The Webel Parsing Analysis recipe for text-driven Model-Based Systems Engineering (MBSE) with Systems Modeling Language (SysML) v1.6

Author: Dr Darren R. C. Kelly, Webel IT Australia, https://www.webel.com.au

Version date: 2021-06-11

Abstract

The Webel Parsing Analysis recipe for SysML is a technique for traceable elicitation of SysML model elements from text extracts quoted from domain source documents. In the SysMLv1.6 version, each text-extract is encapsulated in a Snippet extension of the ElementGroup stereotype. A Snippet must have a unique domain source Document with a known URI.

The recipe has been applied to substantial real-world applications such as modelling of radio telescopes, particle accelerators, neutron beam instruments, electronics, robotics, astronomical instrumentation, green rating schemes for buildings, and for digital twin encapsulation of building and construction projects.

The Webel Parsing Analysis recipe is a meta-process that can be combined with other Model-Based Systems Engineering methodologies and help amplify them.

The recipe is particularly useful in combination with requirements engineering; a modelling layer of elicited model elements that includes elicited SysML Requirements is created before direct treatment of the Requirements. Text statements of requirements can also be used in Snippets for elicitation of general SysML model elements, including those that must Satisfy – or are otherwise involved in – a Requirement.

The recipe has benefits for many aspects of the modelling process; because the modelling is driven by small text extracts, Parsing Analysis Diagrams have a natural scope. Elicited model elements are then integrated progressively step-wise into robust SysML models that are guaranteed to converge as long as the chosen domain source documents are coherent. The technique also offers a highly effective way of identifying and tracking inconsistencies in source domain documents.

The author suggests that the technique offers a powerful cognitive advantage over purely graphically based modelling that is not related to source texts; the human brain understands more when text is presented alongside graphical symbols, which approach also helps make SysML models comprehensible to more stakeholders.

The modelling recipe as for SysMLv1.6 and as tuned for the MagicDraw SysML and Cameo Systems Modeler (CATIA Magic) family of SysML tools is presented, with a word on adapting the recipe for future SysMLv2.

Table of Contents

Table of Figures

Acronyms

WPA = Webel Parsing Analysis

PAD = Parsing Analysis Diagram

PD = Presentation Diagram

SIZ = Source Input Zone

TMZ = Target Model Zone

MBSE = Model-Based Systems Engineering

OMG = Object Management Group

UML = Unified Modeling Language (unless otherwise stated refers to UML2.5.1)

OCL = Object Constraint Language

SysML = Systems Modeling Language (unless otherwise stated refers to SysML1.6)

SysMLv1.6 = Version 1.6 of the Systems Modeling Language

SysMLv1.x = The 1.x family of Systems Modeling Language specifications

BDD = SysMLv1.x Block Definition Diagram

IBD = SysMLv1.x Internal Block Diagram

SysMLv2 = Version 2 of the SysML language (under development by OMG partners)

DSL = Domain Specific Language

URI = Uniform Resource Identifier

URL = Uniform Resource Locator

URN = Uniform Resource Name

ISBN = International Standard Book Number

RDF = Resource Description Framework

RDF/S = Resource Description Framework Schema

OWL = Web Ontology Language

ODM = Ontology Definition Metamodel

Copyright in example text extracts

Conventions

- The tools formerly known via former vendor No Magic Inc as the MagicDraw® SysML Plugin for MagicDraw® UML, the MagicDraw Requirements Plugin, and the Cameo Systems Modeler® product bundle – now offered by parent vendor Dassault Systèmes® as the CATIA Magic family of products – are indicated throughout simply as 'MagicDraw', whereby it is understood this indicates a tool for SysML1.x versions of the Systems Modeling Language (SysML), building upon Unified Modelling Language (UML).

- Where a UML Metaclass or SysML Stereotype is referenced it is written in PascalCase (a.k.a. UpperCamelCase) where appropriate. For example: *ValueType* refers specifically to a SysML profile Stereotype, whereas *value type* refers to the concept; *Block* refers specifically to a SysML profile stereotype, whereas *block* refers to the concept.

- Properties and attributes of Stereotypes or Blocks referenced without their owner are indicated in single quotes. Example: '/member' refers to a derived property of the ElementGroup stereotype. However, when indicated qualified by the owner ElementGroup::/member the single quotes are omitted.

- In some cases, additional tool-specific Stereotypes are also indicated. For example, there is no stereotype for a value property in SysML1.6, but the MagicDraw tool offers a ValueProperty, indicated here as MD:ValueProperty.

- The per-Element values of Properties (attributes) of applied Stereotypes are referred to as *tagged values*.

Introduction

Text documents – frequently also including figures containing graphics, plots, drawings etc. – remain the primary mode of communication and knowledge sharing between project managers, engineers, scientists, software engineers, architects, and other stakeholders on many systems engineering projects. Typically, such domain documents are primarily written in a natural language style that is easily written by authors and easily read by most peers. By contrast, understanding the graphical notations of the Systems Modeling Language (SysML) of the Object Management Group (OMG) requires specific knowledge of, and experience with, the SysML language.

The simple act of including a Comment symbol on a diagram along with SysML model element symbols that correspond to the text of the Comment can greatly enhance the development of robust, accurate SysML models and communication of the model diagrams to a wider stakeholder audience. Text from domain source documents can also be used to help elicit various types of SysML model elements, suggest the relationships between them, and provide quantity data and metadata for model elements, where modelling metadata may be displayed on element symbols as so-called *tagged values* of applied custom Stereotypes.

The Webel Parsing Analysis (WPA) recipe for Model-Based Systems Engineering (MBSE) builds on these basic principles to provide a complete strategy for mapping text extracts quoted from domain source documents to SysML models with traceable elicitation of model elements.

The SysMLv1.6 variant of the WPA recipe employs a customised extension of the ElementGroup called a Snippet (identified by the «snippet» stereotype keyword), which must always have a single domain document that is identified by a URI as a ‘source’, where the URI may be a URL or a URN (such as an ISBN or other unique identifier). Snippets leverage the ‘/member’ tracking facility of the SysMLv1.6 ElementGroup to track elicitation of model elements vs domain ‘source’ documents.

Text extracts flow in; elicited model elements flow out

A text-driven WPA project has two major SysML modelling zones, reflecting the domain source document text processing workflow:

The *Source Input Zone and Parsing Analysis Diagrams (PADs)*

The *Source Input Zone* references external domain sources as Documents and contains *Parsing Analysis Diagrams (PADs)* – identified by the «pa» stereotype keyword – which are used to traceably elicit model elements from the referenced Documents via Snippets containing quoted text extracts.

PADs are “scratchpad” diagrams only, they are not intended as final presentation diagrams. They serve their purpose only once, are typically only ever viewed by a single modeller, they may change after initial use (due to later changes and additions made elsewhere in the underlying model), and they need not even be tidy or presentable, since they are not intended for viewing by anyone but a modeller applying the WPA recipe.

A *focus PAD* always shows at least one Snippet element, including clear display of the element name of its ‘source’ domain Document, and listing of any elicited ‘/member’. An *index PAD* shows an overview of a domain source Document and some related Snippets; it must at least show the ‘source’ of each Snippet (but need not list their ‘/member’).

Any suitable SysML diagram type may be used as a PAD (except those kinds that can’t be consistently owned under the Source Input Zone) and a single Snippet may appear in more than one PAD (and thus in more than one diagram type).

All Packages or Model packages within the Source Input Zone must also have the «pa» stereotype keyword applied to indicate their role in the WPA workflow.

Any new model elements elicited in a PAD must ultimately always be moved outside the Source Input Zone, i.e., every model element (except for PADs themselves and any supporting elements) must ultimately not have a Package, Model package, or other container Element from the Source Input Zone as owner.

The *Target Model Zone and Presentation Diagrams (PDs)*

The *Target Model Zone* includes everything outside the Source Input Zone (except for general model libraries and profiles required by the SysML language itself or the tool).

Presentation Diagrams (PDs) here may choose to also show Snippet symbols containing text extracts relevant to the description or explanation of any of the included model

element symbols. In this case, the ‘source’ of a Snippet must be shown, but the ‘/member’ list is typically not shown (and no other *tagged values* from the extended ElementGroup are shown). Presentation Diagrams are the primary mode of communication to all stakeholders, including those who are not SysML-fluent. The inclusion of such simplified Snippets containing relevant text alongside graphical SysML model elements symbols is at the heart of the powerful cognitive reinforcement afforded by the WPA recipe.

In the purest form of a WPA SysML model, every element in the Target Model Zone has either been directly elicited via one or more Snippets, or has been declared by the modeller to have been *implied* by a Snippet (as indicated by the «pa:implied» keyword), or *assumed* to exist without support of a Snippet (identified by the «pa:assumed» keyword). The modeller may choose to employ additional notations or naming conventions to indicate model elements that were not elicited from a Snippet.

In the case where existing domain source documents offer substantial coverage of description of all aspects of a system, the modeller aims to increase the number of traceably elicited model elements and minimise the number of «pa:implied» and «pa:assumed» elements. Model elements can be upgraded from «pa:assumed» and «pa:implied» to become fully elicited elements if indeed clearly referenced by the text extracts of one or more later Snippets.

Webel Parsing Analysis as a meta-process

The WPA recipe represents a meta-process that feeds into, and can be combined with, any other systems engineering modelling methodology. In particular, WPA precedes application of any requirements engineering methodology within the SysML model (although some formal external requirements elicitation may occur before SysML modelling commences). Under WPA, a SysML Requirement is merely another element type that falls out of the meta-process. A Requirement element may be elicited from any text extract from any domain source document; it may be elicited from an explicitly stated requirement or from a requirement or constraint that is implied by some natural language text (and not necessarily from a domain source document that is even declared to be a “requirements document”).

Indeed, text from a domain source document that does declare itself to be a “requirements document” (or text from a requirements database) can be used as the quoted text extract of a Snippet. Thus, in parallel with elicitation of a SysML Requirement element, other SysML model elements relevant to that Requirement (such as Blocks and other model elements that may be claimed to Satisfy a Requirement or are otherwise related to the Requirement) are also elicited. This approach helps exploit one of the most powerful advantages of fully integrated SysML-based requirements engineering over more traditional text-based requirements engineering.

Webel Parsing Analysis vs freestyle SysML modelling

The WPA recipe does not preclude the creative use of a SysML modelling tool. While it is mandated that any model element that appears in a PAD must be a member of Snippet (somewhere), or must be indicated as «pa:implied» and «pa:assumed», it is not mandated that every single model element in the Target Model Zone must be traceably elicited from text. As a general principle, the WPA recipe is applied as strictly as possible at early stages of a project to robustly and traceably populate an initial SysML model from authoritative, relatively consistent, existing domain source documents. At later stages of a project, modellers may choose to introduce many new model elements without any traceable

reference to domain source document text; this should however be performed in a way that does not undermine the elicited model elements. (It will be shown later how prefix characters can, for example, be used in model element names to help distinguish elicited model elements from introduced model elements).

In cases where a modeller wishes to elicit model elements from figures or tables of a domain source document (rather than from quoted text) the SysML Trace relationship may be used, as shown briefly later.

The term *Parsing Analysis* may be evocative of painfully tedious grammar exercises prone to “analysis paralysis”, with a focus on pedantic syntax parsing sufficient to admit application of machine learning linguistic methods.

WPA is, by contrast, a practical recipe is intended for pragmatic use by human modellers, with a simple emphasis on mapping the interpreted meaning of text sentences into corresponding SysML models, and indeed the process of applying WPA can help a modeller better interpret and understand the intended meaning of the text from the original authors of a domain source document. It promotes fluent, enjoyable SysML modelling.

Basics of the Snippet extension of the ElementGroup

In the SysML1.6 version of the WPA recipe for SysML, a Snippet is a customised extension of the SysML ElementGroup stereotype. It introduces at least a mandatory ‘source[1]’ property for tracking of a single domain source document from which any ‘body’ text extract it displays must be quoted. The Comment-like symbol for a Snippet should always display its unique ‘source’ as a so-called *tagged value*.

As tuned for use with MagicDraw, a Snippet employs MagicDraw Customizations to treat a Snippet as a standalone custom element type with display of most of the properties (as tagged values) of the ElementGroup hidden by default, except for ‘/member’ and the text ‘body’, where ‘/member[0..*]’ is derived from the ‘annotatedElement[0..*]’ of the UML2.5.1 Comment. Display of the list of ‘/member’ may be optionally hidden in Presentation Diagrams and index PADS but must always be shown in focus PADs.

An ‘annotatedElement’ for a Comment (or ‘/member’ for an ElementGroup) may be collected in a tool in diagrams by drawing a dashed line known informally as an *anchor* from a Comment symbol to any Element symbol (for those Elements that have symbols). An anchor is not a first-class Relationship, however the MagicDraw/Cameo tool enables one to nevertheless navigate from an ‘annotatedElement’ to its annotating Comment and vice-versa. In the case of the SysML ElementGroup (and thus also the WPA Snippet extension) it likewise enables navigation from a ‘/member’ to its annotating ElementGroup.

An Element also may be collected as an ‘annotatedElement’ via the specification dialog. This is useful, for example, for collected States from StateMachine diagrams that can’t be owned within the Source Input Zone, and for Elements that do not have diagram symbols.

The tool also offers additional ‘/member’ display features in the model containment tree.

In the context of a Snippet, every ‘/member’ is interpreted as having been either explicitly elicited from its quoted domain source text (held in the ‘body’) or at least «pa:implied» by it. The grouping of ‘/member’ by a Snippet is purely logical; there is no implied ownership (This principle will also be carried over to a future SysMLv2 version of the WPA recipe.)

The domain source Document

Every Snippet has a mandatory single ‘source’ Document with either a URL or a URN acting as a URI. A URN may be an ISBN or other suitable identifier.

In the WPA recipe as tuned for the MagicDraw/Cameo tool, a domain source document is tracked using a custom stereotype Document (with keyword «document») that extends the Document of the UML Standard Profile, which is in turn an extension of the UML Artifact metaclass. The UML4SysML intersection of SysMLv1.6 does not include the UML Artifact, but in practice this makes absolutely no difference in the tool, and the decision to use an Artifact-based extension here is a deliberate provocation; there is a need in SysML for an element that clearly indicates documents (and hopefully one will be supported in SysMLv2). One could, however, just as easily extend a Block or any other SysML-friendly element. A Stereotype icon can be used to reinforce its use for domain source documents.

In the case where a domain source document has a unique URL, a WPA Document element may additionally have that URL assigned as an interactive hyperlink in the MagicDraw tool, so that the external URL loads in a web browser (or other client) on clicking the symbol for the WPA Document in diagrams in the tool.

A WPA Document may optionally also have `author:Person` and/or `org:Organisation` items assigned as tagged values, where the Person and Organisation stereotypes extend the Actor metaclass. Additional properties may be included to indicate, for example, Copyright.

Figure 1 shows part of a typical WPA profile for Snippet, Document, Person, and Organisation. This is provided as a starting point only; users of the WPA recipe may freely adapt the Document, Person, and Organisation stereotypes to include any desired tagged values metadata. The MagicDraw Customizations employed to specify the custom stereotype keywords for these *Domain Specific Language (DSL)* extensions and control the default visibility of tagged values are shown in Figure 23 in **Appendix A** (p.34).

