Tim Weilkiens

SYSMOD – The Systems Modeling Toolbox

Pragmatic MBSE with SysML

3rd edition

MBSE4U Booklet Series

OMG SYSTEMS MODELING LANGUAGE™

SysML®

covers version 1.6

www.model-based-systems-engineering.com

SYSMOD - The Systems Modeling Toolbox, 3rd edition

Pragmatic MBSE with SysML

Tim Weilkiens

This book is for sale at http://leanpub.com/sysmod

This version was published on 2020-08-01

ISBN 978-3-9818529-9-8

/MBSE4U

Publishing on the Pulse of the Markets

MBSE4U is a publishing house for books about MBSE. They are intended to be regularly updated to align the content with the highly dynamic systems engineering domain.

Contents

About MBSE4U . . . . . **i**

About Tim Weilkiens . . . . . **ii**

History and Outlook . . . . . **iii**
SYSMOD Versions iii
SYSMOD Outlook iv

Preface . . . . . **v**

1. SYSMOD - The Systems Modeling Toolbox **1**

2. Engineering for the Planet **5**

3. SYSMOD Profile and Model Libraries **7**
3.1 Activities 9
3.2 Actors 10
3.3 Blocks 13
3.4 Discipline-specific Elements 14
3.5 Relationships 15
3.6 Requirements and Risks 20
3.7 Test Cases 23
3.8 Use Cases 25
3.9 Variants (VAMOS stereotypes) 26
3.10 SYSMOD Enumerations 27

3.11 SYSMOD PartsCatalogue Library 28
3.12 SYSMOD Simulation Library 31
3.13 SYSMOD Engineering4Planet Library . . . 32

Appendix A: Mapping ISO 15288 to SYSMOD 34
A.1 ISO 15288: Technical Processes 34
A.2 ISO 15288: Technical Management Processes 38
A.3 ISO 15288: Agreement Processes 39
A.4 ISO15288: Organizational Project-Enabling . 39

Bibliography **41**

Index **45**

About MBSE4U

Publishing on the Pulse of the Market

MBSE4U is a publishing organization for model-based systems engineering books that are regularly updated to follow the dynamic changes in the MBSE community and the markets.

MBSE4U has published the following books:

- Tim Weilkiens. SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML. 3rd edition. 2020.
- Tim Weilkiens. The New Engineering Game. 2019.
- Benjamin Weinert. Ein Framework zur Architekturbeschreibung von sozio-technischen maritimen Systemen. 2018. (German Edition)
- Christian Neureiter. A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid. 2017.
- Tim Weilkiens. Variant Modeling with SysML. 2016.

Please let us know if you want to write a book to be published by MBSE4U (info@mbse4u.com). We have a lean publishing process to reward the authors well for providing their valuable MBSE knowledge to the community.

About Tim Weilkiens

I am a consultant and trainer, author, publisher, lecturer, executive board member of the German consulting and training company oose, and active member of the OMG and INCOSE organizations. I wrote parts of the initial SysML specification, and I am still active in the ongoing work on SysML. I am involved in many MBSE activities, and you can meet me at many conferences on MBSE and related topics.

As a consultant, I have advised many companies in different domains. The insights into their challenges are one source of my experience that I share in my books and presentations.

I have written many books about modeling, including *Systems Engineering with SysML* (Morgan Kaufmann, 2008) and *Model-Based System Architecture* (Wiley, 2015). I am the editor of the pragmatic and independent MBSE methodology SYSMOD – the Systems Modeling Toolbox.

You can contact me at tim@mbse4u.com and read my blog about MBSE at www.mbse4u.com.

History and Outlook

This chapter gives a brief overview of the SYSMOD version's history and an outlook on future planned changes.

SYSMOD Versions

- 4.2 Third edition of the book *SYSMOD - The Systems Modeling Toolbox. MBSE4U. 2019*
  - Added chapter "Engineering for the Planet"
  - Added SYSMOD Methods: "Analyze the Problem", "Model Risks", "Specify Test Cases", and "Model the Test Architecture"
  - Added SYSMOD Adoption Process
  - Added a chapter about the adoption of MBSE
  - Added a chapter about a brief introduction to SysML 1.6
  - Mapping ISO 15288 to SYSMOD
  - New OCL constraints for SYSMOD stereotypes (*«continuousUseCase»*)
  - New StakeholderKindCategory: Other
  - Updated to SysML 1.6
  - Some minor typos and updates
- 4.1 Second edition of the book *SYSMOD - The Systems Modeling Toolbox. MBSE4U. 2016*
- 4.0.2 Actor stereotypes specialize SysML Block
- 4.0.1 Fixed some typos and minor changes

- 4.0 First edition of the book *SYSMOD - The Systems Modeling Toolbox. MBSE4U. 2015*
- 3.0 Third edition of the German book *Systems Engineering mit SysML/UML. dpunkt-Verlag. 2014*
- 2.0 First edition of the English book *Systems Engineering with SysML/UML. Morgan Kaufman. 2008*
- 1.0 First publication of SYSMOD in the German book *Systems Engineering mit SysML/UML. dpunkt-Verlag. 2006*

SYSMOD Outlook

- More behavior descriptions in architecture models
- More tools for on-site workshops to elaborate SYSMOD Products
- Functional safety modeling
- Methods for Collaborative Engineering
- MBSE and Digital Twins, Data Analytics, Artificial Intelligence, and Machine Learning
- Configuration Management
- Framework Evaluation of MBSE Methodologies for Practitioners (FEMMP)
- Description of SYSMOD using the Essence framework

Preface

Many years ago, I bundled modeling methods and practices to the Systems Modeling Toolbox (SYSMOD). At the same time, I worked together with other MBSE experts on the first version of SysML 1.0 [SysML07]. SYSMOD is a discovery and not an invention. It consists of already well-known methods and practices. I am more an editor than an author of SYSMOD and collected practices, transferred some of them from other disciplines to the systems engineering discipline, and described the links between the practices to combine them to a methodology. Nowadays, SYSMOD is used in many industrial projects worldwide.

In 2006, I published SYSMOD in the German book *Systems Engineering mit SysML/UML* (dpunkt) and 2008 in the English edition *Systems Engineering with SysML/UML* [We08]. The third edition of the German book was published in 2014 [We14]. Besides SYSMOD, the books provide a comprehensive description of the SysML. To release more regular updates, in 2015, I published the first edition of a book specifically on the SYSMOD methodology. You are currently holding the book in your hands (or have it stored on your device or cloud).

This third edition of the book includes additional methods about problem analysis, risk management, and testing, a chapter about SysML, and a chapter about the adoption of MBSE in an organization.

The SYSMOD Method Analyze the Problem (section 4.5) is about an explicit rethinking of the problem statement: "Are we solving the right problem with the system?".

The methods Specify Test Cases (section 4.15) and Model the Test Architecture (section 4.21) add the important verification and validation aspect to the SYSMOD methodology. You find a more detailed list of the changes in the history section above.

I appreciate any feedback on the book. You can reach me by email: tim@mbse4u.com. This book is considered to be an eBook. However, a print version of the book is also available.

I like to write books in a gender-fair language. On the other hand, I avoid cluttering the flow of reading by always using both genders in the same sentence. Therefore, I only used one gender where it was not appropriate to use gender-neutral language. Feel free to replace the gender with your favorite one wherever it is appropriate.

I thank my colleagues at my company for long profound discussions about MBSE.

I thank NoMagic for their support. I created the SysML diagrams in this book with their modeling tool Cameo Systems Modeler.

If you need MBSE training or consulting services, feel free to contact me. My company - the consultancy oose - provides professional MBSE training and coachings, for example, to introduce MBSE in your organization.

Tim Weilkiens, July 2020.

tim@mbse4u.com

1. SYSMOD - The Systems Modeling Toolbox

SYSMOD is the abbreviation for Systems Modeling Toolbox and an MBSE methodology with a strong focus on the individual methods. The modeling language and tools come second. Processes in SYSMOD are guidelines and no strict rules. It is more important to master the craftsmanship than to follow a process.

SYSMOD works perfectly together with the OMG Systems Modeling Language (OMG SysML) [SysML19]. SysML is a general-purpose modeling language for systems engineering and a worldwide standard. It is not mandatory to do SYSMOD with SysML. As a default, however, I would always recommend this combination.

Process, *Method*, and *Methodology* are common terms with many different meanings. SYSMOD follows the definitions given by James N. Martin [Ma96]:

"A Process is a logical sequence of tasks performed to achieve a particular objective. A process defines "WHAT" is to be done, without specifying "HOW" each task is performed."

"A Method consists of techniques for performing a task, in other words, it defines the "HOW" of each task."

Based on Jeff Estefan [Es08], a *Tool* facilitates the accomplishment of the methods, and a *Methodology is a consistent set of related processes, methods, and tools.*

SYSMOD uses an extended version of this definition to explicitly include humans: A methodology is a collection of processes, Methods, Products, Roles, and tools.

The tools are a set of applications, servers, interfaces, and so on, which form the System Modeling Environment (SME).

The SYSMOD toolbox consists of three main artifact kinds:

- The SYSMOD Products are significant artifacts of the systems development like requirements or the architecture descriptions.
- The SYSMOD Methods are best practices on how to create an SYSMOD Product.
- The SYSMOD Roles are work descriptions of a person. A Role is responsible for SYSMOD Products and a primary or additional performer of SYSMOD Methods.

The following figure 1.1 depicts the relationships between SYSMOD Methods, Products, and Roles.

A Role is part of 0..* methodologies, is responsible for 1..* Methods, and supports 0..* Methods as a co-worker.

A Method is part of 0..* methodologies and has precisely one Role that is responsible for the Method and some Roles as additional performers. Each Method requires 0..* Products as inputs and produces 1..* Products as outputs. Exactly one Role is responsible for a Product. A Product is part of 0..* methodologies.

