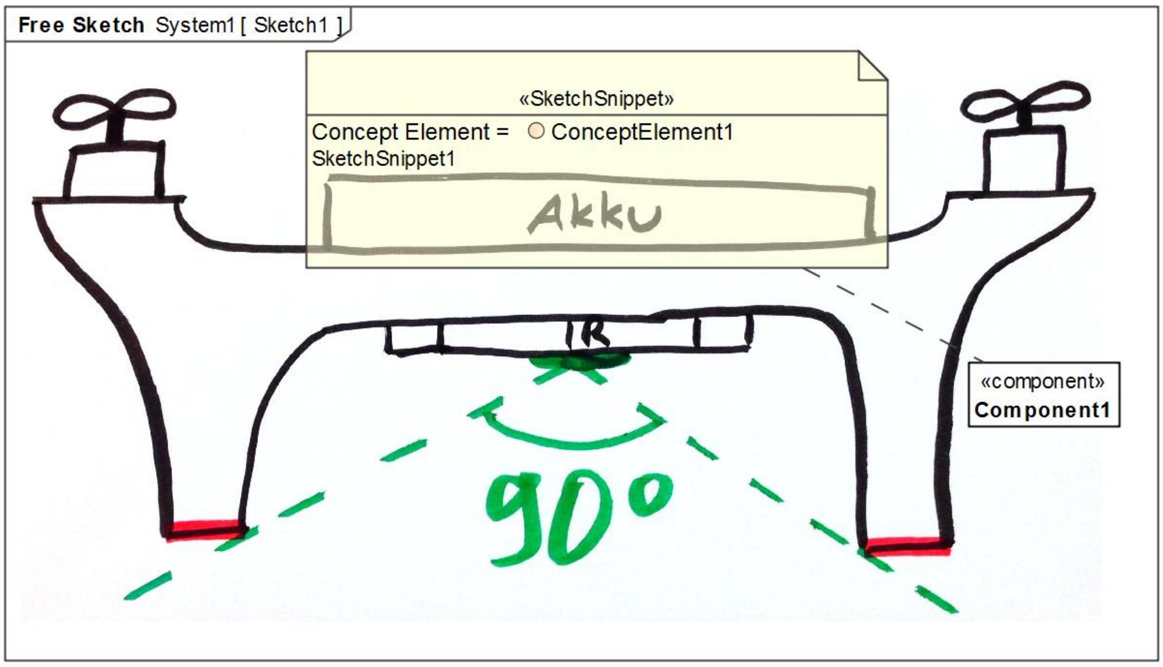
Free Sketch System1 [Sketch1]
«SketchSnippet»
Concept Element = ConceptElement1
SketchSnippet1
Akku
IR
90°
«component»
Component1

Figure 24 – Graphical nodes used for Sketches

The referenced Elements can be listed in the SketchSnippet symbol (here ConceptElement1). If they can be shown on a block definition diagram (which is the basis for this diagram), they could also be displayed directly with a dashed line connecting them to the SketchSnippet (here Component1). The optional textual description appears below the list of referenced elements.