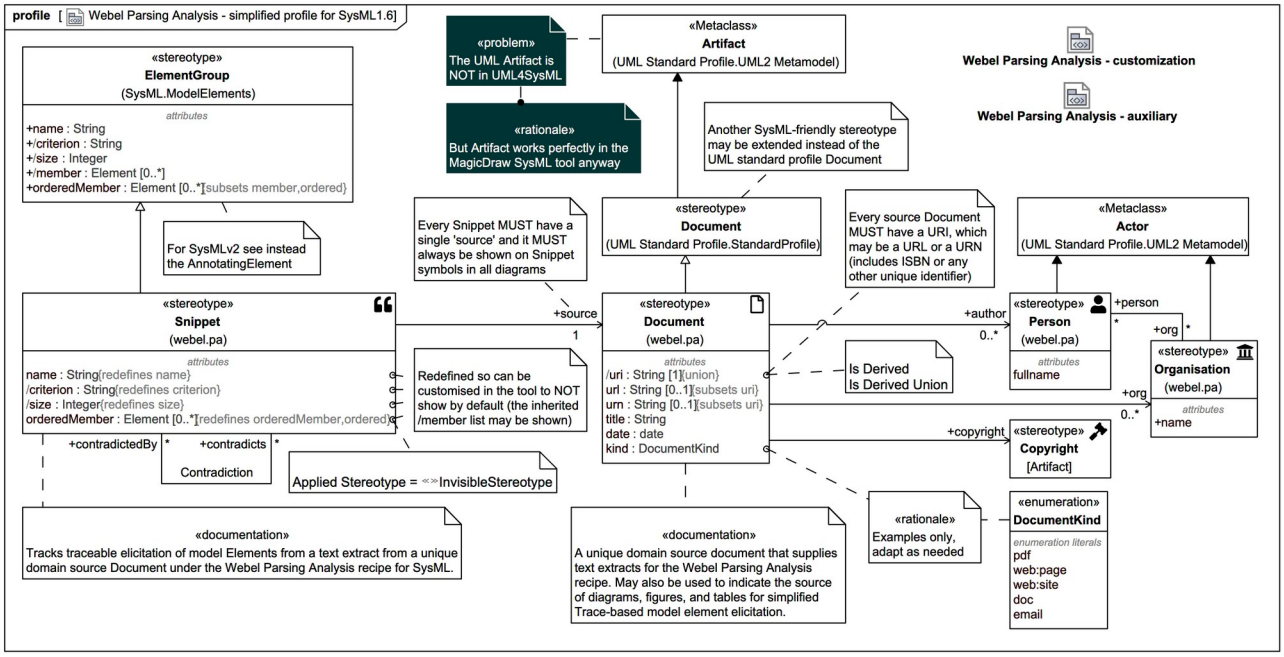

Figure 1: Simplified profile for Webel Parsing Analysis for SysML1.6+ in MagicDraw

Document and Snippets for an English pangram

The Wikipedia tells us:

> '"The quick brown fox jumps over the lazy dog" is an English-language pangram – a sentence that contains all of the letters of the English alphabet.'

The 'source' URL for this quoted text extract is:

https://en.wikipedia.org/wiki/The_quick_brown_fox_jumps_over_the_lazy_dog

The Document representing the external source, any Snippets capturing the text extract, and the PADs used to process them and elicit model element must all live under a top level Package or Model package that defines the Source Input Zone. By convention this top-level is named `0-source` (so that it lists above other modelling zones in the containment tree of the tool), as shown in Figure 2.

Below `0-source` is a Model `wikipedia` for grouping treatment of one or more domain source Documents. In this case there is only one Document `FoxDogPangram`, which is managed by a dedicated Model of the same name for containing related Snippets and the PADs used to elicit model elements from them. There is also in this case an Organisation `Wikipedia`. The Model packages and PADs all have the «pa» keyword applied.

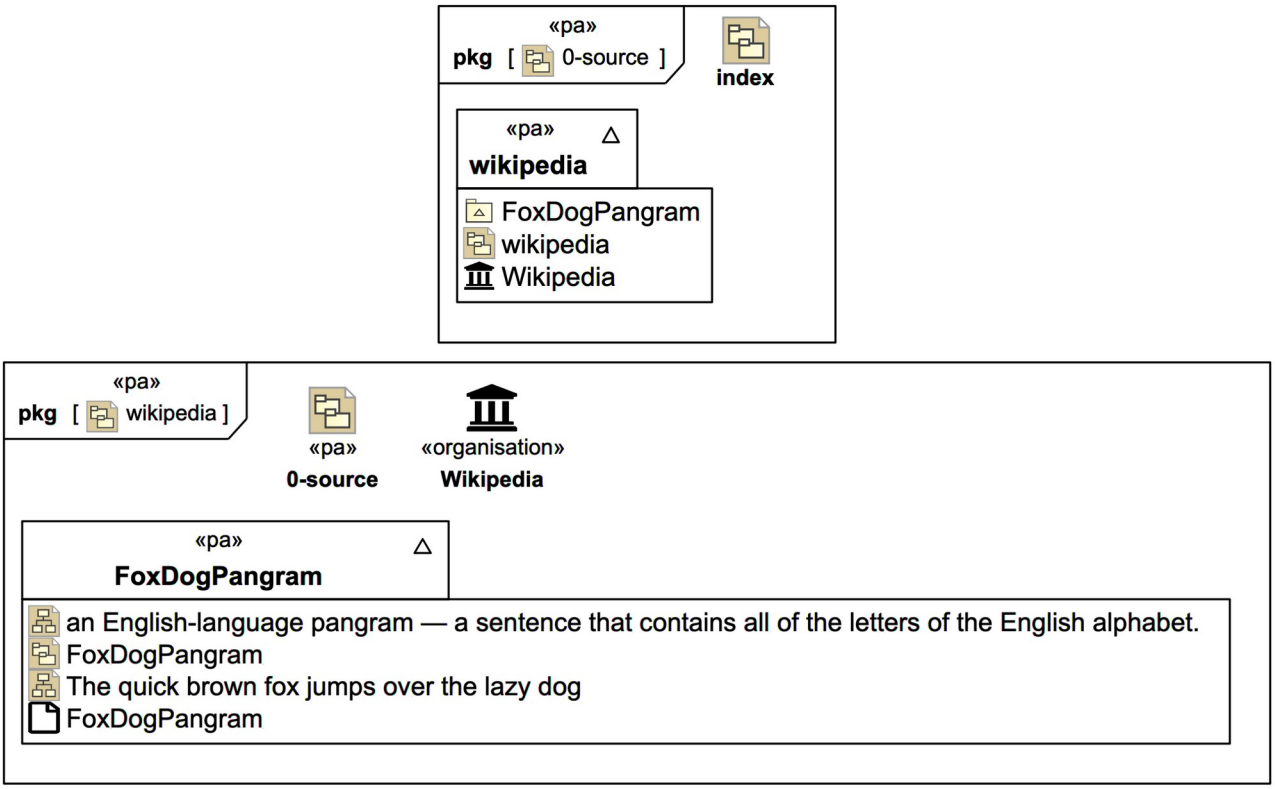

Figure 2: Top-level Package/Model organisation for the Source Input Zone

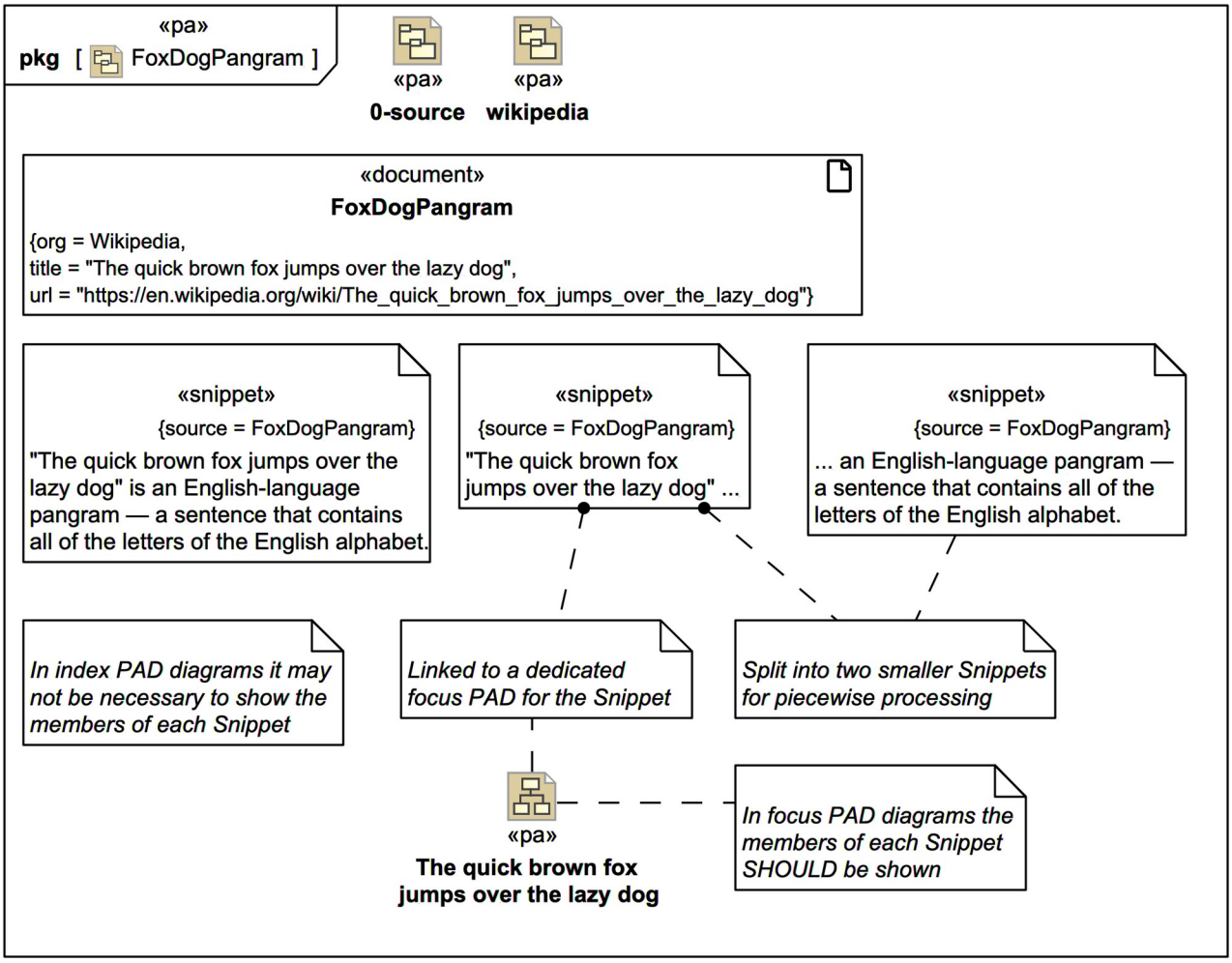

Figure 3: Index PAD with overview of Snippets for the FoxDogPangram source Document

It is usually convenient to have an *index PAD* for each source Document, as shown in Figure 3. (Where many Snippets are processed from one Document one may use an index PAD for each chapter or section.) The «document» `FoxDogPangram` has a 'title', a 'url', and an «organisation» `Wikipedia` set as its 'org'. (In some cases Person items might be set as 'author'.) The name of the Document element need not be as verbose as the title, noting that Snippet symbols may also be used in final Presentation Diagrams, but must always display the 'source' by name.

Each Snippet that is processed is linked to at least one *focus PAD* where model elements are elicited. An index PAD need not always show the '/member' tagged value on every Snippet symbol, but a focus PAD should always display the elicited '/member' list.

For easier processing, the text extract is handled as two separate Snippets with partial text, commencing with:

> '"The quick brown fox jumps over the lazy dog" … '

Note the indication '…' of ellipsis.

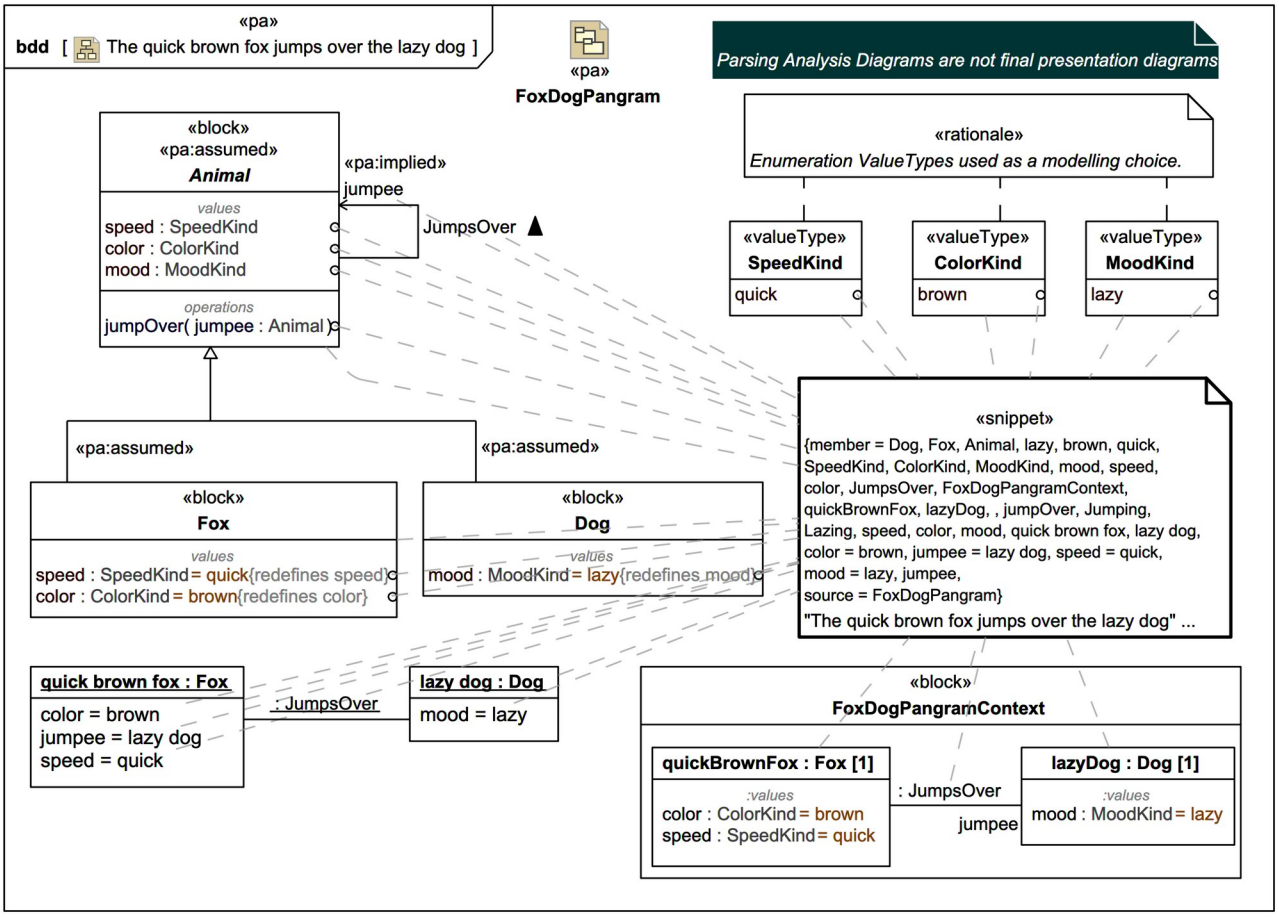

Figure 4: Focus PAD for elicitation of model elements from a Snippet

The focus PAD, for which a SysML BDD was used, is given in Figure 4. Note how the '/member' list and 'source' are both displayed as tagged values on the «snippet» symbol.

Even though such PAD diagrams are only scratchpads for eliciting model elements, it can help in the tool to use a light grey colour style on the dashed line anchors/handles used to collect model elements to improve readability.

At least the following SysML Blocks can be immediately elicited from the nouns (using here a convention that the first letter of the name of a Block must be capital):

- `Fox`, `Dog`

There are some attributes indicated by adjectives, which in SysML may typically be modelled (amongst other modelling choices) by value properties. It seems that a `Fox` may be 'quick' (it will be assumed that this is taken from a «valueType» Enumeration `SpeedKind`), a `Fox` may be 'brown' (from a «valueType» Enumeration `ColorKind`) and a `Dog` may be 'lazy' (from a «valueType» Enumeration `MoodKind`):

We are not explicitly told what a 'dog' or 'fox' are, but the modeller may use prior knowledge about the world and the context to conclude that they are types of animals, which is indicated using the «pa:assumed» keyword on a more general block `Animal` «pa:assumed» and thoroughly also on Generalizations from `Fox` and `Dog` to `Animal`. (If the application of «pa:assumed» to such Generalizations leads to diagram clutter the stereotypes keywords need not always be shown.)

It is not always clear whether «pa:assumed» or «pa:implied» should be used in such cases, the main thing is that the modeller applies at least one of those stereotypes to indicate that an additional modelling interpretation was required.

The Snippet also yields some general value properties (assumed to apply to any animal):

- `Animal::speed:SpeedKind`

- `Animal::color:ColorKind`

- `Animal::mood:MoodKind`

And more specific redefined value properties with default values matching the narrative:

- `Fox::speed:SpeedKind = quick {redefines speed}`

- `Fox::color:ColorKind = brown {redefines color}`

- `Dog::mood:MoodKind = lazy {redefines mood}`

There are also Behaviors and Operations. It would seem that a `Fox` can jump. Can any `Animal` jump? Surely foxes don't only jump over dogs. For now a more general operation is introduced, where a «pa:implied» 'jumpee' is any `Animal` that another `Animal` jumps over:

- `Animal::jumpOver(jumpee:Animal)`

There are InstanceSpecifications classified by `Fox` and `Dog` (with Slot values corresponding to the described scenario):

- `quick brown fox:Fox`

- `lazy dog:Dog`

An Association named `JumpsOver` is used to classify an anonymous link `:jumpsOver` between the instances. An anonymous Element may be collected as a '/member' of a Snippet (it is not important whether collected elements list with a clear name under '/member', only that they are traceably elicited).