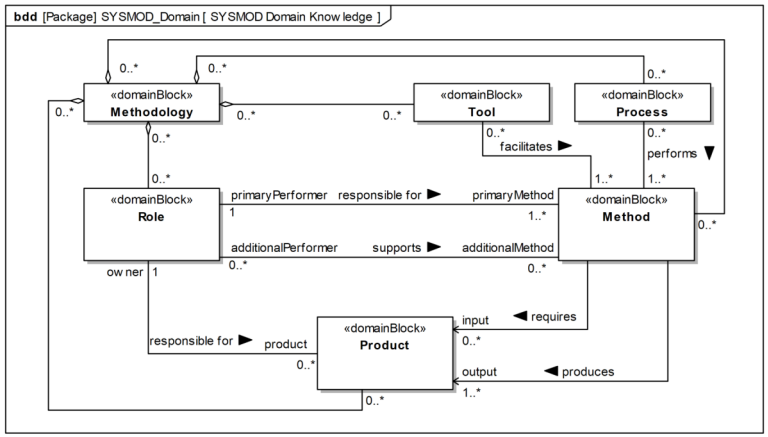

Figure 1.1: Main SYSMOD Artifact Kinds

A process is part of 0..* methodologies and performs 1..* methods. A tool is also part of 0..* methodologies and facilitates 1..* Methods.

A methodology is a collection of processes, Methods, Products, Roles, and tools.

The SYSMOD Infrastructure Process (figure 3.3), SYSMOD Analysis Process (figure 3.4), and the SYSMOD Architecture Process (figure 3.5) demonstrate a typical logical order of execution of the SYSMOD Methods. In practice, a project typically uses a customized set of methods in a different order, and with many iterations and loops.

Before the book starts with the main SYSMOD content, the next chapter 2 solicits your attention to a critical topic. Our planet is in bad shape, and I assume all of you would like to have a good life for yourself, your family, your descendants, and all the life around you. It is quite evident that we must actively do something to make that happen. We engineers must also take responsibility. Chapter 2 deals with

the responsibility of each individual engineer and channeling engineering power for the planet.

Since the SYSMOD Processes provide a good overview of the toolbox, I start to describe SYSMOD with the process chapter. Next, follows the chapters about the SYSMOD Methods, Products, and Roles. Each chapter lists the elements with a brief description.

The Guidances chapter provides practical explanations of how to do the modeling of the SYSMOD Methods and Products.

Examples are always helpful for a better understanding. The Examples chapter demonstrates the application of SYSMOD with a fictitious example. Since a book is a book and a model is a model, the examples chapter can only provide some views on the overall model. The example as a model is available as part of the SYSMOD plugin (see www.mbse4u.com).

Chapter 9 presents the SysML profile for SYSMOD. It describes the stereotypes that are necessary to apply the SYSMOD concepts to a model.

Chapter 10 describes how to define a customized MBSE Methodology using SYSMOD.

Chapter 11 is about more MBSE Tools and opens additional drawers of the MBSE toolbox and presents some helpful patterns and other practices for MBSE.

Chapter 12 provides a brief introduction to the OMG Systems Modeling Language (SysML) Version 1.6.

In the annex, you find a mapping of SYSMOD to the ISO15288 processes.

2. Engineering for the Planet

Engineers build great systems that save lives, protect the environment, provide energy, safety, mobility, economic development, convenience, and many more. But they also build systems that kill and harm people, destroy our environment, and do other bad things.

The people who use such systems take the primary responsibility, but the engineers who built the systems also take part in it. They have the expertise and can influence solutions to ensure a more sustainable future.

We are facing more and more global challenges like climate change, increasing world population, pandemics, environmental destruction, or the possibility of an asteroid impact.

Engineering power makes the impossible possible. Engineers brought humans to the moon, which is still unbelievable, but it was much more unthinkable in the 1960ies. The Chinese built a dam that slows down the earth's rotation. Regardless of if all this is good or bad, it is extremely powerful.

Regarding the critical situation of our planet, we must channelize the power of engineering for a worth living future. Besides large planet-saving projects and heroic deeds, it is also about the many small micro-decisions and actions of each engineer in the world that, in total, will move us in the right direction.

Let *Engineering for the Planet* be part of the engineering mindset, a mandatory part of standards and policies, and part of the engineer's toolbox full of methods & tools.

SYSMOD, for example, provides an actor element *Planet Environment* associated with tasks about sustainability analysis, and an actor category *«environmentalImpact»*. See section 7.8 about how to use the Planet Environment.

X4Planet is an organization to focus on global resources experts on saving the planet. The X stands for any domain like engineering or organizational consulting. Join the community and do your part. Following X4Planet on social media is a good starting point. See www.x4planet.org for more information.

3. SYSMOD Profile and Model Libraries

The SYSMOD Profile is a set of stereotypes adding some SYSMOD-specific elements to the SysML vocabulary. SysML is too general to be used out-of-the-box without any extensions. For example, SysML does not provide elements for system hierarchies like system or subsystem or different requirement kinds and additional requirement properties. See section 12.11 for more details about profiles.

The following sections describe each SYSMOD stereotype and its formal definition to easily define yourself an SYSMOD Profile in your modeling tool. You can also download the SYSMOD Profile for some modeling tools from the website www.mbse4u.com.

Alphabetical list of SYSMOD stereotypes:

- 9.2 *«actuator»*
- 9.2 *«boundarySystem»*
- 9.6 *«businessRequirement»*
- 9.5 *«conjugated»*
- 9.6 *«constraintRequirement»*
- 9.1 *«continuousActivity»*
- 9.8 *«continuousUseCase»*
- 9.3 *«documentBlock»*
- 9.3 *«domainBlock»*
- 9.4 *«electrical»*

- 9.2 *«environmentalEffect»*
- 9.2 *«environmentalImpact»*
- 9.5 *«exDeriveReqt»*
- 9.6 *«extendedRequirement»*
- 9.6 *«extendedStakeholder»*
- 9.7 *«extendedTestCase»*
- 9.2 *«externalSystem»*
- 9.6 *«functionalRequirement»*
- 9.6 *«legalRequirement»*
- 9.4 *«mechanical»*
- 9.2 *«mechanicalSystem»*
- 9.7 *«modelTestCase»*
- 9.6 *«non-functionalRequirement»*
- 9.6 *«objective»*
- 9.3 *«parametricContext»*
- 9.6 *«performanceRequirement»*
- 9.6 *«performanceConstraintRequirement»*
- 9.6 *«physicalRequirement»*
- 9.6 *«reliabilityRequirement»*
- 9.9 *«requires»*
- 9.6 *«risk»*
- 9.2 *«sensor»*
- 9.4 *«software»*
- 9.3 *«subsystem»*
- 9.6 *«supportabilityRequirement»*
- 9.3 *«system»*
- 9.3 *«systemContext»*
- 9.8 *«systemProcess»*
- 9.7 *«systemTestCase»*
- 9.8 *«systemUseCase»*
- 9.6 *«usabilityRequirement»*
- 9.2 *«user»*

- 9.3 *«userInterface»*
- 9.2 *«userSystem»*
- 9.9 *«variant»*
- 9.9 *«variantConfiguration»*
- 9.9 *«variation»*
- 9.5 *«weightedAllocate»*
- 9.5 *«weightedSatisfy»*
- 9.5 *«weightedVerify»*
- 9.9 *«xor»*

Alphabetical list of SYSMOD Enumerations elements:

- 9.10 EffortKind
- 9.10 ObligationKind
- 9.10 Planet Environment
- 9.10 PriorityKind
- 9.10 StabilityKind
- 9.10 StakeholderCategoryKind

SYSMOD provides some additional model libraries. They offer some useful elements and serve as an example of what a model library can look like.

Section 9.11 describes the PartsCatalogue, section 9.12 describes a library for simulation elements, and section 9.13 describes the Engineering4Planet library for modeling of sustainability aspects.

3.1 Activities

The continuous activity is a marker for a Use Case Activity of a continuous System Use Case. A constraint of the continuous

use case stereotype assures that its activity is a *«continuous-Activity»* (see section 9.8).

Figure 9.1 depicts the definition of the SYSMOD stereotype *«continuousActivity»* for Activities.

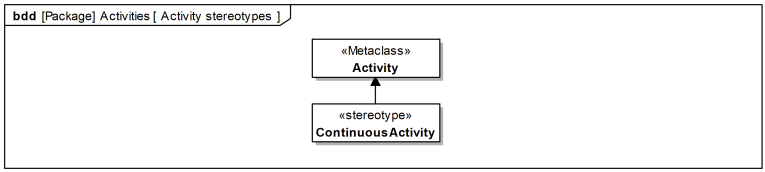

Figure 9.1: SYSMOD stereotype for SysML *Activities*

3.2 Actors

SysML has only a single general model element to model the concept of an actor, i.e., an external entity that interacts with the system of interest. Specific actor kinds like human or external systems are not part of the standard. That is a task for profiles. The SYSMOD Profile provides a set of some specific actor kinds. Figure 9.2 depicts the definition of the stereotypes: *«user»*, *«externalSystem»*, *«environmental-Effect»*, *«mechanicalSystem»*, *«sensor»*, *«actuator»*, *«bound-arySystem»*, *«userSystem»*.

All stereotypes are specializations of the SysML *Block*. See section 11.1: Death of the Actor for more details. If you still want to use the SysML *Actor* model element, you must change the definition of the SYSMOD actor stereotypes. Remove the generalization relationships to the SysML *Block* and change the extended metaclass from *Class* to *Actor*.

For backward compatibility with older SYSMOD versions, the SYSMOD User stereotype can also still be applied to the

SysML Actor element.