In a context block there are corresponding part properties (using a slightly different naming convention) with an anonymous Connector typed by the same Association `JumpsOver`:

- `FoxDogPangramContext::quickBrownFox:Fox`

- `FoxDogPangramContext::lazyDog:Dog`

Two States for a StateMachine within the abstract block Animal have also been elicited:

- `Jumping`, `Lazing`

As StateMachine Diagrams are always owned by an element that should not eventually remain under the Source Input Zone they should not be used as PADs; States may be

elicited instead by adding them via the '/member' section of the ElementGroup specification dialog in the MagicDraw tool.

Figure 5 shows part of the containment tree in the MagicDraw tool showing the logical members within a «snippet» ElementGroup. Note that although this appears in the containment tree it does not imply ownership, it is a logical grouping of '/member' only.

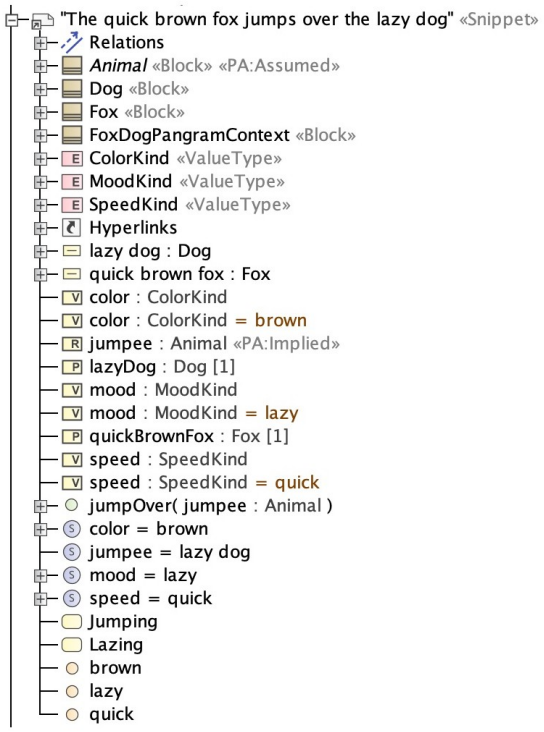

Figure 5: Part of the containment tree of MagicDraw showing the logical grouping of members within a Snippet ElementGroup

The newly elicited elements are then moved out of the Source Input Zone into Packages or Models in the Target Model Zone, the main modelling area for the project, and can be used in Presentation Diagrams for final consumption by general stakeholders. For example, in a focus BDD Figure 6 featuring the FoxDogPangramContext block and some related elements are shown (with owner indicators displayed). A relevant Snippet is also included, but with only the 'source' (not the '/member' list) showing, and only a few dashed line anchors to some elicited '/member' blocks.

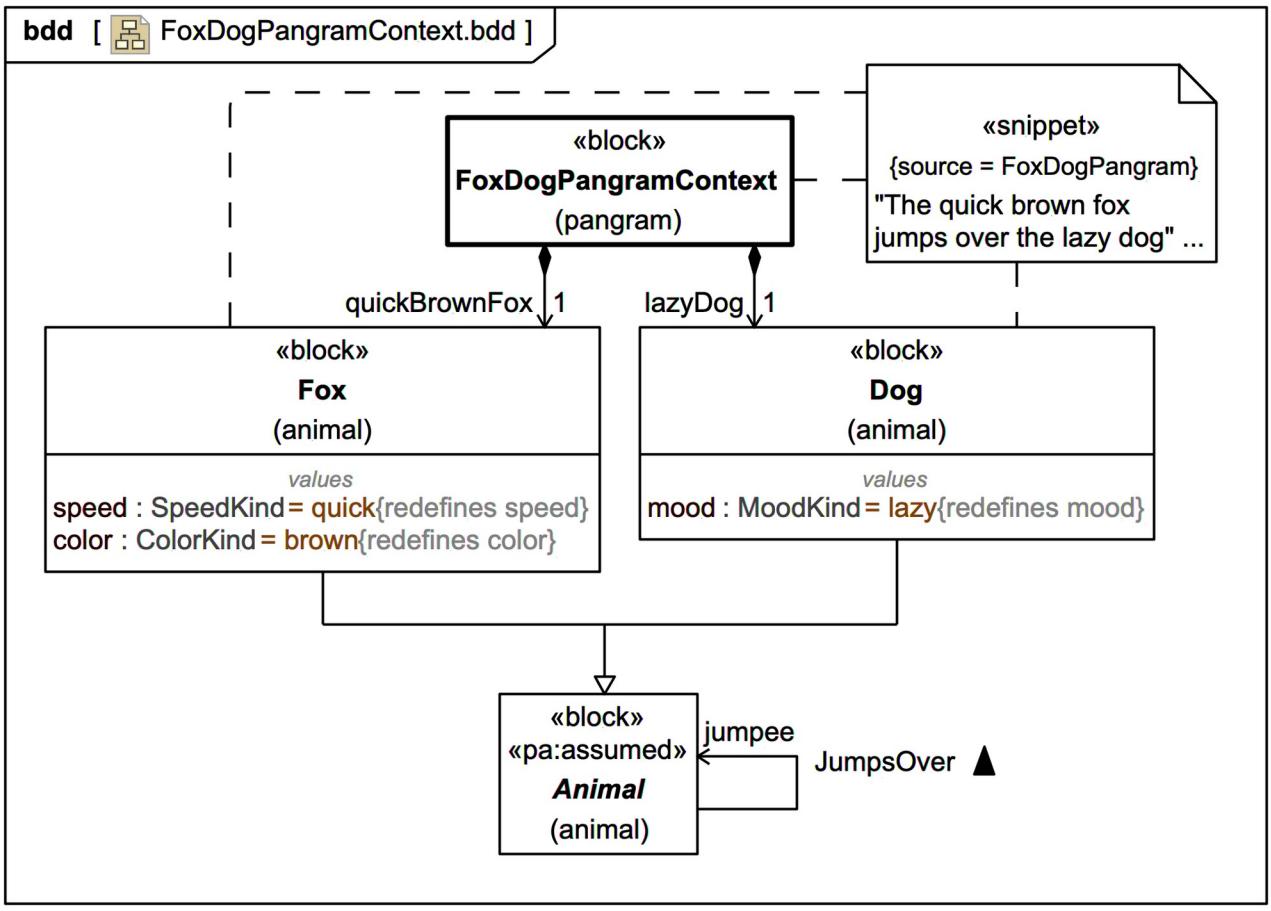

Figure 6 Focus BDD for some elicited model elements that have been moved out of the Source Input Zone into the main Target Model Zone.

Consider now the language-related aspects referred to in the partial Snippet:

> '... an English-language pangram – a sentence that contains all of the letters of the English alphabet.'

A focus PAD for the Snippet is given in Figure 7.

At least the following Blocks can be immediately elicited from nouns:

- `Pangram`, `English`, `Language`, `Sentence`, `Letter`, `Alphabet`

It's seems clear that `English` is a type of `Language`. More specifically, it is implied that `English` is a type of the more abstract `AlphabetLanguage`, where an `Alphabet` references one or more `Letter` (shared aggregation has been used here). A `Pangram` has at least as many letters as the number of unique letters in the `Alphabet` it is associated with, and has at least one of each of those letters (this is difficult to express in OCL).

While Generalizations and Associations may be collected as elicited members of a Snippet this can quickly lead to clutter in the '/member' list display, and is omitted in Figure 7.

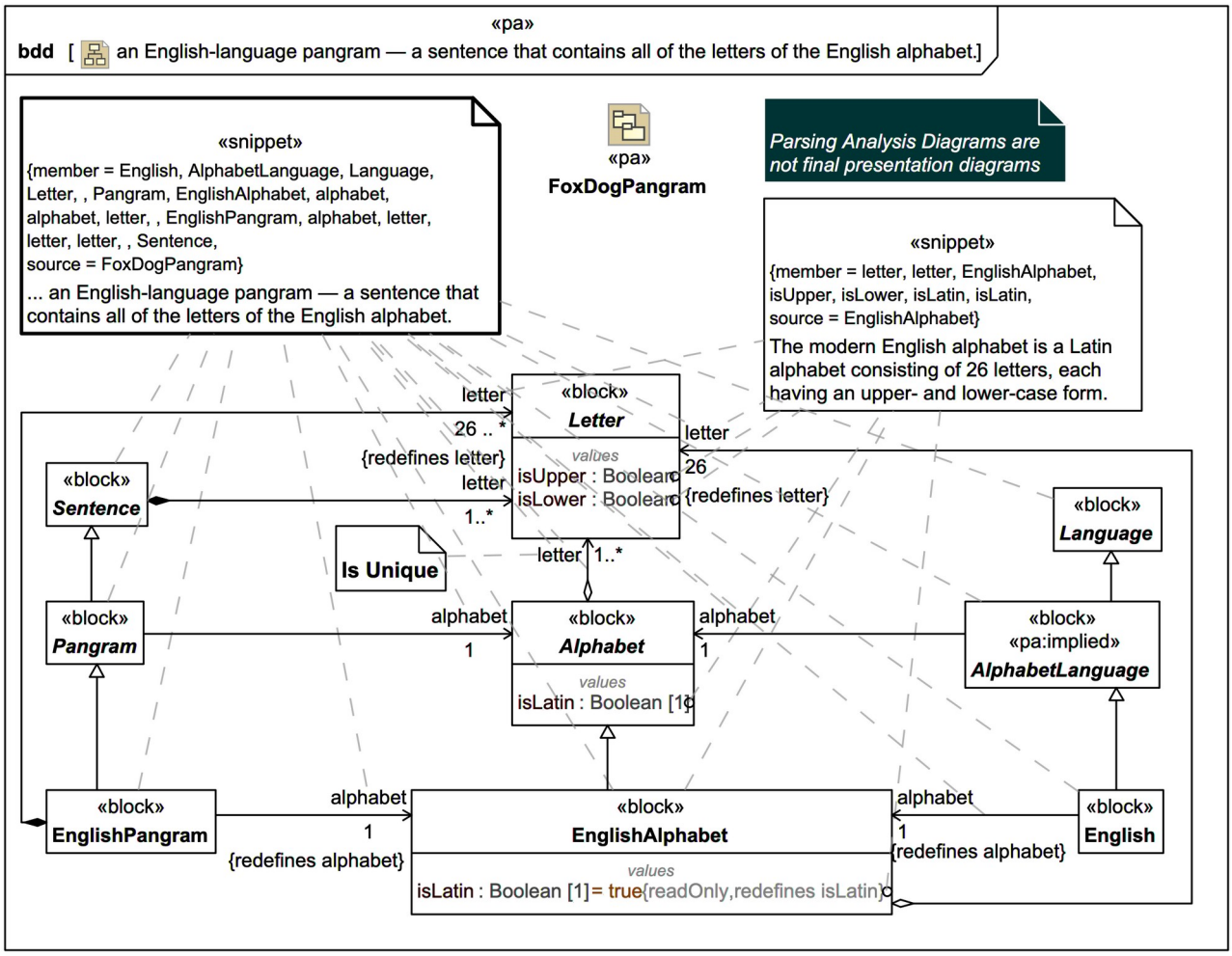

Figure 7: Focus PAD for elicitation of model elements for English-language pangrams

The analysis has also been supported by an additional Snippet from another source:

'The modern English alphabet is a Latin alphabet consisting of 26 letters, each having an upper- and lower-case form.'

https://en.wikipedia.org/wiki/English_alphabet

In addition to obtaining the multiplicity 26 for the number of `letter:Letter` in `EnglishAlphabet`, partial analysis also yields the following value properties:

- `Letter::isUpper:Boolean, Letter::isLower:Boolean`

- `Alphabet::isLatin[1]:Boolean`

- `EnglishAlphabet::isLatin[1]:Boolean=true{readonly,redefines isLatin}`

A Constraint for `isUpper` vs `isLower` could also eventually be included.

The newly elicited elements have once again been moved out of the Source Input Zone into Packages or Models in the Target Model Zone, the main modelling area for the project, and can be used in Presentation Diagrams for final consumption by general

stakeholders. In Figure 8 a focus BDD featuring the `EnglishPangramContext` block and some related elements are shown (with owner indicators displayed). A relevant Snippet is also included, but with only the 'source' (not the '/member' list) showing, and only a couple of dashed line anchors to some elicited '/member' blocks.

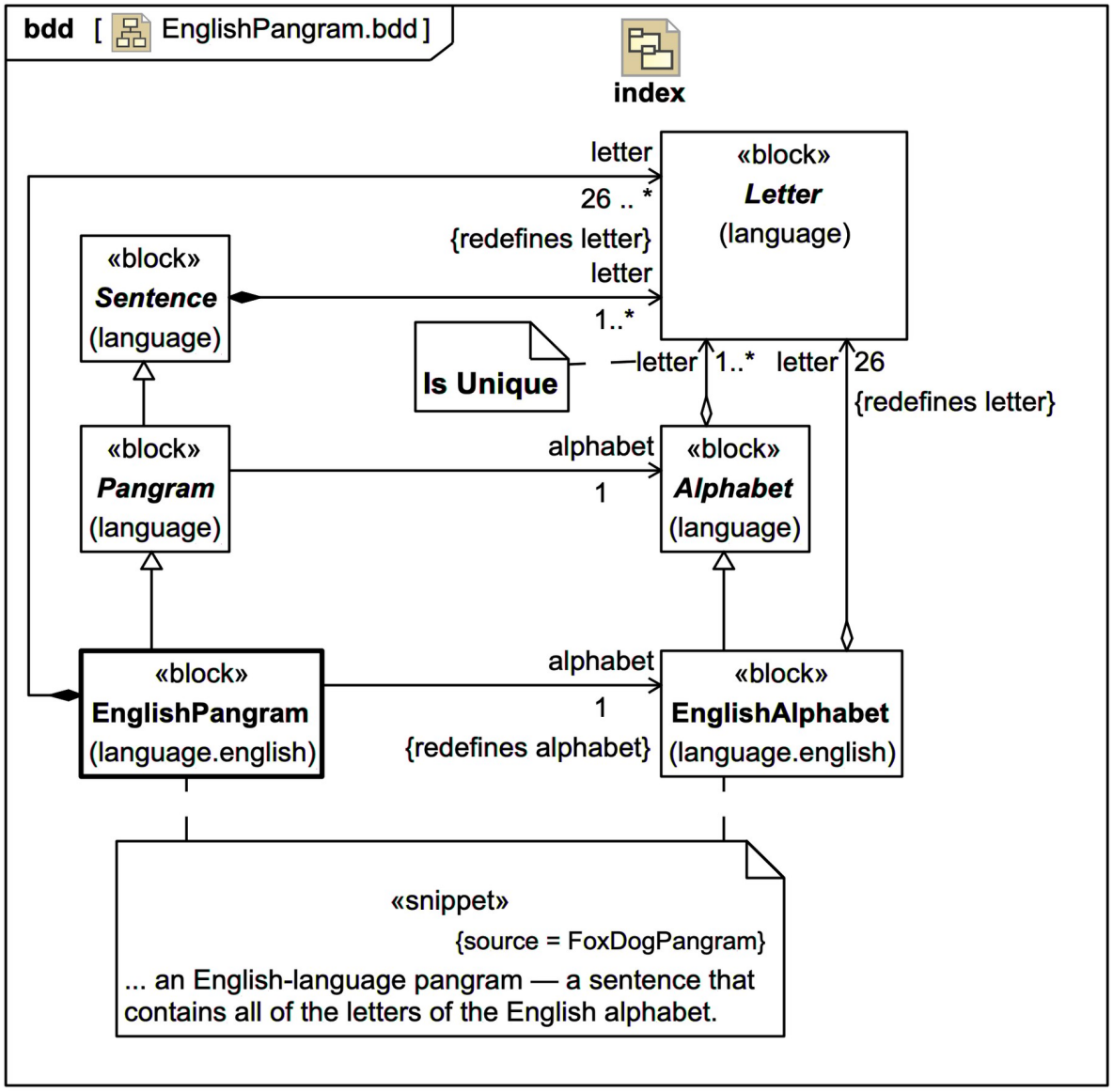

Figure 8: A focus BDD for the EnglishPangram block with a relevant Snippet

The `FoxDogPangram` is specific to the `EnglishAlphabet` and the quoted text extract for the pangram itself can be used to elicit each unique Letter of the `EnglishAlphabet` (as done here using a specialising Block for each `Letter`) as shown in Figure 9.The block `FoxDogPangram` defines also 35 letter part properties as {subsets} of its `Pangram` letters.

There are lots of dashed-line "anchors" from the Snippet symbol to symbols of elicited members in the PAD in Figure 9. In such cases, once anchor lines have been used to collect members, they may be selectively omitted from display in the diagram to reduce clutter.