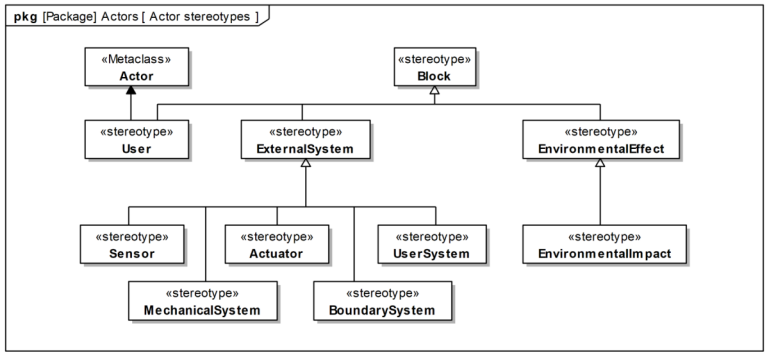

Figure 9.2: SYSMOD stereotypes for SysML *Actors*

There are three top-level actor kinds:

User
: Represents a human actor. The stereotype can be used in combination with the stereotype 9.6 *«extendedStakeholder»*.

Environmental effect
: Represents a relevant effect from the environment on the system, e.g., temperature or humidity.

External system
: Represents a non-human actor.

Further actor kinds are specializations of the external system actor:

Mechanical system
: External system that has only mechanical interfaces to the system of interest, for example, the floor.

Sensor
: External system that provides data from the environment to the system of interest.

Actuator
: External system that has an effect on the environment and is controlled by the system of interest.

User system
: External system that is an interface between a human and a system of interest.

Boundary system
: External system that is an interface between another external system and the system of interest.

Environmental impact
: Impact of the system on the environment in the sense of sustainability (see chapter 2).

The actor kinds *Sensor*, *Actuator*, *User system*, and *Boundary system* are, in particular, useful for embedded systems. In more holistic system development, these entities are typically part of the system of interest and are not actors.

The stereotypes also define their own notation by icons that depict the kind of system actor. Figure 9.3 shows the icons of the SYSMOD stereotypes for actors.

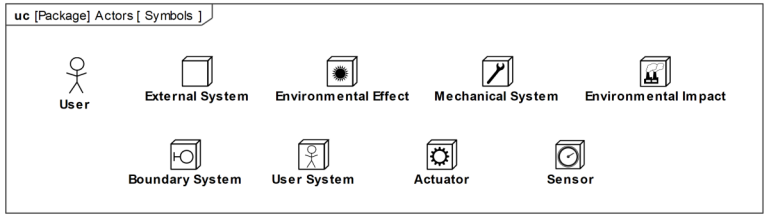

Figure 9.3: Icons for SYSMOD actor stereotypes

3.3 Blocks

SYSMOD defines several stereotypes for SysML *Block* and *InterfaceBlock*.

The stereotype *«userInterface»* marks an interface block that specifies an interaction point between the system and a human. These interfaces are different from technical interfaces like an API, or a mechanical connection. For example, a user interface defines operations that represent the usage of a lever or button.

The stereotype *«system»* marks a block that represents a system. A model can have more than one system element. For example, a system element for the Base Architecture (see section 8.6: Example Base Architecture), and another one for the Logical Architecture (see section 8.15: Example Logical Architecture). Typically, all system elements are specializations of the abstract system element. The system stereotype defines properties for the System Idea and the Problem Statement.

To define another level of a system breakdown structure, the stereotype *«subsystem»* is applied to blocks that represent system-like parts of the system.

The stereotype *«systemContext»* is a block that specifies the communication links between the system and the system actors. Internal block diagrams of the system context element depict the System Context.

The stereotype *«documentBlock»* represents a document and can be used as a proxy for the information of the document in the model. For example, a document block that represents an interface specification document and is used as the type

of a port. The property *reference* stores the source of the document, for example, a URL or a link to the document in a file system.

The stereotype *«domainBlock»* is an element of the Domain Knowledge (section 5.15). It represents a concept of the domain that is "known" by the system.

Figure 9.4 depicts the definitions of the SYSMOD stereotypes for SysML *Blocks*.

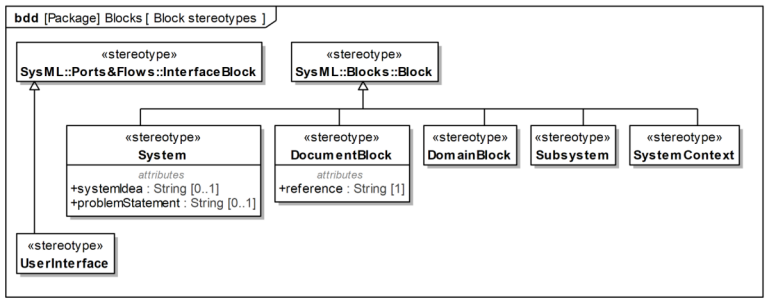

Figure 9.4: SYSMOD stereotypes for SysML *Blocks*

3.4 Discipline-specific Elements

As a rule of thumb, the model elements at the lowest level of the system breakdown structure can be fully allocated to a specific engineering discipline. For example, a block that represents a pure software, mechanical, or electrical artifact.

The SYSMOD stereotypes for discipline-specific elements are markers for those elements. The base class is *NamedElement*, i.e., the stereotypes can be applied to any model element that has a name. Typically, it is used for blocks, parts, and connectors.

The property *reference* stores a link to an external model or document that covers the details of the discipline-specific element.

Figure 9.5 depicts the definitions of the SYSMOD stereotypes for discipline-specific elements: *«software»*, *«mechanical»*, and *«electrical»*. You can define our own additional stereotypes if you need more discipline-specific elements.

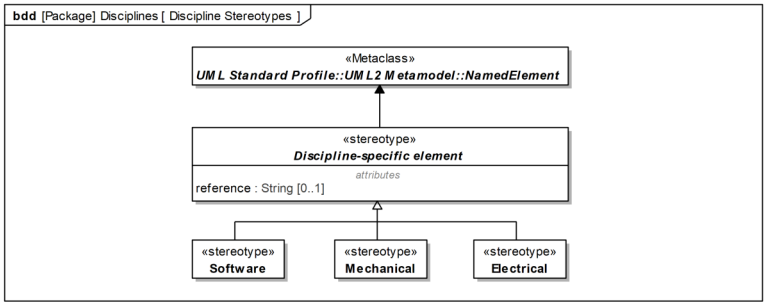

Figure 9.5: SYSMOD stereotypes for discipline-specific Elements

3.5 Relationships

The SysML relationship *Satisfy* specifies that the element at the source of the relationship satisfies the requirement at the target. It works well for one-to-one but is not precise for many-to-many relationships.

If, for example, two blocks have satisfy relationships to a single requirement in SysML (figure 9.6), it is not specified what that means. Does block A and B together satisfy requirement R42? Or each block alone and the requirement R42 is satisfied twice?

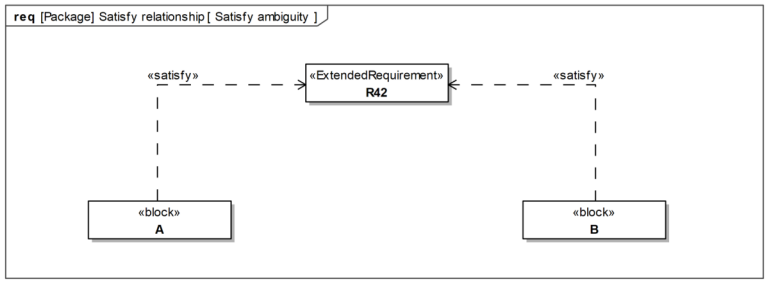

Figure 9.6: Ambiguity of the satisfy relationship

The SYSMOD stereotype *«weightedSatisfy»* adds a property to the satisfy relationship that specifies the coverage of the satisfaction. Figure 9.7 shows the same scenario of figure 9.6 with the weighted satisfy relationship. Block A satisfies 70% and block B 30% of the requirement R42.

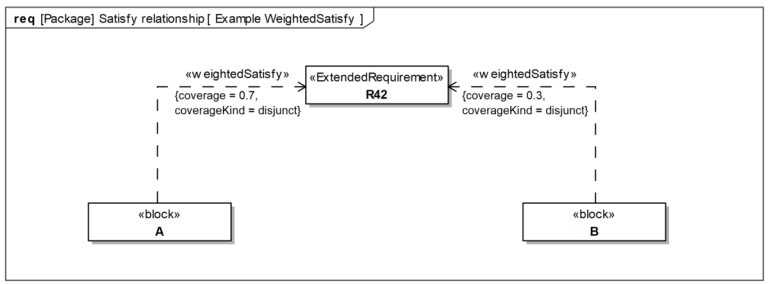

Figure 9.7: Example of SYSMOD weighted satisfy relationship

The property *coverageKind* specifies if the coverage is disjunct or overlapping. Overlapping means that the elements at the source of the *«weightedSatisfy»* relationships satisfy equal parts of the requirement. For example, if the requested feature should be implemented twice for redundancy reasons. In such a case, a satisfy coverage of 70% + 30% is not a 100% coverage of the requirement. If the property *coverageKind* is set to *disjunct*, it specifies a non-overlapping coverage. *Disjunct* is the default.

The same principle is applied to the SysML relationships *Verify* and *Allocate* by the SYSMOD stereotypes *«weighted-Verify»* and *«weightedAllocate»*.

The SysML relationship *DeriveReqt* specifies that a requirement is derived from another requirement. The SYSMOD stereotype *«exDeriveReqt»* adds a property to specify the model elements that led to the derivation, for example, architecture elements, as described in the zigzag pattern (see section 11.6). Figure 9.8 shows an example of the extended derive relationship.

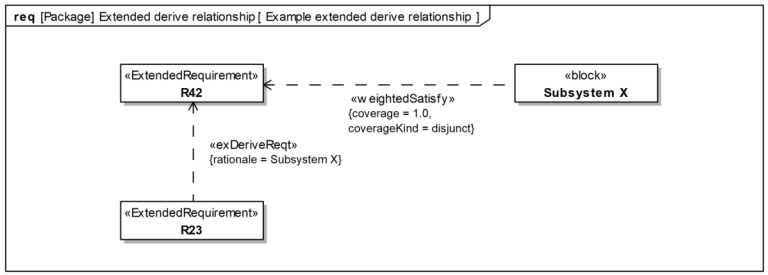

Figure 9.8: Example of SYSMOD extended derive requirement relationship

The SYSMOD stereotype *«conjugated»* is deprecated since SYSMOD v4.2 because SysML v1.6 resolved the issue that was resolved by this stereotype for older SysML versions. It is still part of SYSMOD to support models that are based on SysML < v1.6. The following paragraphs explain the issue and motivation for the stereotype.

The proxy ports of a block are typed with SysML *InterfaceBlocks*. To ensure that the block implements the features specified by its ports, it typically has a generalization relationship to the interface block (figure 9.9).

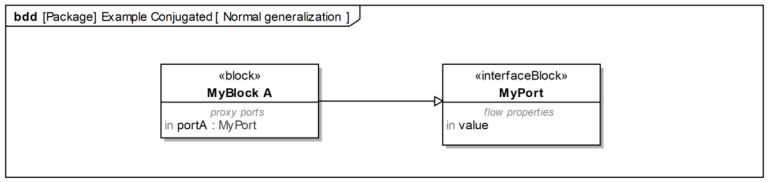

Figure 9.9: Example interface block generalization

If a proxy port is connected to another port by a connector, the connected port has typically the same type but is a conjugated port to invert the direction of the flow properties and directed features specified within the interface block (figure 9.10).

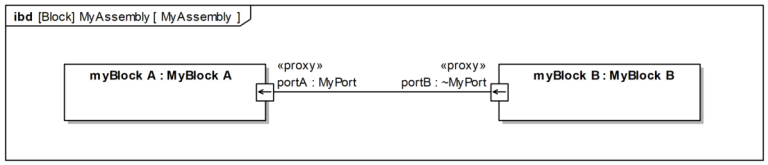

Figure 9.10: Example connected proxy ports

The block *MyBlock B* should also have a generalization relationship to the proxy port type *MyPort* to ensure that the specified features are implemented by *MyBlock B*. However, we need a conjugated type to invert the directions of the flow properties and directed features. The SYSMOD stereotype *«conjugated»* is a generalization relationship that has the same effect as the *isConjugated* property of a SysML *Port*. The direction of the inherited directed features is conjugated (figure 9.11).

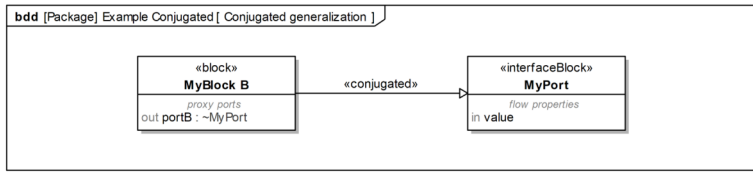

Figure 9.11: Example conjugated generalization

SysML v1.6 does not resolve the issue by a conjugated generalization relationship but by a special conjugated interface block. See section 12.6 for details about the SysML *InterfaceBlock* and *~InterfaceBlock*.

Figure 9.12 depicts the definitions of the SYSMOD stereotypes for relationships.

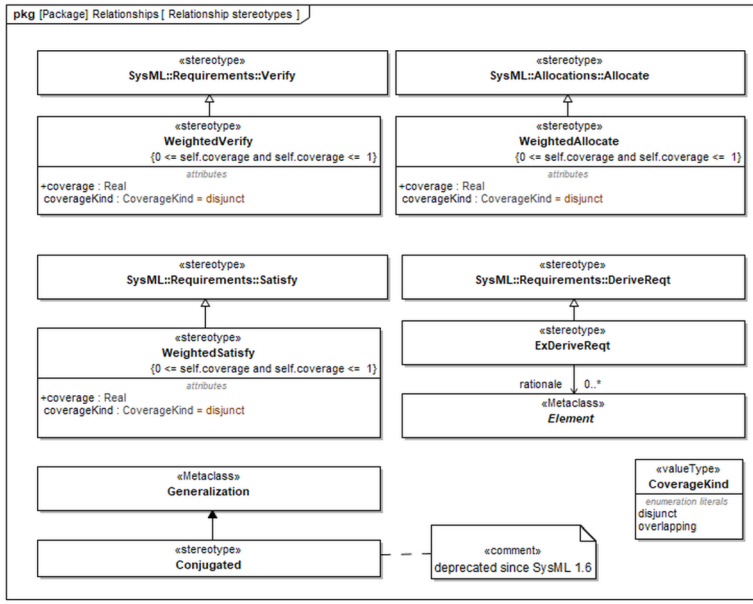

Figure 9.12: SYSMOD stereotypes for Relationships

3.6 Requirements and Risks

Figure 9.13 depicts the definition of the SYSMOD stereotypes for requirements.

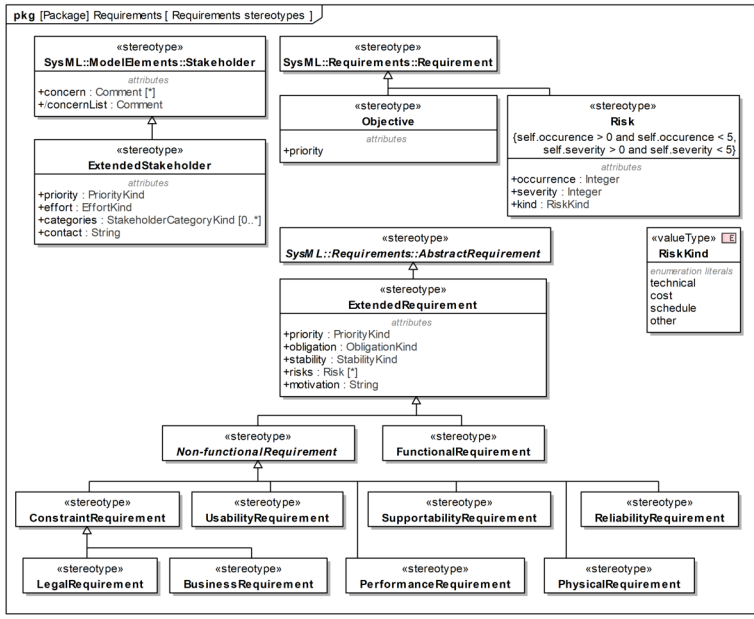

Figure 9.13: SYSMOD stereotypes for Requirements

SysML provides only a general requirement model element that covers the name, ID, and the text of a requirement. The SYSMOD stereotype *«extendedRequirement»* adds additional properties typically important for Requirements:

Priority	Specifies the importance of the requirement.
Obligation	Specifies if the requirement is optional or mandatory.

Stability	Specifies if it is likely that the requirement will change in the future.
Risk	Specifies a list of risks for the implementation of the requirement.
Motivation	Specifies why the requirement is necessary.

The SYSMOD stereotype *«extendedRequirement»* can be applied to any model element that has a name, for example, a state machine or a constraint block to designate that model element itself as a requirement. It is not necessary to translate activities or state machines that represent requirements to textual requirements.

SYSMOD provides further specializations of the stereotype *«extendedRequirement»*. They do no add more properties, but specify each a different kind of a requirement: *«functionalRequirement»*, *«non-functionalRequirement»*, *«physicalRequirement»*, *«usabilityRequirement»*, *«constraintRequirement»*, *«reliabilityRequirement»*, *«performanceRequirement»*, *«legalRequirement»*, *«businessRequirement»*, *«supportabilityRequirement»*.

The SYSMOD stereotype *«objective»* specializes the SysML stereotype *«requirement»* and represents System Objectives (section 5.6).

The stereotype *«extendedStakeholder»* extends the SysML *Stakeholder* to add additional properties that are used for the SYSMOD Stakeholder (section 5.8).

SYSMOD extends the SysML *Requirement* element to provide a model element for Risks. In the broadest sense, a Risk is a potentially undesirable situation that must be considered by the system or systems engineering project. The stereotype *«risk»* adds the following properties

occurrence	Occurrence rating of the risk, for example, 1 - 10 (10 = highest occurrence).
severity	Severity rating of the risk, for example, 1 - 10 (10 = highest severity).
kind	Classification of the risk: technical, cost, schedule, other.

Since the metaclass *NamedElement* of the stereotype *«extendedRequirement»* is abstract, you must specify a concrete model element when you apply the stereotypes. It can be any kind of a named element.

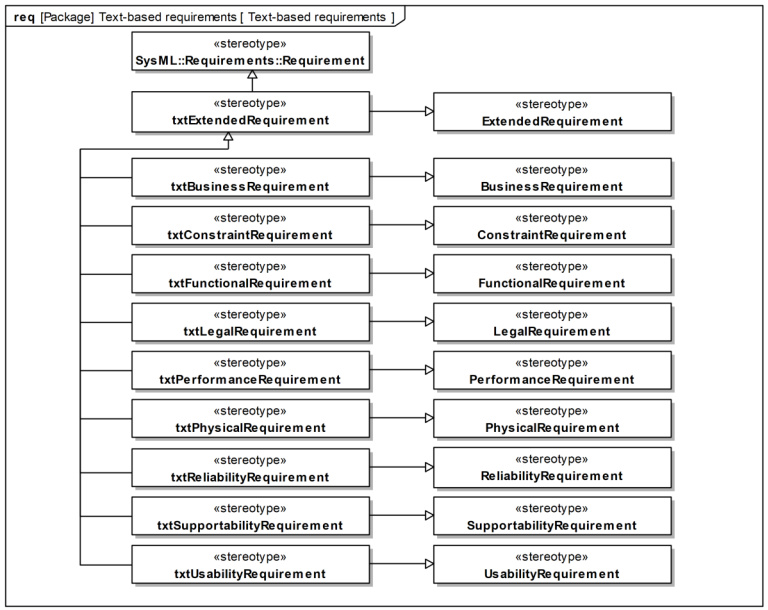

Figure 9.14: SYSMOD stereotypes for classical textual Requirements

Figure 9.14 depicts SYSMOD requirement stereotypes that specializes the concrete SysML *Requirement* instead of the abstract SysML *AbstractRequirement* element. The stereotypes for classical textual requirements have the prefix *txt*.