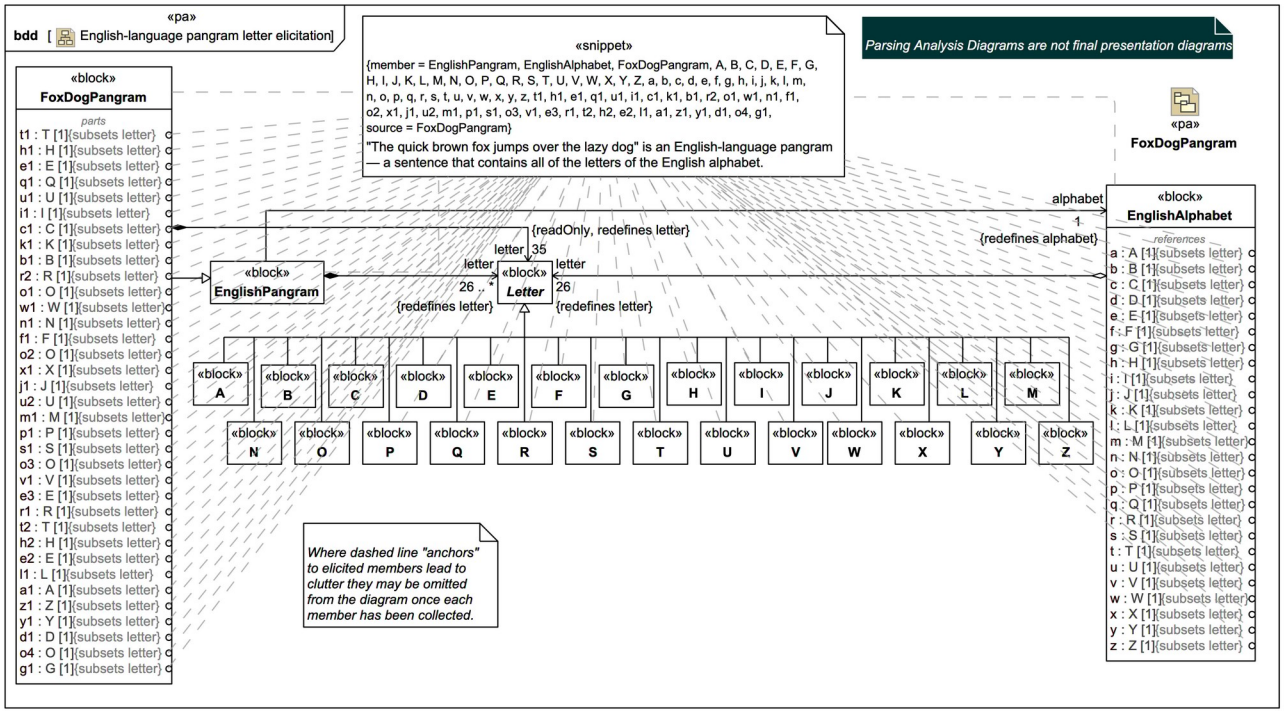

Figure 9: A PAD with a Snippet used to elicit letters of the English alphabet and a pangram

The newly elicited elements have once again been moved out of the Source Input Zone into Packages or Models under the Target Model Zone, the main modelling area for the project, and can be used in Presentation Diagrams for final consumption by general stakeholders. Figure 10 shows a focus BDD featuring the `FoxDogPangram` block and some related elements (with owner indicators displayed). A relevant Snippet is also included, but with only the 'source' (not the '/member' list) showing, and only a couple of dashed-line anchors to some elicited '/member' blocks.

Part property symbols in the structure compartment have been used in Figure 10 to indicate the pangram "a quick brown fox jumps over the lazy dog", with Connectors used to indicate the order of letters within words. (If the analysis had already indicated the existence of a `Space` block and `Period` block as punctuation characters they could have been included too.)

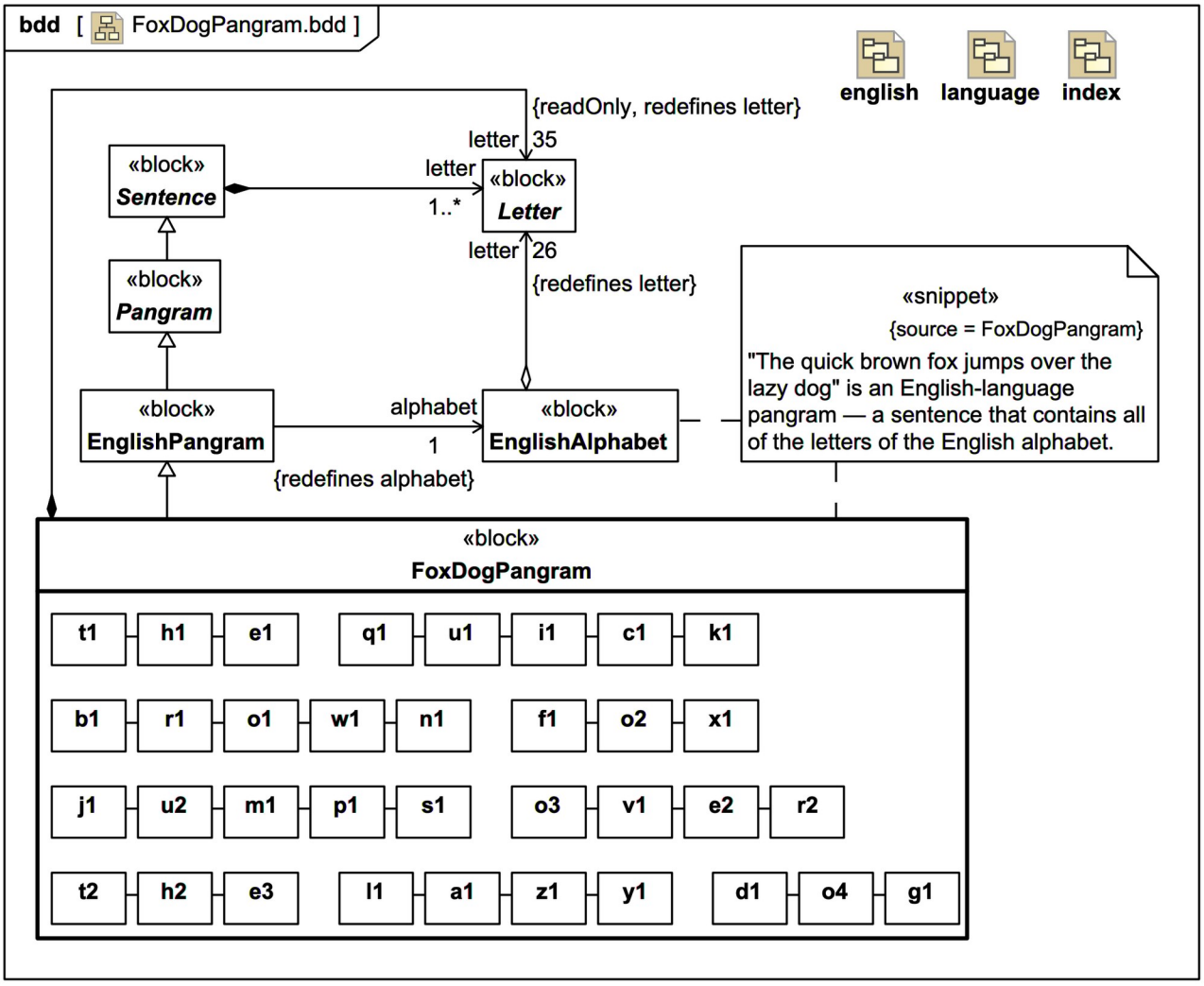

Figure 10: A focus BDD for the block FoxDogPangram including an informative Snippet

On a real-world project, a modeller must decide just how fine-grained the model element elicitation must go. For example, on an electronics project with a board with many components such fine-grained collection of elements may well be required.

Figure 11 shows a SysML Package Diagram used as an index diagram, with the «pa» `0-source` Model package defining the Source Input Zone and Model packages in the Target Model Zone listing elicited model elements. The stereotype keyword «pa:from» has been applied to Dependencies from Model packages to the «pa» `0-source` Model package to indicate that the packaged elements were elicited using the WPA recipe. The ownership flow of model elements as they are elicited is from temporary ownership within the Source Input Zone to ultimate ownership outside it within the Target Model Zone; the «pa:from» Dependencies are in the opposite direction. Use of the «pa:from» stereotype keyword is entirely optional, and is usually only employed for illustrative purposes.

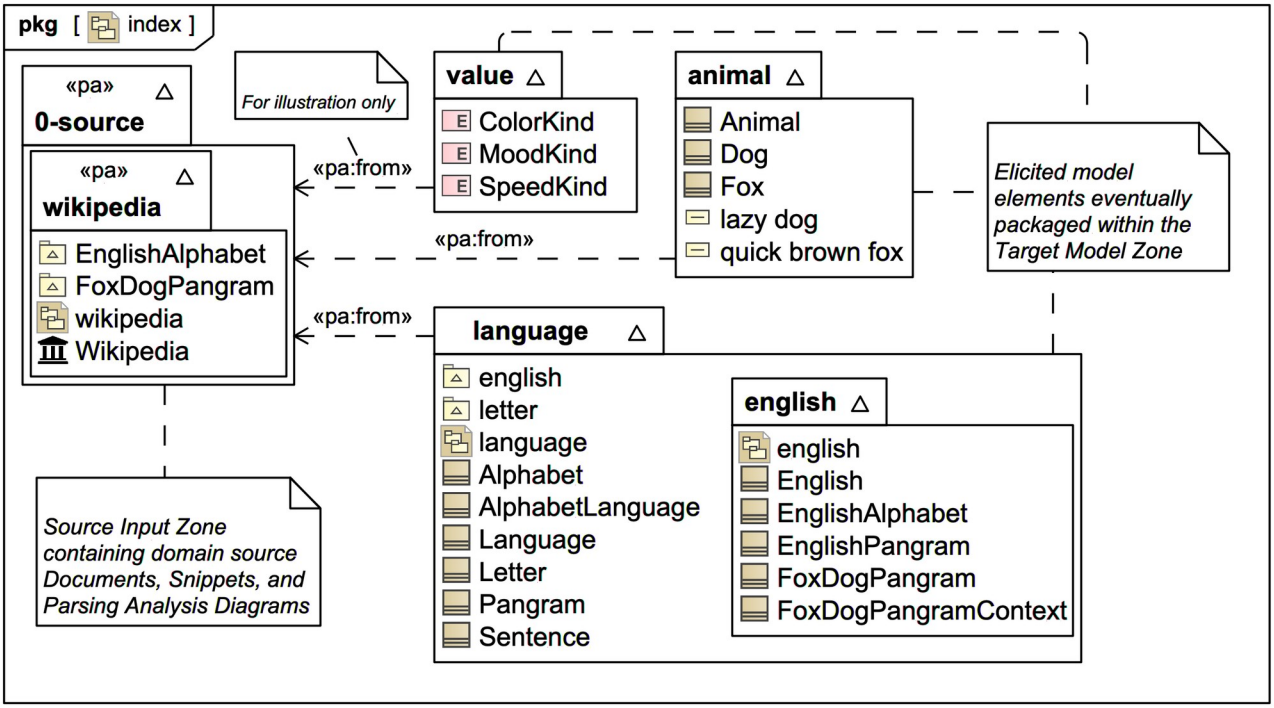

Figure 11: SysML Package Diagram as an index showing final packaging of model elements that were elicited from Snippets (use of the «pa:from» keyword is illustrative)

Handling synonyms, alternative names, and identifiers

Some entities are known by more than one name, and sometimes the primary human-friendly name does not match a desired code-friendly element naming pattern. An elicited Element may also be a member of more than one Snippet. In some cases, a single concept is referred to by different names (synonyms) in different domain source documents. An Element may also correspond to an entity known within an organisation by one or more machine-friendly identifiers.

Synonyms can be handled using, for example, a custom stereotype PA:Term (with keyword «pa:term») with tagged values such as ‘aka’, ‘realName’, and ‘id’, to carry additional names and identifiers as known to humans or within an organisation. Especially where strict machine-friendly element naming conventions are employed such overloading naming carried as per-element metadata can help communicate with stakeholders. The specific stereotype properties (attributes) may be adapted according to the needs of a modeller or their organisation.

In Figure 12 tagged values for the «pa:term» stereotype keyword have been used to capture such overloaded naming for a model of a children’s story. A code-friendly post-adjectival naming pattern has been used for the block `Giant_big_friendly`, which has tagged values `acronym = "BFG"` and `realName = "The Big Friendly Giant"`. For the sake of demonstration here it has also been given an `id = "G01"`, because it’s the first giant mentioned. It seems that a Human is (also known as) `aka = "human bean"`. The human `Sophie` has been given `id = "H01"`, since she's the first human mentioned.

On some real-world projects an entity could be known by different names and multiple identifiers between different project teams or different software tools. In such cases, such tagged values metadata in the SysML model can be used to record and unify such name

and identifier overloading across all projects of an organisation. This strategy has proven effective on substantial commercial applications of the WPA recipe.

As a general modelling policy, it is recommended that more systematic code-friendly element naming conventions are used, rather than familiar (yet typically less consistent) names, which are best carried instead as tagged values metadata.

A typical auxiliary profile including PA:Term is given in Figure 24 in **Appendix A** (p.34).

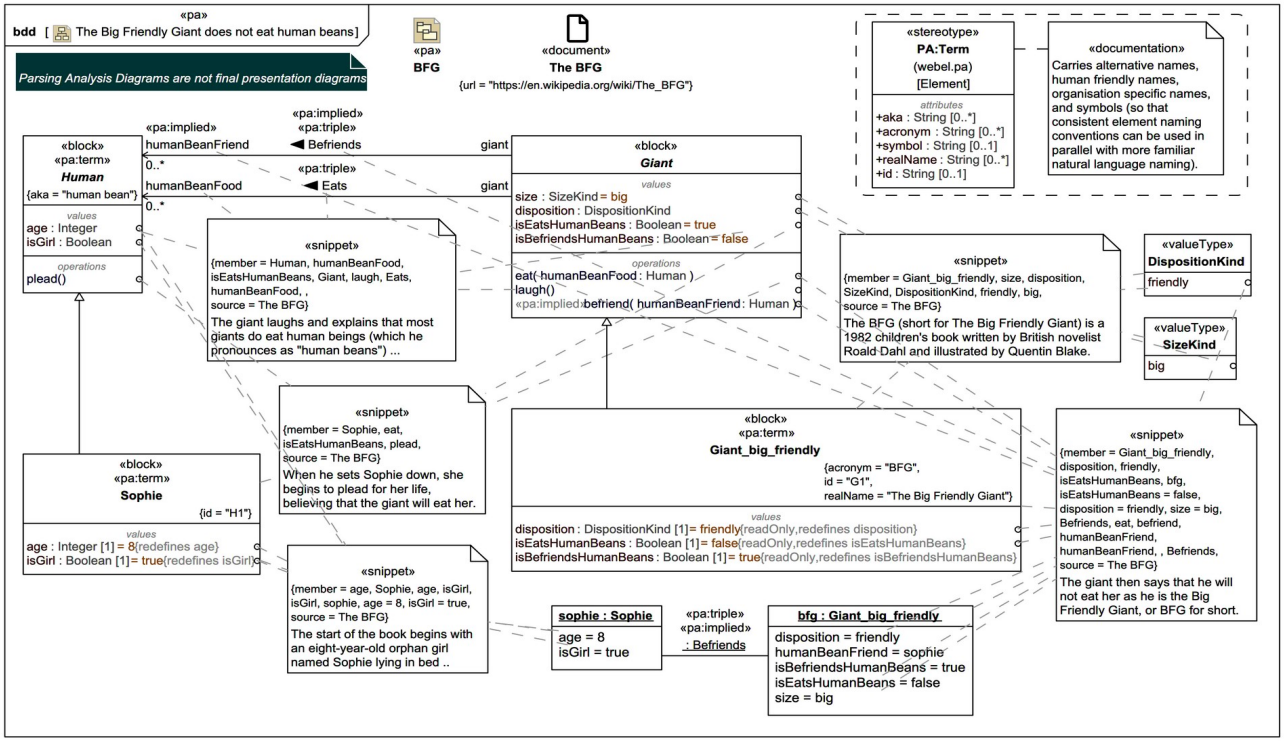

Figure 12: A PAD with tagged values for «pa:term» used to capture overloaded naming

Pseudo semantic triples

The WPA recipe introduces RDF-like subject-predicate-object semantic triples via the «pa:triple» stereotype keyword, which can be applied to a Dependency from a single source to a single target, or to a uni-directional one-to-one or one-to-many Association. They are denoted here *pseudo semantic triples (*more formal semantic modelling could be introduced through integration with the Ontology Definition Metamodel).