Figure 9.15 depicts the SYSMOD stereotype *«performanceConstraintRequirement»* which specializes the SysML *AbstractRequirement* and the SysML *ConstraintBlock* element. Figure 8.26 shows an example of how to use it.

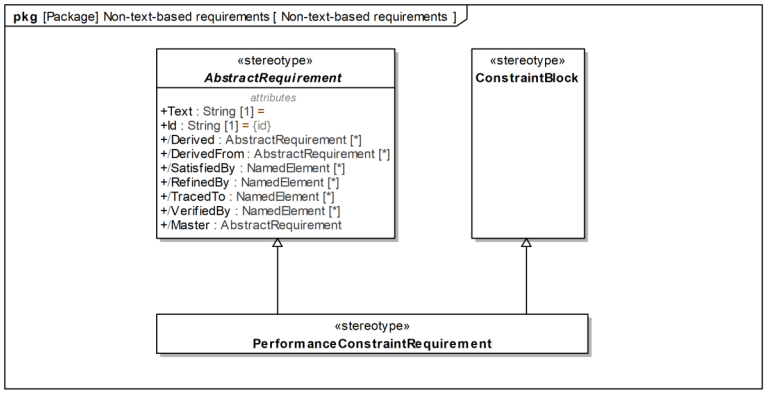

Figure 9.15: SYSMOD stereotype for parametric-based performance requirement

3.7 Test Cases

The abstract stereotype *«extendedTestCase»* specializes the SysML test case by adding the following properties. They are similar to the definitions in the UML Testing Profile [UTP13]. If you need more test modeling features, you should use the complete UML Testing Profile in addition to SysML and SYSMOD.

priority	Priority of the test case for test case management.
type	Type of the test case. The set of types is not predefined.

componentUnderTest	References the block that is tested by the test case, for example, the whole system or a subsystem.

The *priority* may be used to support test planning activities.

The *type* describes the type of the test case, for example, usability test or performance test.

The *componentUnderTest* references the block that is in the focus of the test case, for example, the whole system itself, a subsystem, or any other block of the system architecture.

The specialized stereotype *«systemTestCase»* is used to specify test cases that tests the real physical system. The stereotype *«modelTestCase»* is used to specify test cases that test the model, i.e., the specification of the system.

The following figure 9.16, depicts the definition of the stereotypes.

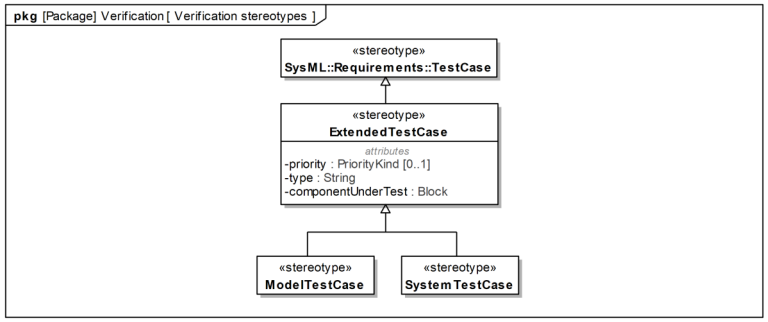

Figure 9.16: SYSMOD stereotypes for Test Cases

3.8 Use Cases

The stereotype *«systemUseCase»* adds more properties to the SysML *UseCase* element. A property to specify the trigger that starts the System Use Case, and a property for the result of the System Use Case (section 5.12).

You can specify a descriptive textual trigger, or reference a signal model element as a trigger that, for example, can be used in a state machine to trigger transitions (see figure 8.64 in section 8.18).

The stereotype *«continuousUseCase»* is a specialized *«systemUseCase»* to model a continuous behavior. A continuous use case must own a *«continuousActivity»*. The constraint is defined with OCL[1] as follows:

```
context ContinuousUeCase inv ContinuousUseCaseOwnsC\
ontinuousActivity:
self.classifierBehavior.oclIsTypeOf(SYSMOD::Activit\
ies::ContinuousActivity)
```

The SYSMOD stereotype *«systemProcess»* is applied to use cases that represent a System Process (section 5.13), that means, a specification of the logical order of execution of Use Case Activities.

Figure 9.17 depicts the definitions of the SYSMOD stereotypes for use cases.

[1]Object Constraint Language (OCL14).

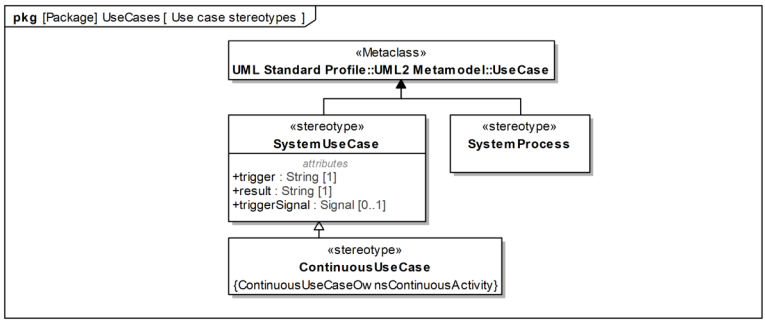

Figure 9.17: SYSMOD stereotypes for Use cases

3.9 Variants (VAMOS stereotypes)

The modeling of variants is a topic in itself. I call the method for modeling variants with SysML VAMOS (VAriant MOdeling with SysML). You find a detailed description of VAMOS and examples in the book *Variant Modeling with SysML*[2]. A brief overview is given in section 11.5.

The SYSMOD stereotypes depicted in figure 9.18 are necessary to model the variant modeling approach: *«variationPoint»*, *«variant»*, *«variation»*, *«variantConfiguration»*, *«XOR»*, *«REQUIRES»*, and the enumeration *BindingTimeKind*.

[2] https://leanpub.com/vamos

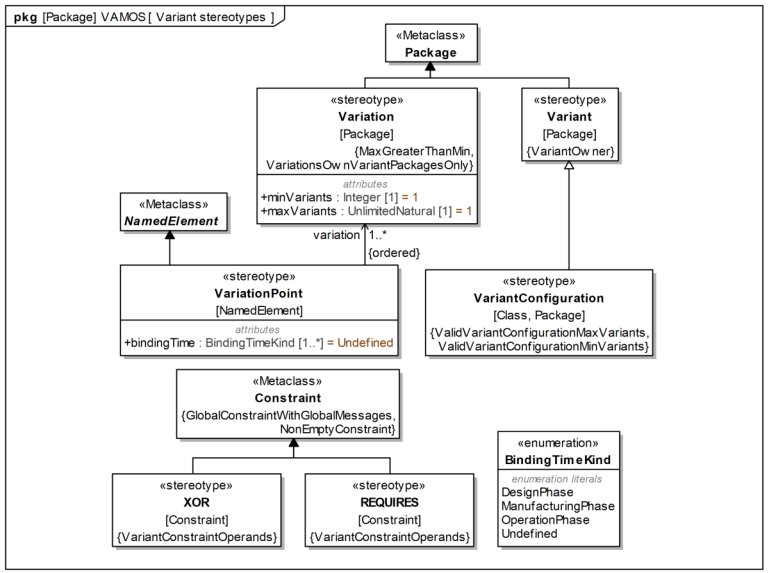

Figure 9.18: SYSMOD stereotypes for Variants

3.10 SYSMOD Enumerations

The *SYSMOD Enumerations* package contains enumeration types used by SYSMOD stereotypes: *EffortKind*, *ObligationKind*, *PriorityKind*, *RiskKind*, *StabilityKind*, and *StakeholderCategoryKind* (figure 9.19).

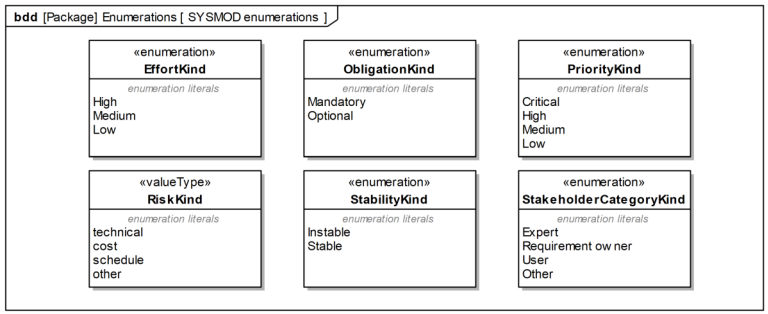

Figure 9.19: SYSMOD Profile Library: Enumerations

3.11 SYSMOD PartsCatalogue Library

The model library *PartsCatalogue* provides common model elements useful for system modeling. The library is independent of the SYSMOD profile and vice versa. The library is intended to be an example of a model library, and not to be a complete library.

The current version of the library includes four sections: Analysis, Constraints, Physics, and Units (figure 9.20).

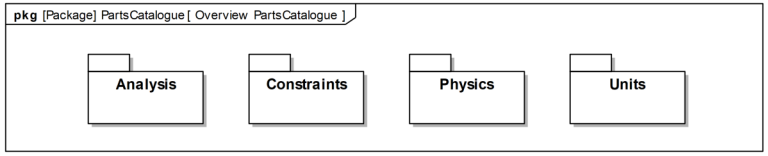

Figure 9.20: SYSMOD PartsCatalogue Library

The *Analysis* package contains an abstract block *PerformanceBehaviorProperties* (figure 9.21). The properties store the minimum, maximum, and medium duration of a behavior. The

appropriate derived properties *totalXXX* sum up the values of a breakdown structure, for example, an activity tree.

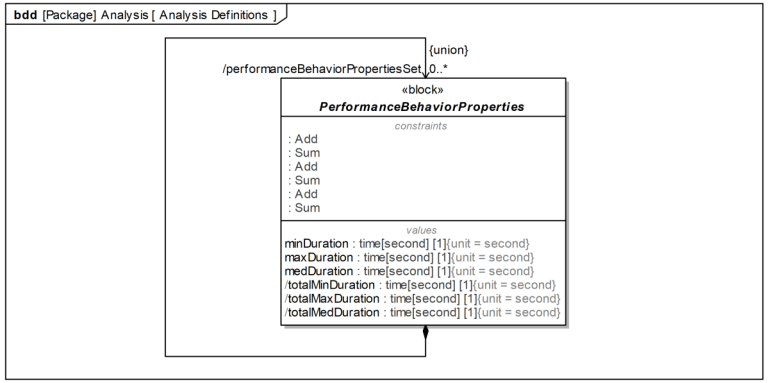

Figure 9.21: SYSMOD PartsCatalogue Library: Analysis

The calculation of the sums is performed by the constraint properties specified in the parametric diagram depicted in 9.22.

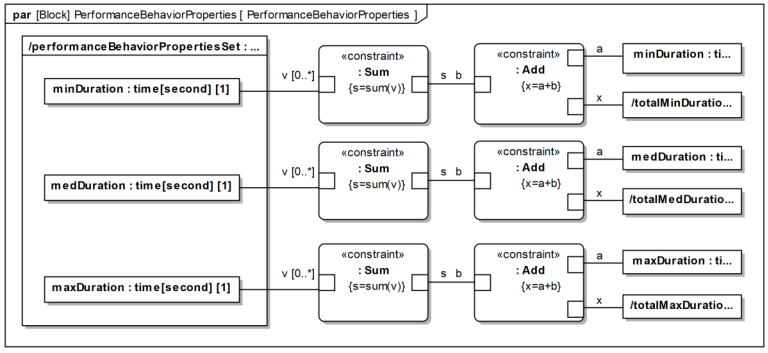

Figure 9.22: SYSMOD PartsCatalogue Library: Analysis Parametrics

The *Constraints* package includes two simple constraint blocks, *Add* and *Sum* (figure 9.23) used, for example, in the *Analysis* package above.

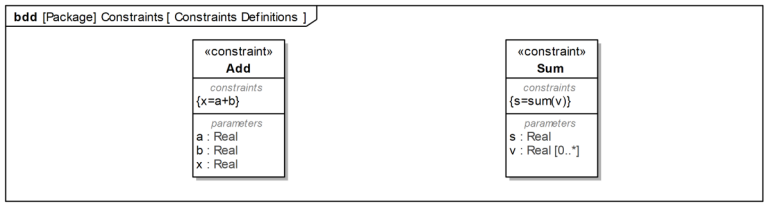

Figure 9.23: SYSMOD PartsCatalogue Library: Constraints

The *Physics* package includes an abstract block as a base block for blocks specifying physical things with a mass (figure 9.24). Similar to the *PerformanceBehaviorProperties* block in the *Analysis* package, the *PhysicalElement* provides the capability to sum up the masses of a breakdown structure.

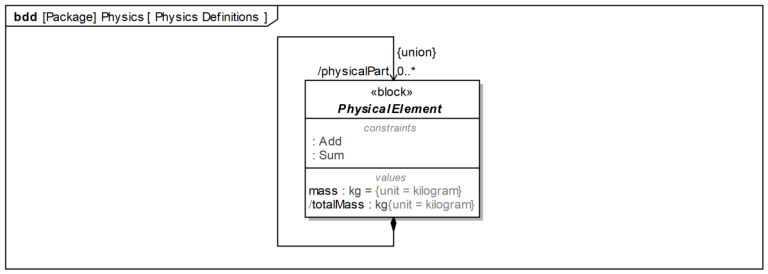

Figure 9.24: SYSMOD PartsCatalogue Library: Physics

Figure 9.25 shows an example of how to use the *PhysicalElement*. If you create an instance specification of the block *Smoke Sensor*, the value property *totalMass* has the value 0.9 kg. Of course only, if your modeling provides the necessary calculation capabilities.

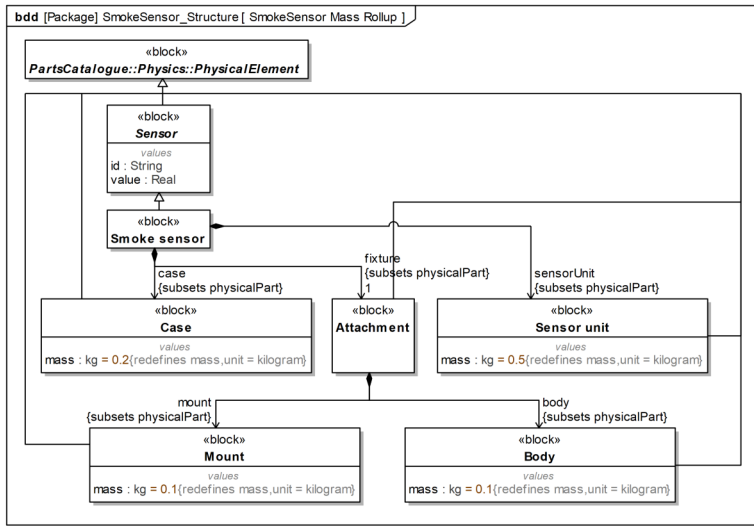

Figure 9.25: SYSMOD PartsCatalogue Library: Physics Example

The *Units* package includes only two value types specifying kilogram and timestamp values (figure 9.26). They are used in the Physics package, and the model of the FFDS.

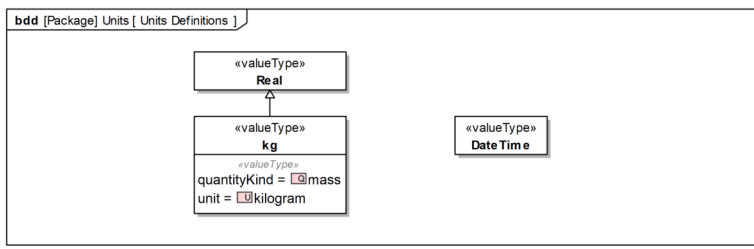

Figure 9.26: SYSMOD PartsCatalogue Library: Units

3.12 SYSMOD Simulation Library

The *SYSMOD SimLib* library provides some elements useful to create an executable model (figure 9.27). The library is

only a simple example of a model library. Typically, you need many more additional elements to create an executable system model.

The library also includes a profile with the stereotype *«simElement»* to tag elements in the model that only exists to enable the model execution, for example, an action in an activity that was modeled just for executability (see figure 8.54).

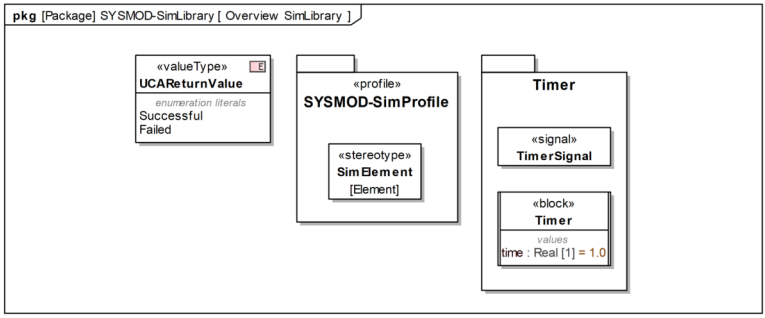

Figure 9.27: SYSMOD Simulation Library

3.13 SYSMOD Engineering4Planet Library

The SYSMOD *Engineering4Planet* Library provides two elements to model sustainability aspects (figure 9.28); see also section 2.

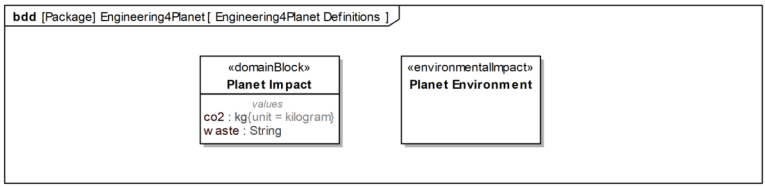

Figure 9.28: SYSMOD Engineering4Planet Library

PlanetEnvironment is an actor that, typically, should be considered in any system context.

PlanetImpact is a type of an item flow to model the impact of the system of interest on the planet.

Appendix A: Mapping ISO 15288 to SYSMOD

The following list shows a mapping of the ISO 15288 processes [ISO15288] to the SYSMOD Methods whereby the mapping makes no distinction between full or partial coverage:

A.1 ISO 15288: Technical Processes

ISO 15288	*SYSMOD*
Business Mission Analysis: Defines business or mission problem and characterize the solution space.	4.5 Analyze the Problem, 4.6 Describe the System Idea and the System Objectives, 4.7 Describe the Base Architecture, 4.8 Identify Stakeholders, 4.11 Identify the System Context

ISO 15288	*SYSMOD*
Stakeholder Needs and Requirements Definition: Define the stakeholder requirements.	4.8 Identify Stakeholders, 4.10 Model Requirements, 4.11 Identify the System Context, 4.12 Identify System Use Cases, 4.13 Identify System Processes, 4.14 Model Use Case Activities, 4.15 Model the Domain Knowledge, 4.21 Define System States
System requirements definition: Transform the stakeholder-oriented desired capabilities into a technical view of a solution.	4.10 Model Requirements, 4.12 Identify System Use Cases, 4.13 Identify System Processes, 4.14 Model Use Case Activities, 4.15 Model the Domain Knowledge, 4.21 Define System States
Architecture Definition: Create system architecture alternatives and select the most appropriate one(s).	4.11 Identify the System Context, 4.17 Model the Functional Architecture, 4.18 Model the Logical Architecture, 4.20 Revise an Architecture with Scenarios, 4.21 Define System States

ISO 15288	*SYSMOD*
Design Definition: Provide detailed information about the system for implementation aligned with the architecture.	4.19 Model the Product Architecture, 4.20 Revise an Architecture with Scenarios, 4.21 Define System States
System Analysis: Provide data to aid decision-making.	not explicitly covered
Implementation: Realize a specified system element.	not explicitly covered
Integration: Synthesize a set of elements into a realized system.	not explicitly covered
Verification: Provide evidence that a system fulfills its requirements.	4.16 Specify Test Cases, 4.22 Model the Test Architecture
Transition: Transfer of custody of the system from development to operation organizational entities.	not explicitly covered
Validation: Provide evidence that a system fulfills its intended use.	4.16 Specify Test Cases, 4.22 Model the Test Architecture
Operation: Perform operation of the system.	not explicitly covered