When the «pa:triple» keyword is applied to a Dependency, the source of the Dependency becomes the subject, the name of the Dependency becomes the predicate, the target becomes the object. The target and subject must be SysML Blocks and/or Actors or Properties typed by them. Custom stereotypes extending «pa:triple» may be used instead of names to capture recurring concepts after the Don't Repeat Yourself (DRY) principle.

When «pa:triple» is applied to an Association, the non-navigable end becomes the subject, the name of the Association is the predicate, and the navigable end is the object. The ends must be SysML Blocks and/or Actors. When displayed in diagrams, the Association line symbol should have the name and a direction arrow consistent with the navigation direction displayed. (The sense of direction is lost when used to type Connectors.)

A typical auxiliary profile including PA:Triple is given in Figure 24 in **Appendix A** (p.34).

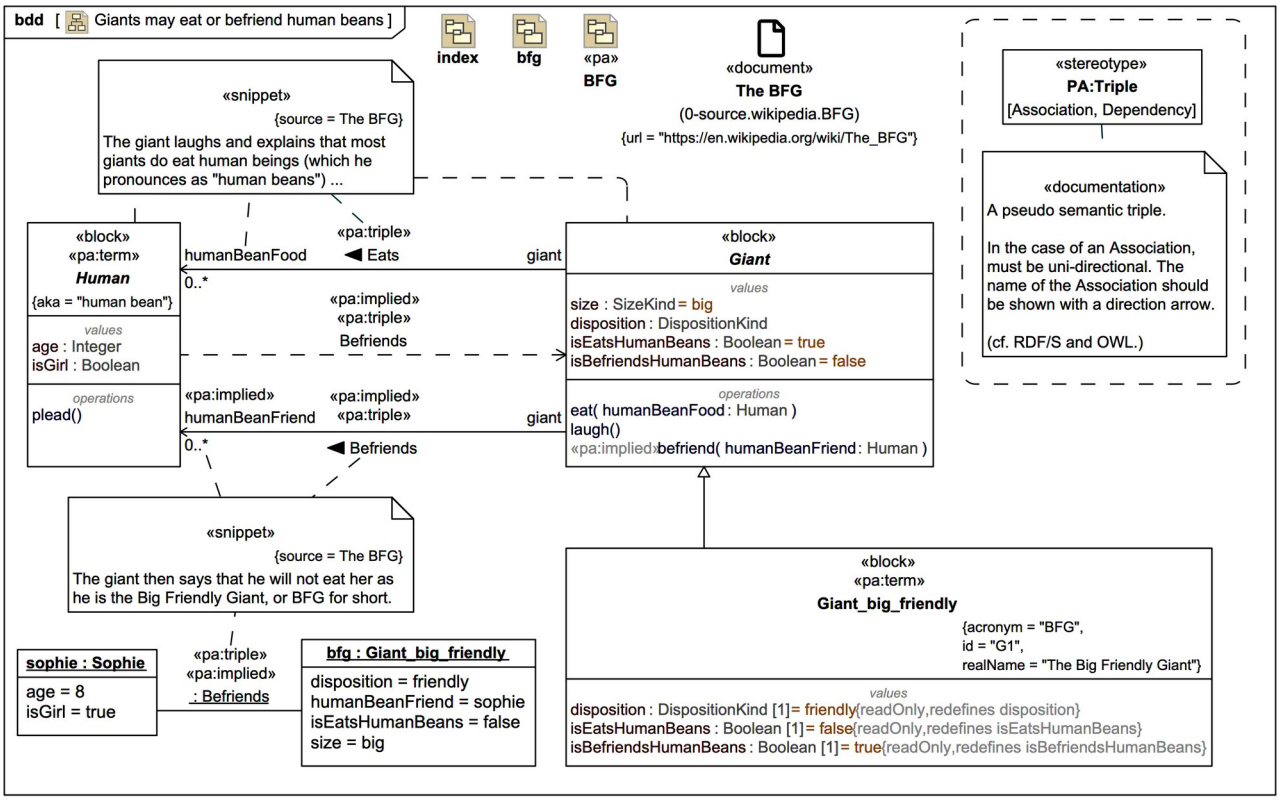

Figure 13: The «pa:triple» stereotype keyword applied to indicate Pseudo Semantic Triples on Associations and a Dependency

In Figure 13 the «pa:triple» keyword applied to an Association named `Befriends` between blocks `Giant` and `Human` yields these pseudo semantic triples:

- `Giant – Befriends – Human` *(Classifier level)*

- `giant:Giant – Befriends – humanBeanFriend:Human` *(Property level)*

Presumably a human can also befriend a giant, which is indicated for comparison using a Dependency named `Befriends` from block `Human` to block `Giant`.

More on indicating implied elicited elements

Formally, if «pa:implied» has been applied to indicate an implied elicited Element, the 'snippet' tagged value should be set using at least one Snippet that the modeller claims implies the existence of the Element, as shown in Figure 14. (Note, the Snippet should have identical 'name' and 'body'.) What matters is that this information is recorded in the underlying model, not that the ‘snippet’ tagged value is displayed, as it can be quite verbose.

A typical auxiliary profile including PA:Implied is given in Figure 24 in **Appendix A** (p.34).

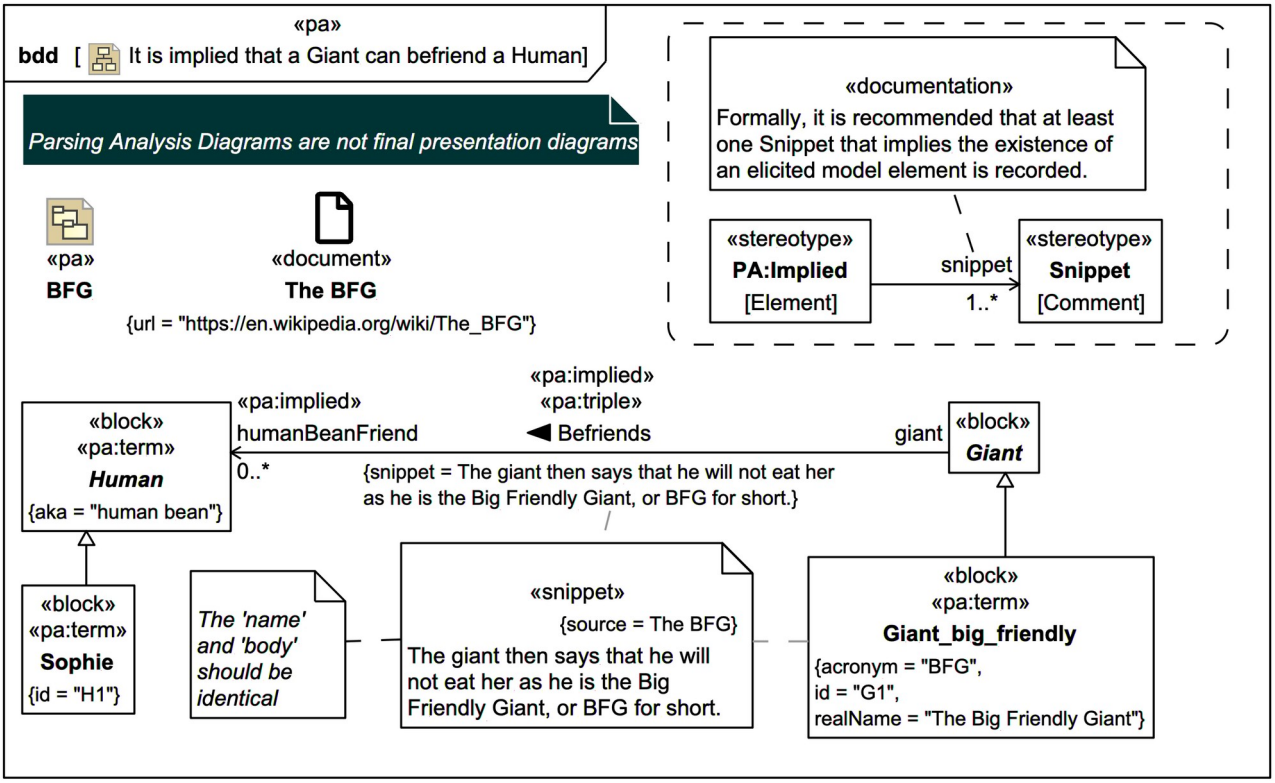

Figure 14: An Association with «pa:implied» applied and the 'snippet' tagged value set (and displayed) to record one Snippet that suggested the existence of the element

Indicating relationships between Snippets in SysML1.6 using tagged values

In SysML1.6, ElementGroups are not directly relatable. Relationships between Snippet extensions can, however, be indicated using additional tagged values for additional Snippet-typed attributes on the Snippet stereotype. This can be quite verbose in diagrams when Snippet names are long, although the tagged values need not always be displayed.

In Figure 15 a modeller has decided that the following 2 proverbs contradict each other:

> ‘Don't judge a book by its cover’
>
> ‘clothes maketh the man’

This has been indicated using the opposing tagged values `contradicts` and `contradictedBy`.

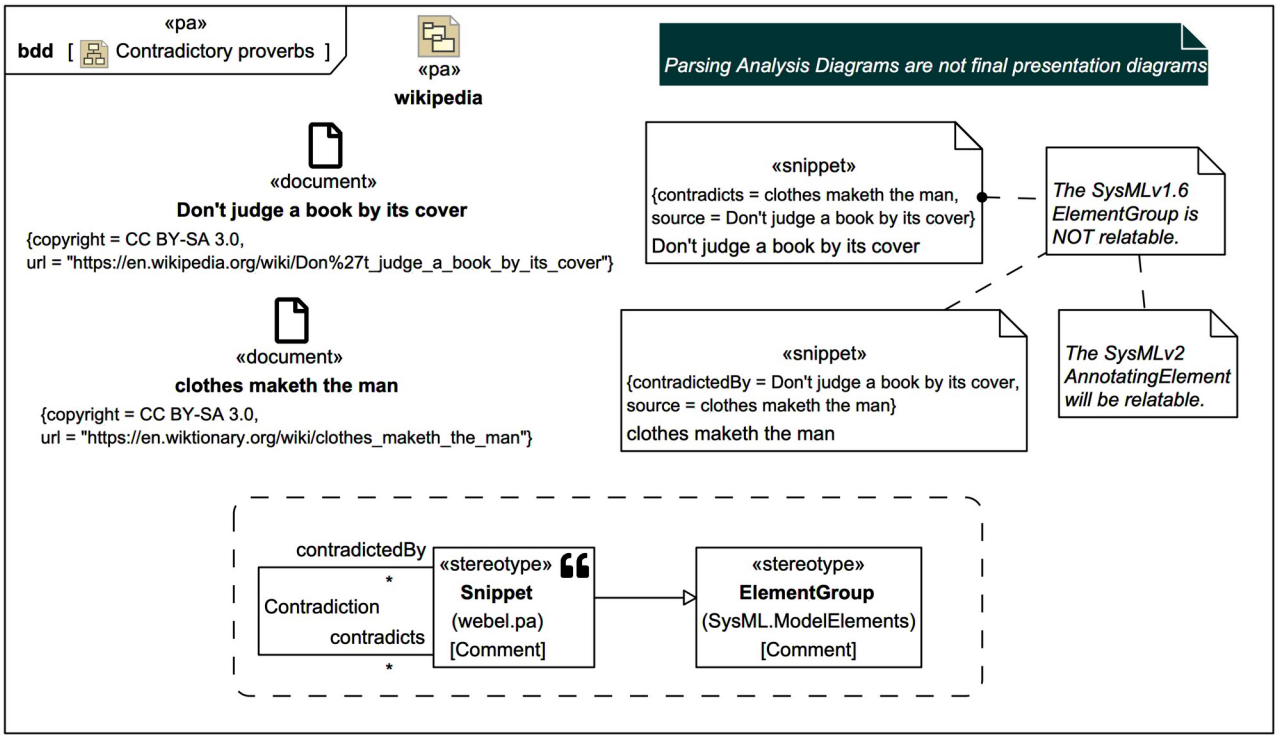

Figure 15: Additional Stereotype attributes used to indicate a Contradiction relationship between Snippets

The following text from Wikipedia is offered as a description of all optical telescopes, but in fact only applies well to Keplerian reflector style telescopes:

> ‘The basic scheme is that the primary light-gathering element … focuses that light from the distant object to a focal plane where it forms a real image.’
>
> ‘This image may be … viewed through an eyepiece, which acts like a magnifying glass. The eye … then sees an inverted magnified virtual image of the object.’

As shown in Figure 16, the modeller decides that this is contradicted by the following more accurate qualifying text:

> ‘There are telescope designs that do not present an inverted image such as the Galilean refractor and the Gregorian reflector. These are referred to as erecting telescopes.’

The claimed contradiction is again recorded using tagged values for `contradicts` and `contradictedBy` on the apparently contradictory Snippets.

This tagged values approach is robust, but it is quite visually verbose. It is also not as easily queried or traced as a first class Relationship. Hopefully SysMLv2 will help address this through introduction of the relatable AnnotatingElement, a candidate for extension by a v2 version of the WPA Snippet.

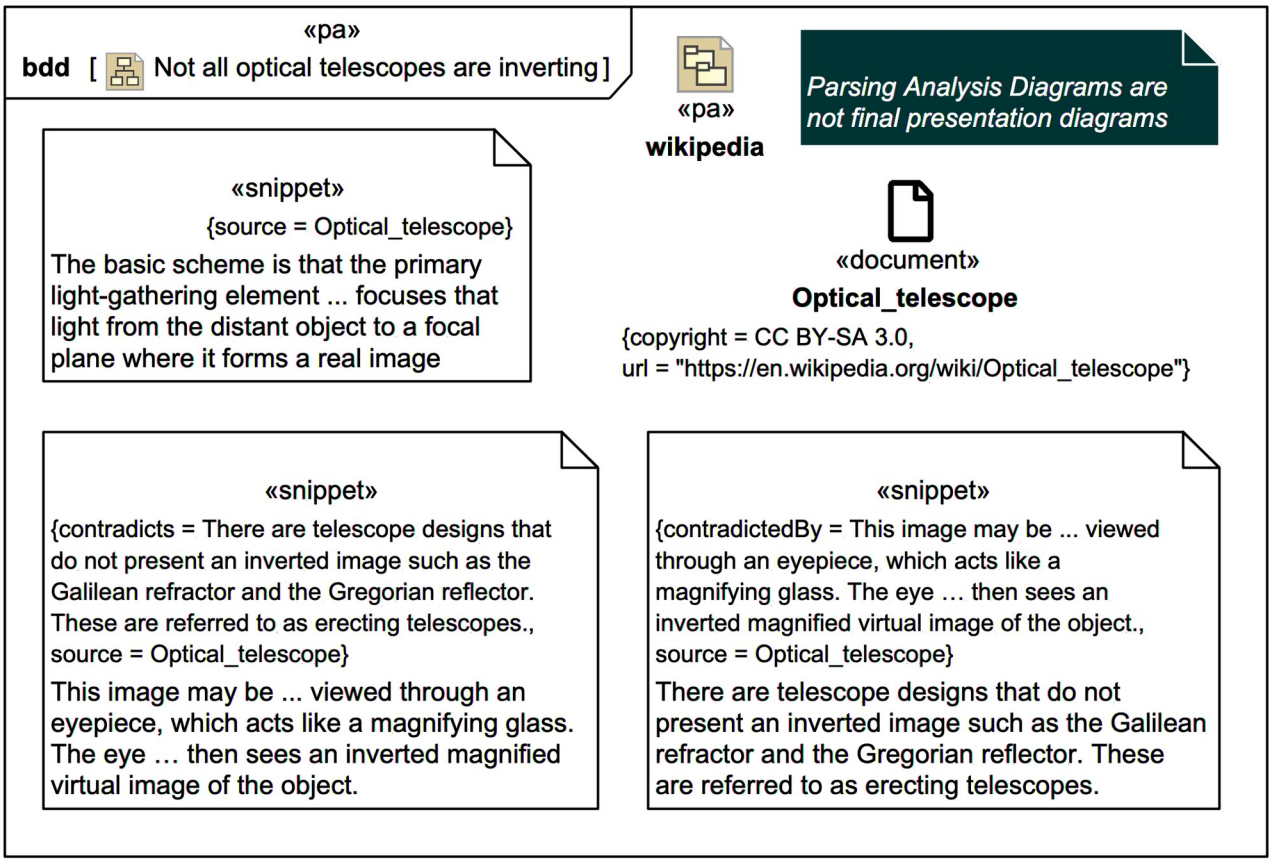

Figure 16: Another example of recording claimed contradictory text descriptions

Snippet trumps Requirement

The WPA recipe for text-driven model element elicitation is a meta-process that can be used in combination with other systems engineering methodologies including requirements engineering methodologies. Application of the WPA recipe typically precedes and informs the SysML-based requirements engineering process, because it is can be used not only to traceably elicit general SysML model Elements, it is used to traceably elicit Requirement elements (from any domain source document or database source), and often simultaneously the Elements that are to Satisfy a Requirement or are otherwise involved in it. This holds even when an existing external requirements database is to be incorporated.