ISO 15288	***SYSMOD***
Maintenance: Perform maintenance to sustain the system services.	not explicitly covered
Disposal: End the existence of a system.	not explicitly covered

A.2 ISO 15288: Technical Management Processes

ISO 15288	*SYSMOD*
Project Planning: Produce and coordinate project plans.	not explicitly covered
Project Assessment and Control: Assess and control the plans.	not explicitly covered
Decision Management: Provide a framework for decisions.	not explicitly covered
Risk Management: Identify and manage risks.	4.9 Model Risks
Configuration Management: Manage system elements and configurations over the life cycle.	not explicitly covered
Information Management: Perform and coordinate communication of information with the stakeholders.	not explicitly covered
Measurement: Obtain objective data to support the quality management of the system.	not explicitly covered

ISO 15288	*SYSMOD*
Quality Assurance: Ensure the application of the organization's quality management process.	not explicitly covered

A.3 ISO 15288: Agreement Processes

ISO 15288	*SYSMOD*
Acquisition: Obtain a product or service.	not explicitly covered
Supply: Provide a product or service.	not explicitly covered

A.4 ISO15288: Organizational Project-Enabling

ISO 15288	*SYSMOD*
Life Cycle Model Management: Manage life cycle models, processes, and policies.	not explicitly covered
Infrastructure Management: Provide infrastructure and services to projects.	4.1 Tailor the MBSE Methodology, 4.2 Set up and maintain the SME, 4.3 Deploy the MBSE Methodology

ISO 15288	***SYSMOD***
Portfolio Management: Initiate suitable projects to meet the strategic objectives of the organization.	not explicitly covered
Human Resource Management: Provide the necessary human resources.	4.4 Provide MBSE Training and Coaching
Quality Management: Assure that the quality management process meets organizational and project quality objectives.	not explicitly covered
Knowledge Management: Create the capability to re-apply knowledge.	not explicitly covered

Bibliography

[Ar65] Bruce L. Archer. Systematic Method for Designers. Council of Industrial Design. H.M.S.O.. 1965.

[Bl56] Benjamin S. Bloom. Taxonomy of educational objectives: The classification of educational goals. Handbook I. David McKay. 1956.

[Br09] Tim Brown. Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. HarperBusiness. 2009.

[Cr81] Philip B. Crosby. The Art of Getting Your Own Sweet Way. McGraw-Hill. 1981.

[DaKl14] Matthias Dänzer, Sven Kleiner, Jesko G. Lamm, Georg Moeser, Fabian Morant, Florian Munker, Tim Weilkiens. Funktionale Systemmodellierung nach der FAS-Methode: Auswertung von vier Industrieprojekten. Tag des Systems Engineering (TdSE) 2014. Bremen. 12. – 14. November 2014.

[De10] John Dewey. How we think. D.C. Heath & Co. 1910.

[Es08] Jeff A. Estefan. Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE MBSE Initiative. 2008.

[Gy12] Craig Gygi, Bruce Williams, Neil DeCarlo, Stephen R. Covey. Six Sigma For Dummies. 2nd Edition. 2012.

[Ha19] Reinhard Haberfellner, Olivier de Weck, Ernst Fricke, Siegfried Vössner. Systems Engineering - Fundamentals and

Applications. Birkhäuser Basel. 2019.

[Im86] Masaaki Imai. Kaizen: The Key to Japan's Competitive Success. McGraw-Hill/Irwin. 1986.

[ISO15288] ISO/IEC/IEEE 15288:2015. Systems and software engineering — System life cycle processes. 2015.

[ISO195051] ISO/IEC 19505-1:2012. Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure. 2012.

[ISO195052] ISO/IEC 19505-2:2012. Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure. 2012.

[ISO42010] ISO/IEC/IEEE 42010:2011. Systems and software engineering – Architecture description. 2011.

[LaWe10] Jesko G. Lamm, Tim Weilkiens. Functional Architectures in SysML. In M. Maurer and S.-O. Schulze (eds.). Tag des Systems Engineering 2010. pp. 109–118. Carl Hanser Verlag. Munich. Germany. November 2010.

[LaWe14] Jesko G. Lamm, Tim Weilkiens. Method for deriving functional architectures from use cases. Systems Engineering. 17(2):225-236. 2014.

[LaM17] Jesko G. Lamm, Andreas Mettauer, Georg Moeser, Tim Weilkiens, Albert Albers. Storyboards in der Systementwicklung: eine neue Methode und ihr Zusammenspiel mit der FAS-Methode. In S.-O. Schulze (eds.). Tag des Systems Engineering 2017. pp. 105–114. Carl Hanser Verlag. Munich.

[Le18] Michael Lewrick, Patrick Link, Larry Leifer. The Design Thinking Playbook: Mindful Digital Transformation of Teams, Products, Services, Businesses and Ecosystems. Wiley.

2018.

[Ma96] James N. Martin. Systems Engineering Guidebook: A Process for Developing Systems and Products. CRC Press, Inc.. 1996.

[MBSECook] Robert Karban, Tim Weilkiens, et al.. MBSE Cookbook. http://mbse.gfse.de

[Mo98] Niko Mohr, Jens Marcus Woehe. Widerstand erfolgreich managen: Professionelle Kommunikation in Veränderungsprojekten. Campus Verlag. 1998.

[MOF16] Object Management Group. Meta Object Facility, Version 2.5.1. formal/19-10-01.

[OCL14] Object Management Group. Object Constraint Language, Version 2.4. formal/14-02-03.

[Oh88] Taiichi Ohno. Toyota Production System: Beyond Large-Scale Production. Cambridge, MA: Productivity Press. 1988.

[Oh20] Taiichi Ohno. "Ask 'why' five times about every matter". https://www.toyota-myanmar.com/about-toyota/toyota-traditions/quality/ask-why-five-times-about-every-matter. Retrieved February 2020.

[Os14] Alexander Osterwalder, Yves Pigneur, Gregory Bernarda, Alan Smith. Value Proposition Design: How to Create Products and Services Customers Want. John Wiley & Sons. 2014.

[Pi16] Roman Pichler. Strategize: Product Strategy and Product Roadmap Practices for the Digital Age. Pichler Consulting. 2016.

[SE16] BKCASE Editorial Board. 2016. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.6. R.D.

Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed August 2016. www.sebokwiki.org.

[SysML07] Object Management Group. OMG Systems Modeling Language (OMG SysML), Version 1.0. formal/2007-09-01.

[SysML19] Object Management Group. OMG Systems Modeling Language (OMG SysML), Version 1.6. formal/19-11-01.

[UML17] Object Management Group. Unified Modeling Language (UML), Version 2.5.1. formal/17-12-05.

[UTP13] Object Management Group. UML Testing Profile (UTP), Version 1.2. formal/2013-04-03.

[We08] Tim Weilkiens. Systems Engineering with SysML/UML. Morgan Kaufmann. 2008.

[We14] Tim Weilkiens. Systems Engineering mit SysML/UML. 3rd edition. dpunkt. 2014.

[We15] Tim Weilkiens, Jesko G. Lamm, Stephan Roth, Markus Walker. Model-Based System Architecture. Wiley. 2015.

[We16] Tim Weilkiens. Variant Modeling with SysML. MBSE4U. 2016.

[We18] Tim Weilkiens. Alexander Huwaldt. Jürgen Mottok. Stephan Roth. Andreas Willert. Modellbasierte Softwareentwicklung für eingebettete Systeme verstehen und anwenden. dpunkt. 2018.

[Wo90] James P. Womack, Daniel T. Jones, Daniel Roos. Machine that Changed the World. Rawson Associates. 1990.

Index

«~interfaceBlock» 304
«actor» 290
«actuator» 212
«allocate» 328
«block» 299
«boundarySystem» 212
«businessRequirement» 223
«conjugated» 219
«constraint» 321
«constraintRequirement» 223
«continuousActivity» 212, 227
«continuousUseCase» 72, 130, 227
«deriveReqt» 325
«documentBlock» 215
«domainBlock» 76, 139, 216
«electrical» 217
«environmentalEffect» 212
«equal» 308
«exDeriveReqt» 219
«extendedRequirement» 67, 222
«extendedStakeholder» 65, 120, 223
«extendedTestCase» 225
«externalSystem» 212
«full» 303
«functionalRequirement» 187, 223
«interfaceBlock» 304
«legalRequirement» 223
«mechanical» 217
«mechanicalSystem» 212

«modelTestCase» 78, 226
«non-functionalRequirement» 223
«objective» 62, 114, 223
«performanceRequirement» 223
«physicalRequirement» 223
«problem» 330
«proxy» 303
«rationale» 330
«refine» 326
«reliabilityRequirement» 223
«REQUIRES» 228
«risk» 66, 223
«satisfy» 327
«sensor» 212
«simElement» 234
«software» 217
«subsystem» 215
«supportabilityRequirement» 223
«system» 59, 60, 69, 113, 144, 163, 215
«systemContext» 69, 126, 215
«systemProcess» 133, 227
«systemTestCase» 78, 226
«systemUseCase» 72, 130, 227
«trace» 325
«usabilityRequirement» 223
«use» 331
«user» 212
«userInterface» 215
«userSystem» 212
«variant» 228
«variantConfiguration» 228
«variation» 228
«variationPoint» 228
«verify» 326
«weightedAllocate» 219
«weightedSatisfy» 218

«weightedVerify» 219
«XOR» 228
3 Amigos 280
4-layer architecture 286
6M method 275