In Figure 17 a Snippet for a Policy Note web page about WPA from the Webel IT Australia site has been used to elicit SysML Requirements, whilst also collecting members relevant to the Requirements. The source web page is:

https://www.webel.com.au/node/1805

The analysed text is:

> ‘Webel Parsing Analysis: A Snippet (keyword «snippet») MUST always have exactly one domain 'source' Document (keyword «document»).’

This yields 3 Requirements, one of which is claimed to be satisfied by the ‘source’ attribute within the `Snippet` stereotype within the `webel/pa` profile. The /member tagged value of the Snippet lists the 3 elicited Requirements along with some other relevant Elements.

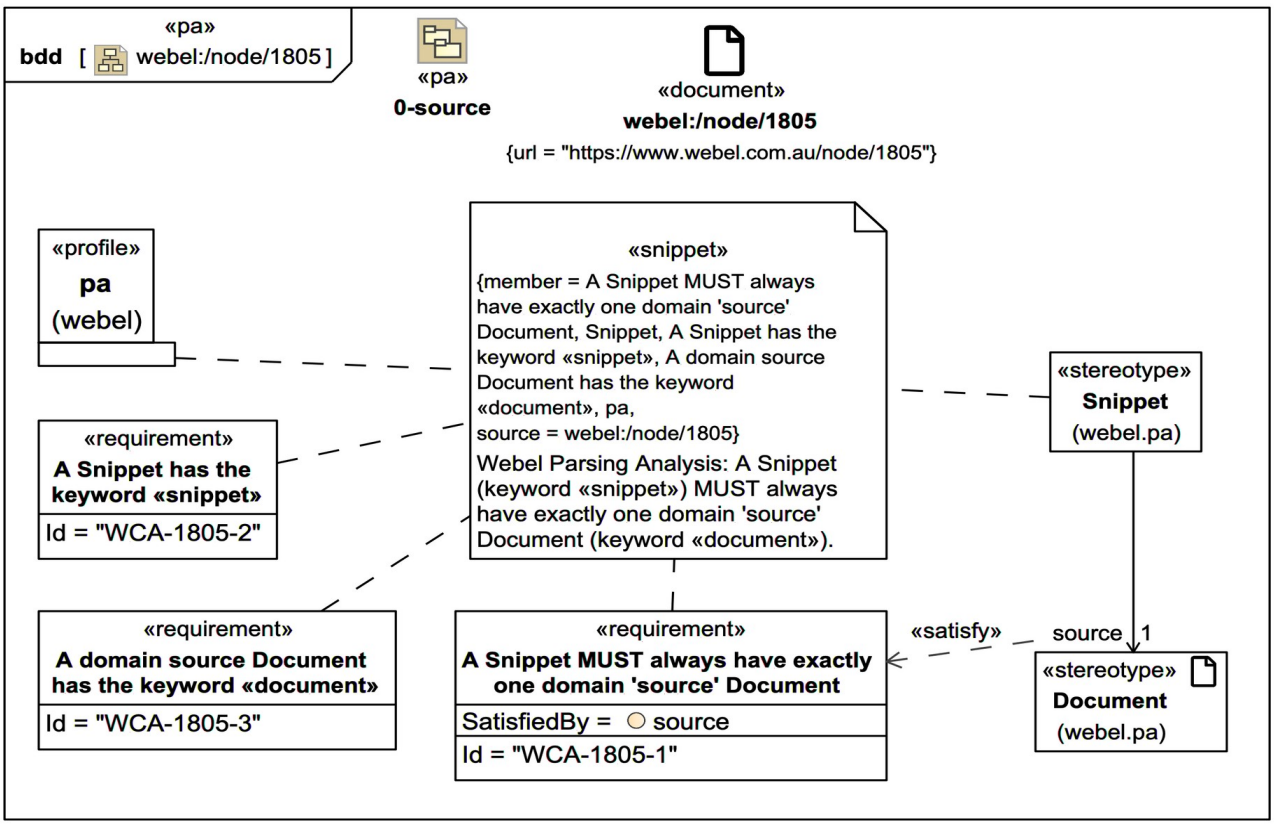

Figure 17: A satisfied SysML Requirement for Webel Parsing Analysis extracted from a text statement

Using Trace to quickly elicit model elements

Sometimes, instead of using text from a domain source document, one wishes to elicit model elements from a figure, drawing, diagram, graphic, or table (which may or may not also contain text and/or engineering values). The SysML Trace relationship can be used to quickly elicit model elements from a Document identifying a non-text source.

In Figure 18 the analysed domain source «document» with diagrams was a PDF that shows pin allocations for In-Circuit Serial Programming (ICSP) headers for the Serial Peripheral Interface (SPI) for ATmega2560 and ATmega16u2 chips on an Arduino Mega2560Rev3 micro-controller board. The elements that were elicited from the diagram «document» can be nicely listed in MagicDraw using a compartment for the SysMLv1.6 operation `Trace::getTracedFrom(in ref)`.

The BDD Figure 18 verbosely shows the dashed line arrow symbol and «trace» keyword for every Trace used; these symbols can be visually removed from diagrams as each element is elicited to reduce visual clutter (with the information remaining in the underlying model).

There are currently some minor limitations when using Trace in SysMLv1.6 for the purpose of general model element elicitation tracing. One issue is that only a NamedElement can

be a source or target of a Trace, so a Slot, for example, can't be traced this way. That's a concern when Slots are used to hold elicited engineering values, such as the header-pin to chip-pin breakout mappings shown in Figure 18.

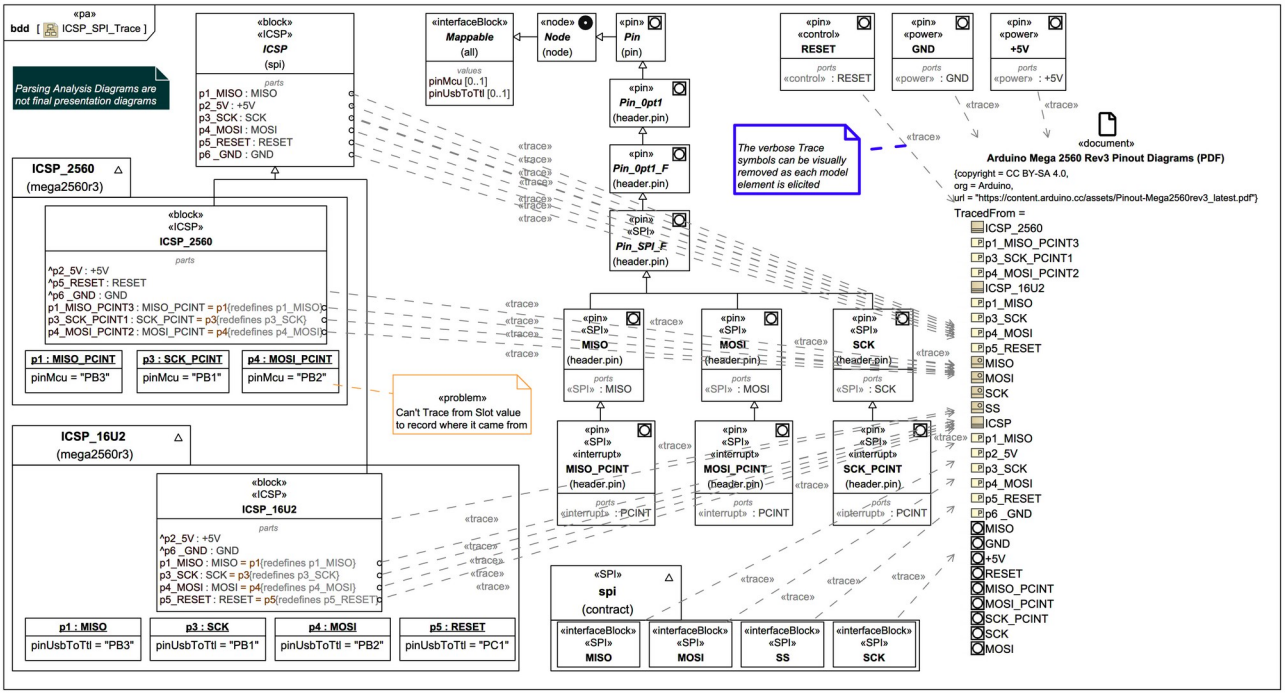

Figure 18: SysML Trace used to traceably elicit model elements from an identified diagram as domain source document

Another issue is that the SysMLv1.6 `/tracedTo` operation is only available on AbstractRequirement. However, in the MagicDraw tool you can use a *derived relationship* to achieve the equivalent on other kinds of NamedElement, as shown in the query table in Figure 19.

#	Name	Owner	▽ Trace target to source
1	+5V	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
2	GND	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
3	ICSP	spi	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
4	ICSP_16U2	ICSP_16U2	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
5	ICSP_2560	ICSP_2560	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
6	MISO	spi	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
7	MISO	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
8	MISO_PCINT	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
9	MOSI	spi	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
10	MOSI	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
11	MOSI_PCINT	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
12	p1_MISO	*ICSP*	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
13	p1_MISO	ICSP_16U2	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
14	p1_MISO_PCINT3	ICSP_2560	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
15	p2_5V	*ICSP*	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
16	p3_SCK	*ICSP*	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
17	p3_SCK	ICSP_16U2	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
18	p3_SCK_PCINT1	ICSP_2560	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
19	p4_MOSI	*ICSP*	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
20	p4_MOSI	ICSP_16U2	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
21	p4_MOSI_PCINT2	ICSP_2560	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
22	p5_RESET	*ICSP*	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
23	p5_RESET	ICSP_16U2	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
24	p6 _GND	*ICSP*	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
25	RESET	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
26	SCK	spi	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
27	SCK	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
28	SCK_PCINT	pin	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)
29	SS	spi	Arduino Mega 2560 Rev3 Pinout Diagrams (PDF)

Figure 19: A generic MagicDraw table with a /tracedTo column for any element type using a derived relationship

Not every sentence from every domain source document

The Webel Parsing Analysis recipe adds value progressively and incrementally. The more it is used, the more correspondence between a SysML model and external resources describing a system can be achieved. It is usually neither practical nor necessary to process every single sentence of every single available domain source document.

The recipe works best with high quality domain source documents with little repetition and a high semantic “signal-to-noise” ratio.

Typically, a domain source document contains key paragraphs that initially yield many elicited model elements. As one progresses through further paragraphs, the number of newly elicited model elements per Snippet reduces, a sign that the process is converging to an accurate, consistent SysML model.

For example, sentences from a document describing the transformation of signals along the signal path of a radio-telescope may map well to elements that are eventually represented in an IBD with parts connected via ports with signal flows.

Indicating non-WPA elements using special character name prefixes

During highly creative SysML modelling, a modeller who is otherwise employing the WPA recipe might wish to quickly create elements that are not directly elicited from the text extract handled by a Snippet. Such elements can be indicated using a special prefix character such as '@', '$', or '*'.

For example, a newly created block named `@DesignIdea` could indicate that the block is not yet related to any external domain source document. If an external domain source document referencing the same design idea then later becomes available, the modeller may choose to then apply the WPA recipe to newly sourced text and then upgrade the element to be fully elicited by changing the block's name to simply `DesignIdea`.

In practice, the use of some special punctuation characters in Element names can sometimes have unwanted side effects in tools, so some care must be taken.

Snippets lend scope to SysML diagrams and modelling

One challenge often encountered by SysML modellers is deciding exactly how much of the underlying model a particular diagram should expose. The WPA recipe promotes a clearly scoped modelling workflow.

Since the purpose of Parsing Analysis Diagrams (PADs) is to elicit model elements from the text of one or more Snippets, they are by nature well scoped. Elements that are not mentioned by a Snippet do not belong on a PAD for it (except for immediately supporting general elements of implied Generalizations).

The inclusion of Snippets in Presentation Diagrams can also suggest a clear scope through the text narrative. Typically, a well scoped Presentation Diagram only includes elements that are explicitly mentioned in the text of any Snippets shown in the diagrams, and nearest neighbour elements and/or elements that are relevant to the text narrative.

The WPA recipe promotes robust convergence of SysML models; the more the text of the Snippets processed covers the description of a system, the more the WPA SysML model converges on a consistent model of that system.

From DISE to MBSE

The WPA recipe for SysML acts also as part of a broader strategy for migrating from *Document-Intensive Systems Engineering (DISE)*[1] to Model-Based Systems Engineering with SysML. Through selective mapping of text extracts from multiple domain source documents into a consistent, centralised SysML model, breakages of Single Source of Truth can be identified, whilst also compiling a registry of critical domain source documents, and with complete traceability to elicited SysML model elements.

1 The term Document-Intensive Systems Engineering (DISE) was coined by Prof. Michael Vinarcik.

Super-relational Policy Note and Snippet web pages

The WPA recipe is supported by a growing list of online Policy Note pages on a *Content Management System (CMS)* web site (Figure 20):

https://www.webel.com.au/sysml/parsing_analysis/notes

Each Policy Note page has a unique CMS identifier. The title of each Policy Note page is verbose and may be more than one sentence. Policy Notes are categorised by type and policy level strictness using tag links. Each dedicated Policy Note page references one or more relevant linked content pages and/or Snippet pages and vice versa (Figure 21).

The verbose linked title of a Policy Note page may be displayed embedded in other content pages – such as tutorial trail pages – in various formats (Figure 22). This approach – using verbose linked titles – is denoted *super-relational* in the Webel modelling recipe.

Note also how the verbose title of a Snippet CMS web page represents quoted text, and may likewise be referenced from other pages as a verbose, specially formatted hyperlink.

While the WPA recipe can be used entirely within a SysML tool such as MagicDraw, the parallel use of a CMS can be effective especially for tutorial trails and demonstrations.

Please refer also to the **Appendix C – Additional examples and resources** section (p.52).

Index of Webel Parsing Analysis recipe ❑ Notes for SysML

Currently NOT ordered or prioritised. The format is:

`[NOTE_TYPE_TAG, ...]{POLICY_LEVEL_TAG} LINKED_TITLE`

❑ [ASSERTION]{TIP} Humans work well with natural language: Many stakeholders – including those who are not necessarily familiar with UML or SysML modelling symbols and diagrams - benefit enormously from having plain text in diagrams side-by-side with graphics.

❑ [TIP]{INFORMATIVE} In professional applications of the Webel Parsing Analysis recipe for SysML, a robust reusable profile for «part»/«assembly»/«leaf» components is used, along with stereotypes keywords like «assembles» for indicating physical composition hierarchies.

❑ [POLICY]{STRICT} Webel Parsing Analysis: A 'source' Document must have one (only) of a URL or a URN (includes ISBN or other unique identifier) for use as a URI

❑ [ASSERTION]{TIP} Translating authoritative technical documents written in "natural" engineering language (snippet-by-snippet) into Webel Parsing Analysis diagrams creates consistent underlying systems models and can bridge easily to existing methodologies.

❑ [TIP]{INFORMATIVE} In Webel Parsing Analysis tutorials, Wikipedia pages are often used as a placeholder sources to represent authoritative domain source documents from which text extract "snippets" are sourced, and from which SysML model elements are traceably elicited.

❑ [CONVENTION, MODELLING, TIP]{RECOMMENDED} Webel Parsing Analysis: It does not matter whether you use a Package or a Model package. (The informal Webel convention is that Models are used for systems engineering analysis with SysML and Packages are reserved for code-related software engineering.)

❑ [ASSERTION]{TIP} Even relatively informal elicitation of model elements using the SysML ElementGroup and a Parsing Analysis approach is extremely powerful.

❑ [POLICY]{TIP} In the Webel Parsing Analysis recipe you are not obliged to map every single part of a Snippet's text extract to a UML or SysML model, just traceably elicit model elements of immediate or anticipated interest to your task or goal.

❑ [CAPABILITY, FEATURE, NAVIGATION, TIP, TOOL]{RECOMMENDED} Webel Parsing Analysis: If you have a unique URL for a domain source «document» you MAY additionally set it as an external hyperlink in the MagicDraw/Cameo tool (as well as setting it as the unique 'url' tagged value that acts as the '/uri').

❑ [TIP]{INFORMATIVE} The Webel Parsing Analysis recipe for SysML1.x does not use the ElementGroup directly, it extends it as a user-defined Snippet stereotype with keyword «snippet» and with MagicDraw Customization.

❑ [ASSERTION]{TIP} The Webel Parsing Analysis recipe promotes a clearly scoped modelling workflow.

❑ [QUESTION]{STRONG} Webel Parsing Analysis: SysMLv1.x: Q: "Instead of using a Snippet extension of an ElementGroup, can't I just use a Requirement and Trace to achieve the same thing?" A: No! The ElementGroup extended as the Snippet is far more powerful and fit for purpose.

❑ [DISPLAY, MODELLING, NAMING, POLICY, TIP]{STRICT} Webel Parsing Analysis: Pseudo Semantic Triple: For «pa:triple» applied to a named one-to-one or one-to-many uni-directional Association, the non-navigable end Property is the subject, the name is the predicate, the navigable end Property is the object.

❑ [MODELLING, POLICY, TOOL]{STRICT} Webel Parsing Analysis: As StateMachine Diagrams are always owned by an element that is not eventually under the Source Input Zone they should not be used as «pa» diagrams. Elicit States instead via the /member section of the specification dialog.

❑ [MODELLING, POLICY]{STRICT} Webel Parsing Analysis: SysML: Pseudo Semantic Triple: A «pa:triple» may be applied to a uni-directional Association between SysML Blocks and/or Actors.

❑ [ASSERTION, DISPLAY, POLICY, SETTINGS, STYLE, TOOL]{STRICT} Webel Parsing Analysis: If you wish to show a «snippet» comment symbol (with its body text) in a presentation diagram (that is NOT a «pa» diagram) remove the '/members' tagged value from display so the only visible tagged value is 'source'.

❑ [TIP]{SUGGESTED} Webel Parsing Analysis: You may create custom stereotypes applicable to a Dependency or a uni-directional Association to represent a «predicate» much like a semantic triple (subject-predicate-object) in OWL/RDF.

❑ [MODELLING, NAMING, POLICY]{STRICT} Webel Parsing Analysis: Acronym: PAD = Parsing Analysis Diagram (may be nearly any diagram type, except those types that must be owned by an elicited model element)

❑ [OPTION]{STRONG} Webel Parsing Analysis: Formally, if «pa:implied» has been applied to indicate an implied elicited Element, the 'snippet' tagged value should bet set using at least one Snippet that implies it. (Note, the Snippet should have identical 'name' and 'body'.)

❑ [TIP, TOOL]{RECOMMENDED} Webel Parsing Analysis: If you are processing lots of text from one source «document» create a STUB «snippet» with that «document» set as 'source' and just make copies of it in the model browser as needed. Big time saver!

❑ [TIP]{SUGGESTED} Webel Parsing Analysis: You do not need to map every word of every snippet to the UML/SysML model! Extract only the information required for your domain modelling task to add incremental benefit.

❑ [DISPLAY, MODELLING, POLICY]{STRONG} Webel Parsing Analysis: An "index" Parsing Analysis Diagram showing a collection of Snippets need not always show the members of each Snippet; a "focus" Parsing Analysis Diagram for one or more Snippets SHOULD always show the members of every Snippet.

❑ [OPTION]{RECOMMENDED} Webel Parsing Analysis: Track and display alternative names, human friendly names, organisation-specific names, and identifiers using tagged values for a custom stereotype «pa:term» (adapt or extend as required).

❑ [POLICY, STYLE]{STRICT} Webel «pa» Parsing Analysis Diagrams (PADs) are "scratchpads" used to elicit model elements traceably from text «snippets». While BDDs are a good initial choice, most types of SysML diagram can be used as a «pa» diagram.

❑ [CAVEAT]{INFORMATIVE} One limit of the SysML-1.6 ElementGroup for use as a Snippet in the v1.x version of Webel Parsing Analysis for SysML is that an ElementGroup is not directly relatable.

❑ [MODELLING, NAMING, POLICY]{RECOMMENDED} Webel Parsing Analysis: An anonymous Element may be collected as a /member of a Snippet (it is not important whether collected elements list with a clear name under /member, only that they are traceably elicited).

❑ [CAPABILITY, TIP]{INFORMATIVE} The SysML Trace relationship can be used as a quick way to traceably elicit model elements from an identified diagram or table from a domain source document. You may visually remove the Trace symbols as each element is elicited to reduce clutter.

❑ [DISPLAY, MODELLING, NAMING, POLICY, TIP]{STRICT} Webel Parsing Analysis: Pseudo Semantic Triple: For «pa:triple» applied to a named one-to-one or one-to-many uni-directional Association, the non-navigable end Property is the subject, the name is the predicate, the navigable end Property is the object.

❑ [TIP]{INFORMATIVE} The grouping of member elements by an ElementGroup is denoted here "logical" in the sense that it does not "steal ownership" of member elements (as opposed to "physical" ownership of elements in the model)

❑ [ASSERTION]{INFORMATIVE} Webel Parsing Analysis (with strict traceability of all elicited model elements) can be used side-by-side with "freestyle" modelling with incremental benefit the more it is used.

❑ [QUESTION]{INFORMATIVE} SysMLv1.x: Q: Why can't a Package with PackageImports be used as a Parsing Analysis text container? Why is the SysML1.6 ElementGroup (extended and customised as the Webel «snippet») far better suited for text-driven model element elicitation?

❑ [DISPLAY, MODELLING, NAMING, POLICY, TIP]{STRICT} Webel Parsing Analysis: Pseudo Semantic Triple: For «pa:triple» applied to a named one-to-one Dependency, the source is the subject, the name of the Dependency is the predicate, the target is the object.

❑ [OPTION]{RECOMMENDED} In Webel Parsing Analysis if you want to track elicited model elements that have not been explicitly mentioned in source text you can introduce Element-level custom tracking Stereotypes with the keywords «pa:implied» or «pa:assumed».

❑ [TIP]{EDITORIAL} In Webel Parsing Analysis every Package, Model, or Diagram used to perform Parsing Analysis MUST have the stereotype keyword «pa» applied, and MUST live under a top-level Model or Package 'source' separate from the rest of the project model.

❑ [MODELLING]{INFORMATIVE} SysMLv2: On the v2 Comment extension of the v2 AnnotatingElement as a candidate Parsing Analysis Container

❑ [DISPLAY, MODELLING, NAMING, POLICY, TIP]{STRICT} Webel Parsing Analysis: Pseudo Semantic Triple: When «pa:triple» is applied to a named uni-directional Association the name and a consistent direction arrow should be displayed on the Association symbol in diagrams.

❑ [CONVENTION, TIP]{STRONG} Webel Parsing Analysis: typically the quoted extract text (body) of a «snippet» is a short phrase, a sentence, or a couple of short sentences (but not dozens of sentences).

❑ [ASSERTION]{STRICT} In Webel Parsing Analysis elicited model elements MUST eventually be moved out of the 'source' (or '0-source') Package/Model zone and under a main project Package/Model area. (Use the owner display option to check.)

❑ [TIP]{INFORMATIVE} Webel Parsing Analysis: The absolute basics of the process (SysMLv1.x form)

❑ [MODELLING, POLICY]{STRICT} Webel Parsing Analysis: SysML: Pseudo Semantic Triple: A «pa:triple» may be applied to a Dependency between SysML Blocks and/or Actors or Properties typed by them.

❑ [POLICY, STYLE]{STRICT} Webel Parsing Analysis «pa» diagrams are NOT intended as final presentation diagrams! They serve merely to traceably elicit model elements, which may then be shown in other (typically much tidier) presentation diagrams elsewhere.

❑ [DISPLAY, STYLE, TIP]{SUGGESTED} Webel Parsing Analysis: If the dashed-line "anchors" from your «snippet» ElementGroup symbols to elicited elements make a «pa» Parsing Analysis Diagram (PAD) too hard to read, try making them light grey using symbol properties.

❑ [POLICY]{STRICT} Webel Parsing Analysis: Create a top-level 'source' or '0-source' Package/Model SEPARATE from your project model. All source «document» elements and «snippet» extracts and «pa» diagrams MUST be ultimately owned by that top-level Package/Model as ancestor!

❑ [TIP]{RECOMMENDED} Webel Parsing Analysis: MagicDraw/Cameo: In parsing diagrams use the element compartment properties to only display the features relevant to the context of a particular snippet's text extract.

❑ [MODELLING, NAMING, POLICY]{STRICT} Webel Parsing Analysis: While Generalizations may be collected as elicited members of a Snippet this can quickly lead to clutter in the /member list display.

❑ [POLICY]{STRICT} Webel Parsing Analysis: A custom Document stereotype (keyword «document») extends the UML standard profile Document that extends Artifact (which is NOT in the UML4SysML intersection). Such a Document may act as the 'source' of a «snippet».

❑ [OPTION]{RECOMMENDED} TIP: If you use the name '0-source' instead of 'source' for the top-level Package/Model for containing Parsing Analysis elements it will list nicely at the top of the containment tree.

❑ [MODELLING, NAMING, POLICY]{STRICT} Webel Parsing Analysis: While Associations may be collected as elicited members of a Snippet this can quickly lead to clutter in the /member list display. Often just collecting an end Property as member is sufficient.

❑ [POLICY]{STRICT} Webel Parsing Analysis: A Snippet (keyword «snippet») MUST always have exactly one domain 'source' Document (keyword «document»).

❑ [MODELLING, NAMING, POLICY]{RECOMMENDED} Webel Parsing Analysis: Where too many dashed line "anchors" from Snippets to elicited members lead to clutter they may be selectively omitted from a Parsing Analysis Diagram (PAD) once each member has been collected.

❑ [ASSERTION]{TIP} Not even the most experienced requirements engineers can easily state requirements perfectly in consistent requirements-friendly language first shot; requirements and constraints are often buried in the natural text of domain source documents.

❑ [TIP]{RECOMMENDED} Webel Parsing Analysis: Relationships between «snippet» items can be tracked using additional tagged values such as 'contradictedBy' and 'contradicts'. This is robust, but it can be verbose when Snippets have long names if the tagged values are displayed.

❑ [MODELLING, NAMING, POLICY]{STRICT} Webel Parsing Analysis: The name of a Parsing Analysis Diagram (PAD) may be drawn from a focus Snippet OR may simply indicate a topic of interest to the analysis

❑ [ASSERTION]{INFORMATIVE} Webel Parsing Analysis is a "meta-process" that can be applied to general domain source documents to elicit Requirements (and related model elements) or to a Requirements Specification itself as a special case of a domain source document!

❑ [POLICY]{STRICT} Webel Parsing Analysis diagrams do their job once - namely traceable elicitation of model elements - and are then only kept as a reference! The elicited model elements are then used elsewhere in the final model.

❑ [ASSERTION]{SUGGESTED} Webel Parsing Analysis: A stereotype with keyword «pa:from» may be applied to a Dependency from a Package to another Package within the Source Input Zone to indicate that all of its Elements were directly or indirectly elicited from source Documents.

❑ [CAVEAT, ISSUE]{INFORMATIVE} The markup of Snippet page titles on this CMS web site is sometimes restricted (especially concerning subscripts and superscripts such as used in mathematical and scientific notation). Visit the linked Snippet page for the quote with full markup.

Figure 20: Overview of WPA Policy Note page verbose linked titles

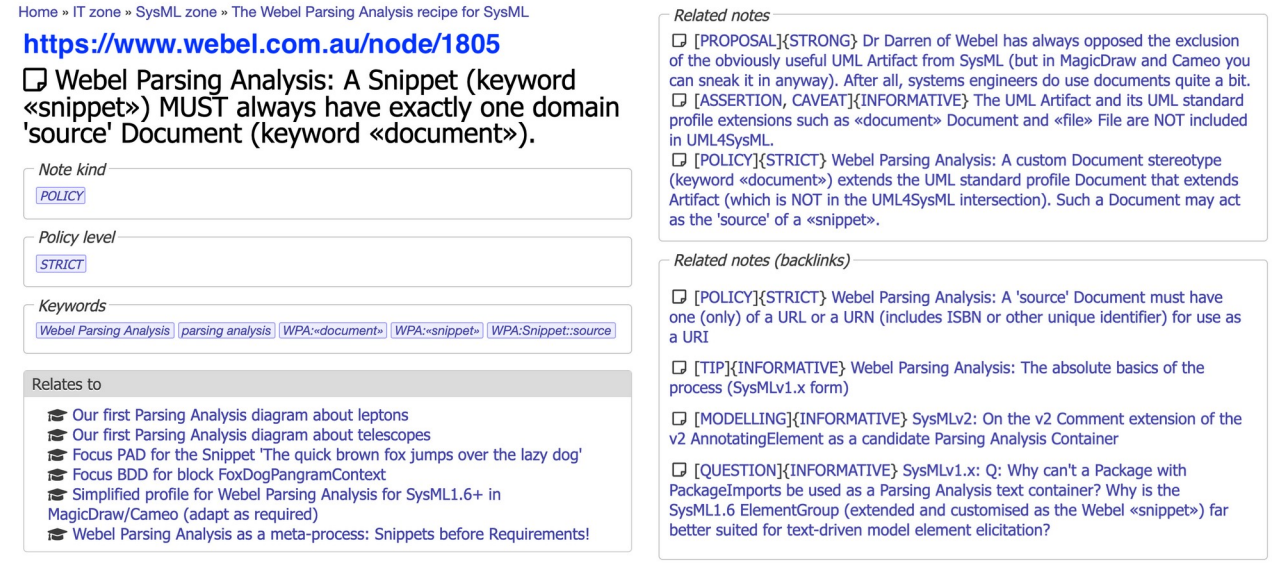

Figure 21: A Policy Note page linked to relevant content and other Notes and Snippets