A

abstract requirement 224
abstract syntax 283
accept event action 294, 306
action 293
activity 134, 136, 211, 293
activity diagram 73, 75, 78, 136, 190, 292
activity final node 296
activity parameter 293
actor 68, 69, 185, 212, 256, 290
actuator 214
adjunct property 75, 301
adoption 237
aggregation 300, 305
aggregation kind 300
agile 7
allocate 219, 327
allocation activity partition 328
alt 314
anti-pattern 239
any receive event 317
architecture kinds 78
artificial intelligence 14
assert 314
association 300, 304
asynchronous message 312

B

base architecture 27, 62, 78, 83, 116, 124, 163, 171, 181, 246, 269, 270
beermat architecture 63, 171
behavior 302
behavior port 303
behavioral feature 302
bill of material see BOM
binding connector 308, 320
binding time 228
BindingTimeKind 228
block 215, 298
block definition diagram 63, 69, 75, 76, 78, 81, 84, 85, 88, 117, 126, 139, 141, 144, 155, 173, 182, 198, 207, 298
Bloom taxonomy 91
BOM 263
Booch, Grady 279
boundary system 214
break 314
business requirement 223

C

call behavior action 133, 136, 294
call event 317
Cameo Systems Modeler 159
center of competence 96
change event 317
change process 19, 238
choice 319
coaching 20

collaborative engineering 14
combined fragment 313
comment 330
communication 261
competency 91
componentUnderTest 226
composition 301, 305
concept model 41
concrete syntax 283
configuration management 14, 56
conjugated generalization 220
conjugated port 303
conjugation 304
connector 308
consider 314
constraint 330
constraint block 177, 321
constraint parameter 321
constraint property 321
constraint requirement 27, 124, 223
containment 288
continuous activity 211
continuous use case 130, 227
control flow 295
coupling 270
coverageKind 218
create message 312
critical 314
cross-cutting elements 329

D

data analytics 14

data model 41
decision node 296
deep history 319
dependency 331
dependency management 289
deployment 19
derive 325
deriveReqt 219
Design 26
Design Thinking 22, 58
digital twin 14
discipline-specific element 216
disruptive innovation 28
do behavior 316
document block 215
domain block 139, 192, 216
domain knowledge 40, 75, 136, 138, 191, 216, 249, 251
domain object 40, 76

E

effect 317
EffortKind 229
electrical 217
elevator pitch 25, 60
Engineering4Planet 211, 234
entry behavior 316
entry point 319
enumeration type 299
environmental effect 213
environmental impact 214
exit behavior 316
exit point 319

external system 213

F

FAS method 13, 44, 80, 83, 159, 257
FFDS 159
final state 318
five whys method 276
flat history 319
flow property 303
Ford, Henry 22
forest fire detection system see FFDS
fork 319
fork node 297
full port 263, 302
function bucket 71
functional architecture 44, 78, 79, 83, 257, 270
functional decomposition 39, 74
functional requirement 36, 177, 184, 223
functional safety 14

G

generalization 306
guard 317

H

hardware-in-the-loop see HIL
HIL 51

I

ignore 314

import 289
INCOSE 282
inheritance 306
initial node 296
initial state 318
input pin 293
instance specification 300
intensity model see model purpose model
interaction 86, 150, 310
interaction operator 313
interface block 215, 303
internal block diagram 63, 69, 81, 84, 85, 88, 118, 127, 144, 156, 173, 181, 199, 207, 259, 307
ISO 15288 13, 333
ISO/IEC 2010 81
ISO/IEC/IEEE 24748-4 80
ISO/IEC/IEEE 42010:2011 83
item flow 181, 192, 309

J

Jacobson, Ivar 279
join 319
join node 297
junction 319

K

Kaizen 276

L

language architecture 283

lean manufacturing 276
learning curve 243
legal requirement 223
lifeline 311
logical architecture 45, 47, 78, 82, 84, 143, 197, 268, 270
loop 314
loose coupling 271

M

machine learning 14
MARTE 323
matrix 62, 65, 66, 67, 72, 259
MBSE 237, 243, 262
MBSE Methodologist 96
MBSE4U 9
mechanical 217
mechanical system 213
merge node 296
message 312
Meta Object Facility see MOF
metamodel 284
method 17, 2, 15, 107, 302
methodology 3, 16, 19, 54, 96, 237, 243
methodology context 247
minimal SysML 286
model 288
model library 56, 193, 211, 230, 233, 234, 262, 289
model purpose model 19, 261
model structure 107, 161
model template 110
model test case 43, 77, 195, 226

Modeling And Analysis Of Real-Time Embedded Systems see MARTE
modeling guidance 107
modeling language 251
modeling tool 18, 94, 253
model-supported systems engineering see MSSE
MOF 285
motivation 223
MSSE 261
multiplicity 301

N

named element 216, 224
namespace 288
napkin architecture 63, 171
neg 314
nested property 320
non-functional requirement 223

O

Object Constraint Language see OCL
object flow 295
Object Management Group see OMG
Object Modeling Technique see OMT
obligation 222
ObligationKind 229
occurrence 224
OCL 227, 283

OCUP 281
Ohno, Taiichi 276
OMG 280
OMG-Certified UML Professional see OCUP
OMT 279
OOSE 279
OpaqueAction 294
operation 302
opt 314
output pin 293
overmodeling 242

P

package 288
package diagram 107, 161, 287
package structure 107
par 314
parameter 302
parametric diagram 319
part property 300
parts catalogue 211, 230
performance requirement 177, 223
physical architecture 45, 47, 78, 81, 258
physical requirement 223
PhysicalElement 232
pin 293
Planet Environment 6, 34, 127, 235
Planet Impact 127, 235
port 263, 302
prioritization 29
priority 222, 225
PriorityKind 229
problem 22, 330

problem statement 58, 110, 162, 215, 245, 275
problem-solving process 23
process 17, 3, 7
process model 55
product 53
product architecture 47, 78, 82, 84, 147, 200, 270
product box 26, 112, 113, 272
product line 265
product tree 198
product vision board 26, 112, 113, 273
profile 56, 209, 321
project manager 98
property 300
proxy port 263, 303
pseudostate 319
purpose-driven methodology 237
query-driven modeling 237

R

rationale 330
reception 306
reference card 55
reference property 300
refine 326
reliability requirement 223
requirement 32, 66, 123, 141, 175, 222, 247, 268
requirements diagram 62, 65, 66, 67, 115, 124, 167, 169, 177, 324
requirements engineer 100, 103, 257
requirements management tool 32

resistance 239
result 36, 70
return message 313
Revision Task Force see RTF
risk 31, 65, 121, 174, 177, 223
risk management 31, 65
RiskKind 229
role 2, 91
RTF 282
Rumbaugh, James 279

S

Safety and Reliability Profile 323
SAMS method 26, 112, 113, 274
satisfy 217, 327
scalability 109
scenario 48, 85, 149, 201
SEBoK 80, 81, 83
selector 311
semantics 283
send signal action 294, 306
sensor 213
seq 314
sequence diagram 86, 150, 202, 309
severity 224
signal 130, 306
signal event 317
SimLib 233
simulation 233, 262
six sigma 276
skills map 91
SMAP 8, 243
SME 2, 17, 55, 94, 247
SME Administrator 94

software 217
specification 262
stability 223
StabilityKind 229
stakeholder 29, 64, 66, 114, 120, 167, 168, 223, 246
stakeholder requirement 32
StakeholderCategoryKind 229
StandardProfile 323
state 49, 316
state machine 50, 73, 152, 315
state machine diagram 73, 87, 152, 204, 314
stereotype 209, 252, 321
storyboard 26, 274
Storyboard Activity see SAMS method
Modeling for Systems method
strict 314
strong coupling 271
subsystem 215
supportability 223
requirement
synchronization 297
synchronous message 312
SysML 18, 251, 279, 283
SysML diagrams 283
SysML tool 56
SYSMOD 15, 17
SYSMOD Analysis 10
Process
SYSMOD Architecture 12
Process
SYSMOD Infrastructure 9
Process
SYSMOD Methodology See SMAP
Adoption Process

system 215
system actor 34, 126, 130
system architect 101, 103, 257
system architecture 34, 48, 78, 81, 84, 270
system boundary 33
system breakdown 198
system context 33, 67, 125, 180, 192, 215
system context diagram 127
system idea 25, 59, 112, 164, 215, 245, 272
system interface 34
system objective 25, 60, 113, 166, 223, 245, 272
system process 37, 72, 132, 186, 227
system requirement 32
system state 86, 151, 203
system tester 103, 104
system use case 35, 37, 39, 69, 129, 133, 183, 227
systems engineer 100, 102, 103, 104
Systems Engineering Body of Knowledge see SEBoK
Systems Modeling Environment see SME
Systems Modeling Language see SysML
Systems Modeling Toolbox see SYSMOD

T

table 62, 65, 66, 67, 72, 120, 122, 123, 167, 169, 175, 177, 188, 195
tailoring 16
technical concepts 45
technical decision 27

technical principle 45
termination 319
test architecture 51, 78, 88, 154, 205
test case 42, 77, 88, 140, 194, 225, 326
time event 317
tool 2
Toyota Production System see TPS
TPS 276
trace 325
traceability 261, 325
trade study 265
training 20, 57
transition 49, 87, 317
trigger 36, 70, 317

U

UML 279, 283
UML 3.0 281
UML diagrams 280
UML Testing Profile see UTP
UML4SysML 323
Unified Modeling Language see UML
unit 76
usability requirement 176, 223
usage 331
use case 227, 290
use case activity 39, 74, 134, 150, 184, 189, 258
use case diagram 72, 73, 130, 133, 184, 187, 195, 289
user 213
user interface 215
user system 214

UTP 51, 323

V

validation 42, 141, 250
Value Proposition Design 26
value type 76, 299
VAMOS 51, 228, 266
variant 228
variant configuration 228
variant modeling 51, 265
variation 228
variation point 228
verdict 77, 326
VerdictKind 326
verification 42, 141, 250
verify 219, 326

W

waterfall 7
Weilkiens, Tim 11

X

X4Planet 6

Z

zigzag pattern 219, 268