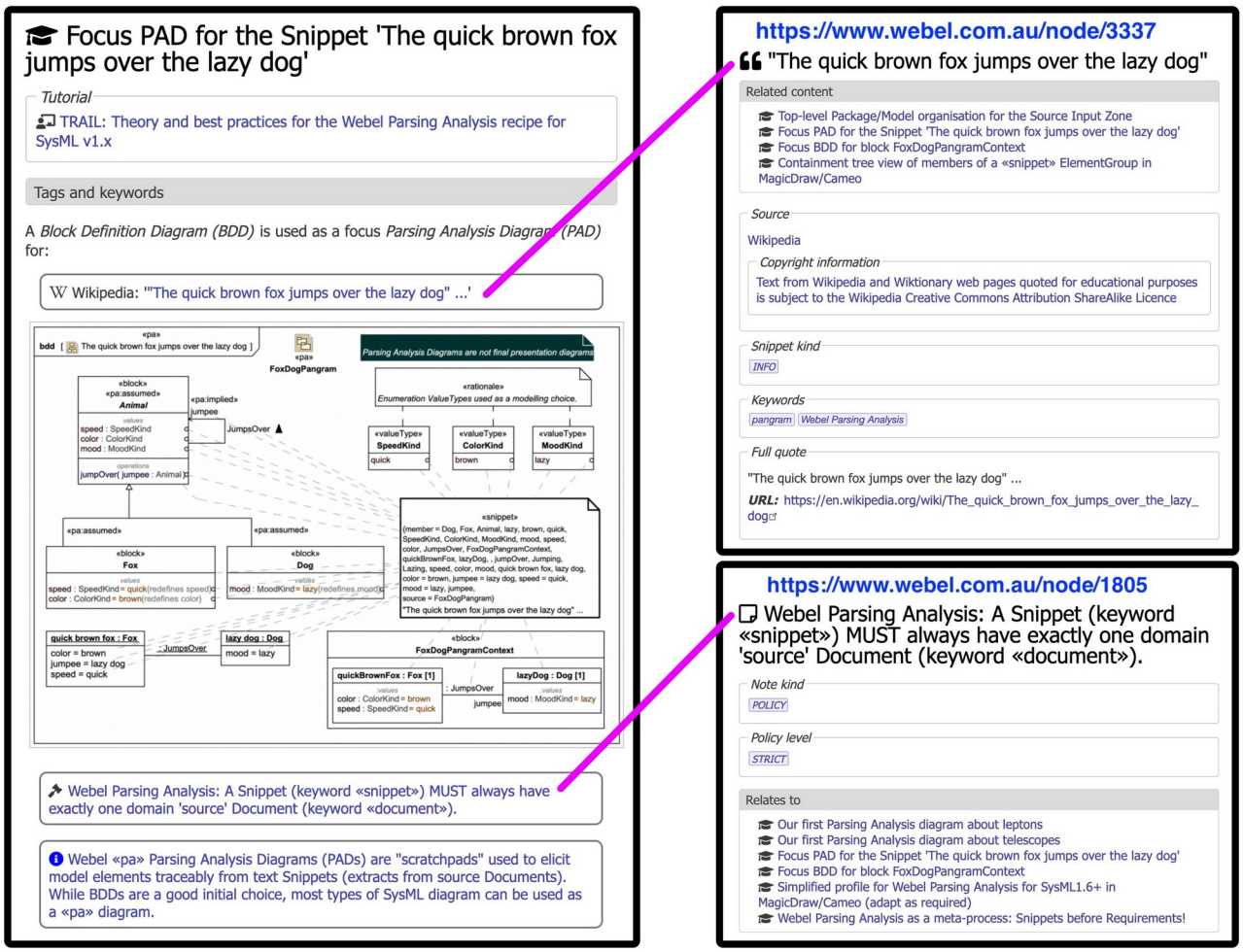

Figure 22: A content page referencing Policy Note pages and a Snippet page

Appendix A – Typical Profiles and DSL Customizations

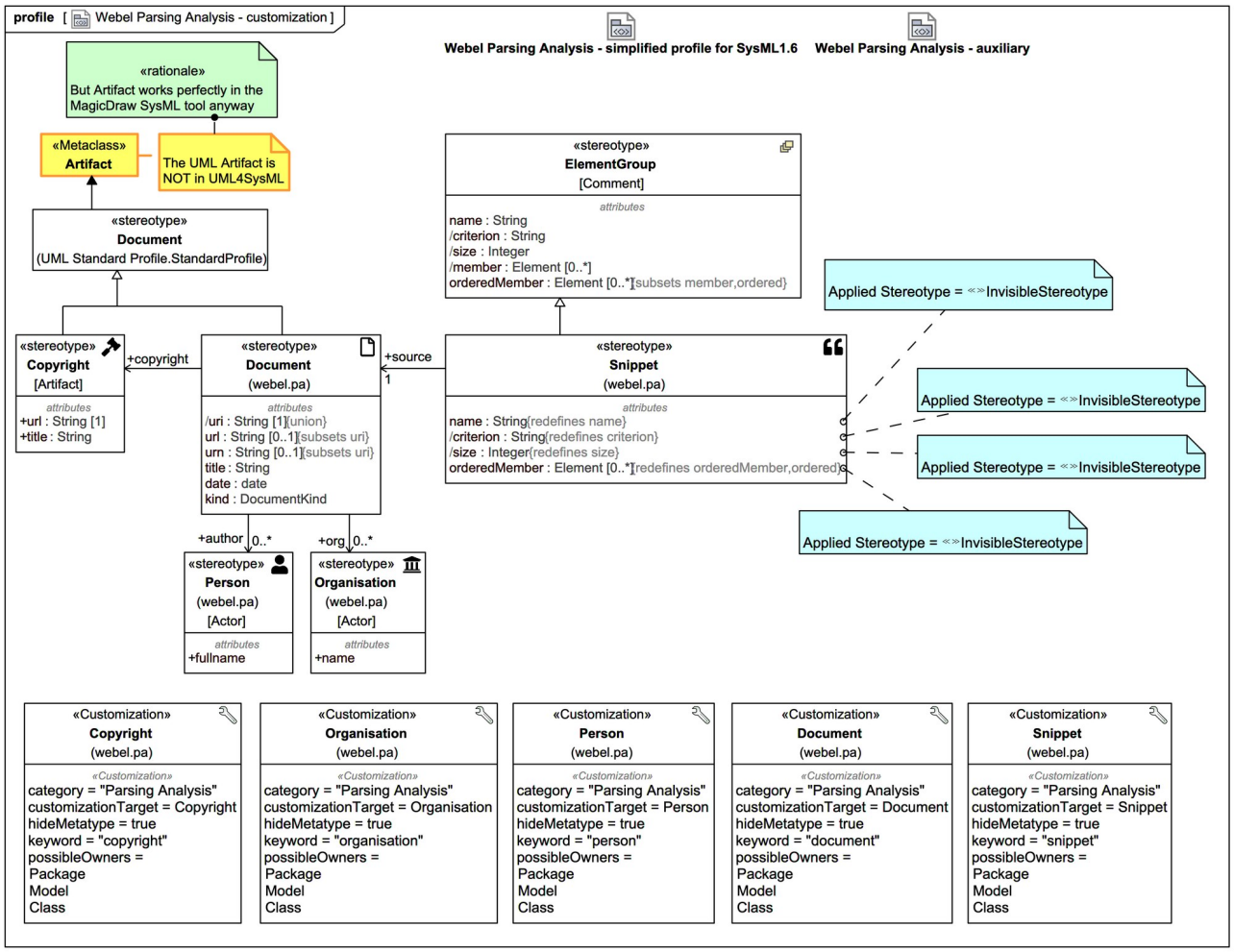

Figure 23: MagicDraw Customizations for a typical Webel Parsing Analysis profile

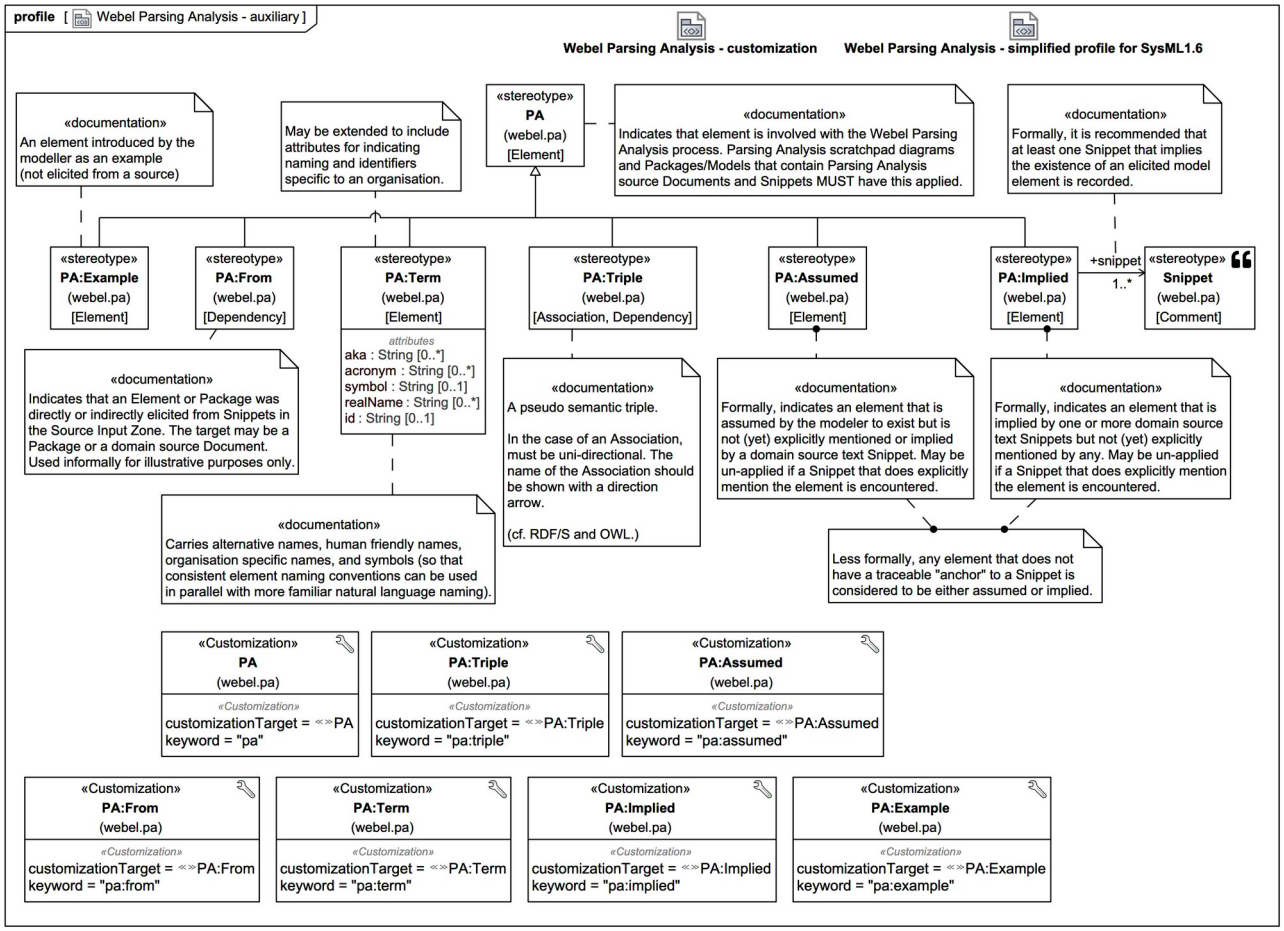

Figure 24: Additional stereotypes for a typical Webel Parsing Analysis profile

Appendix B – Example applications

Selected Parsing Analysis Diagrams (PADs) and resulting Presentation Diagrams illustrating elicited model elements from realistic example applications are shown.

For links to the full resources please refer to: Appendix C – Additional examples and resources (p.52).

CASE STUDY: Wikipedia optical telescopes

Selected diagrams from an online tutorial trail:

> TRAIL: Webel SysML Parsing Analysis example: Optical telescopes from Wikipedia: Structure and port-based light flow model
>
> https://www.webel.com.au/sysml/parsing_analysis/telescopes

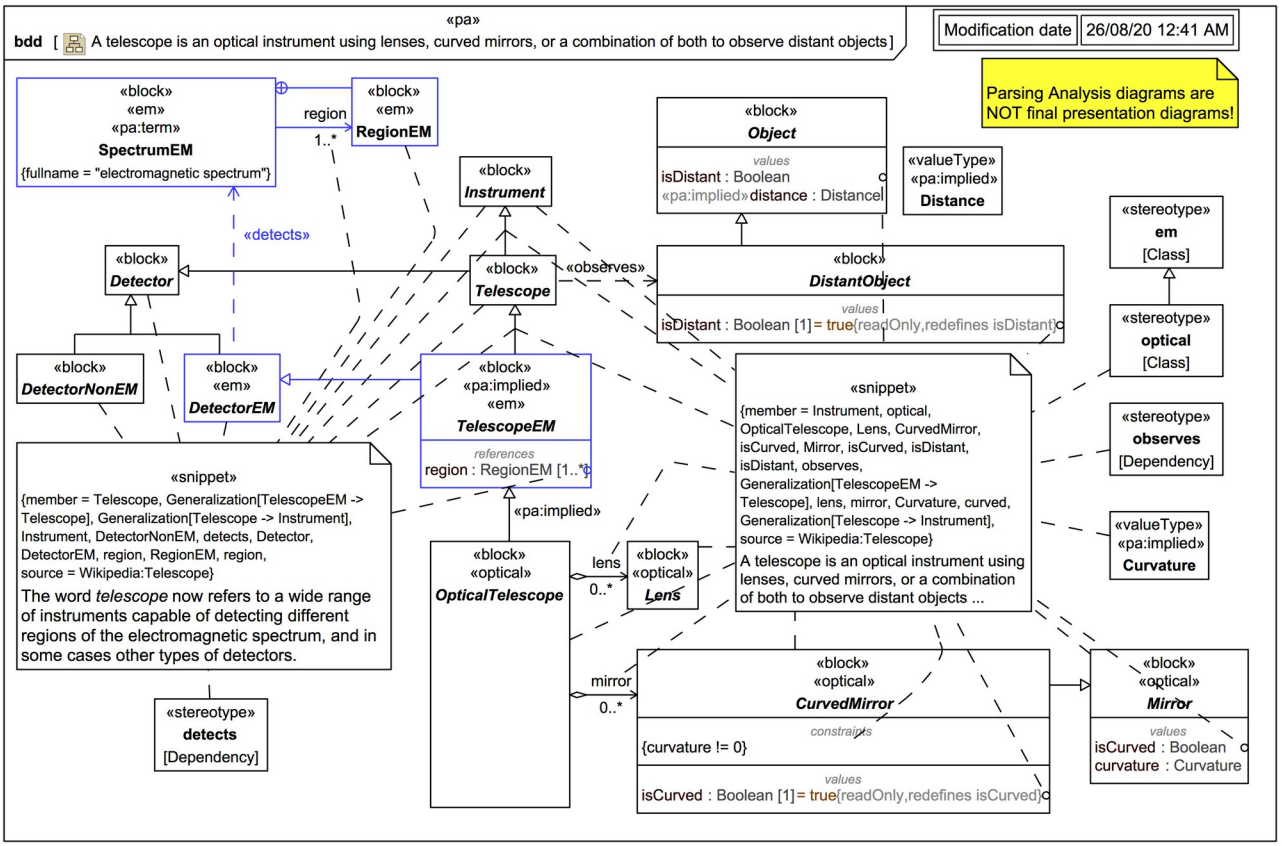

Figure 25: A BDD as focus PAD for eliciting model elements from a Snippet telescopes

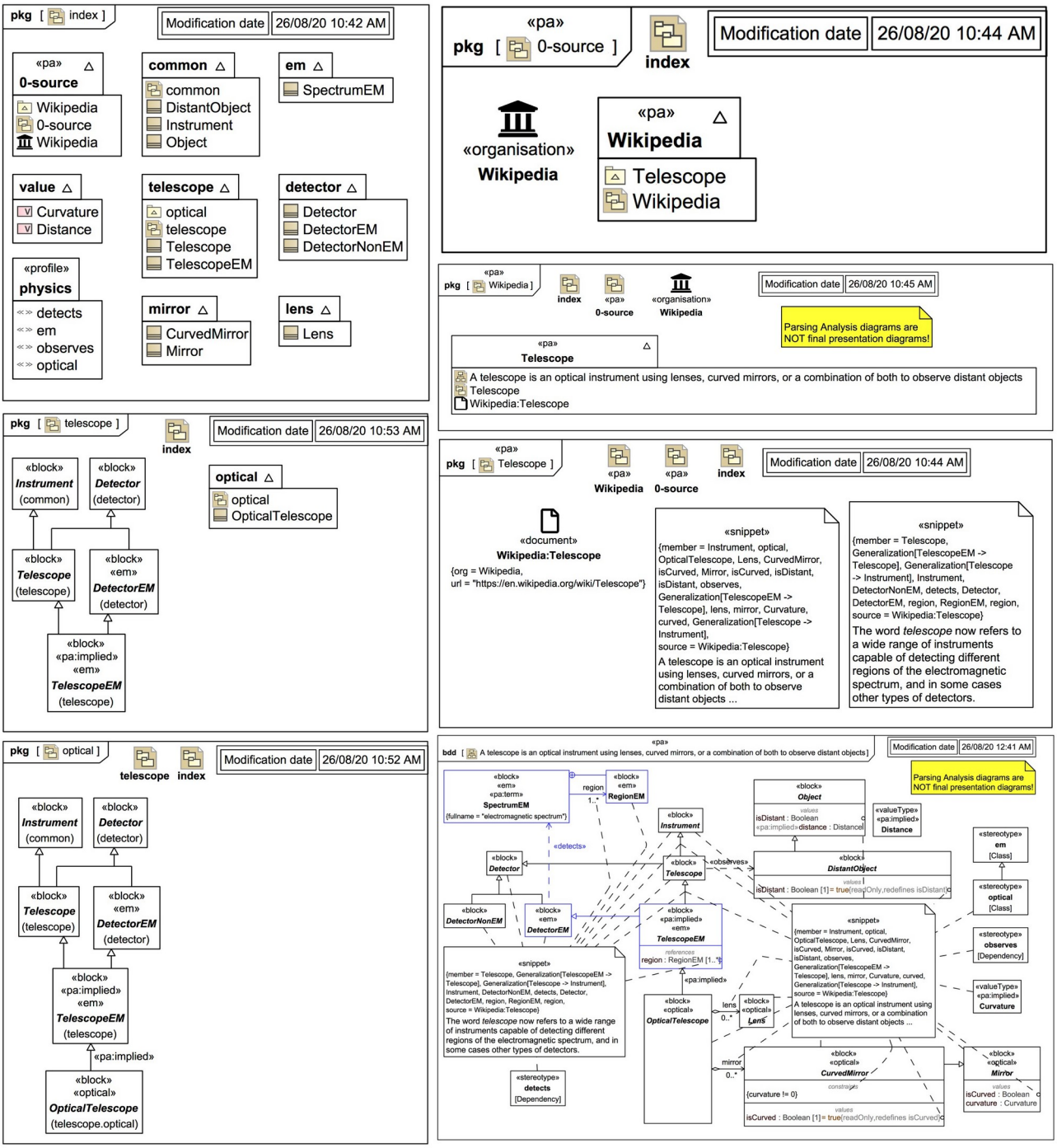

Figure 26: Overview of diagrams after moving elicited elements out of the '0-source' zone

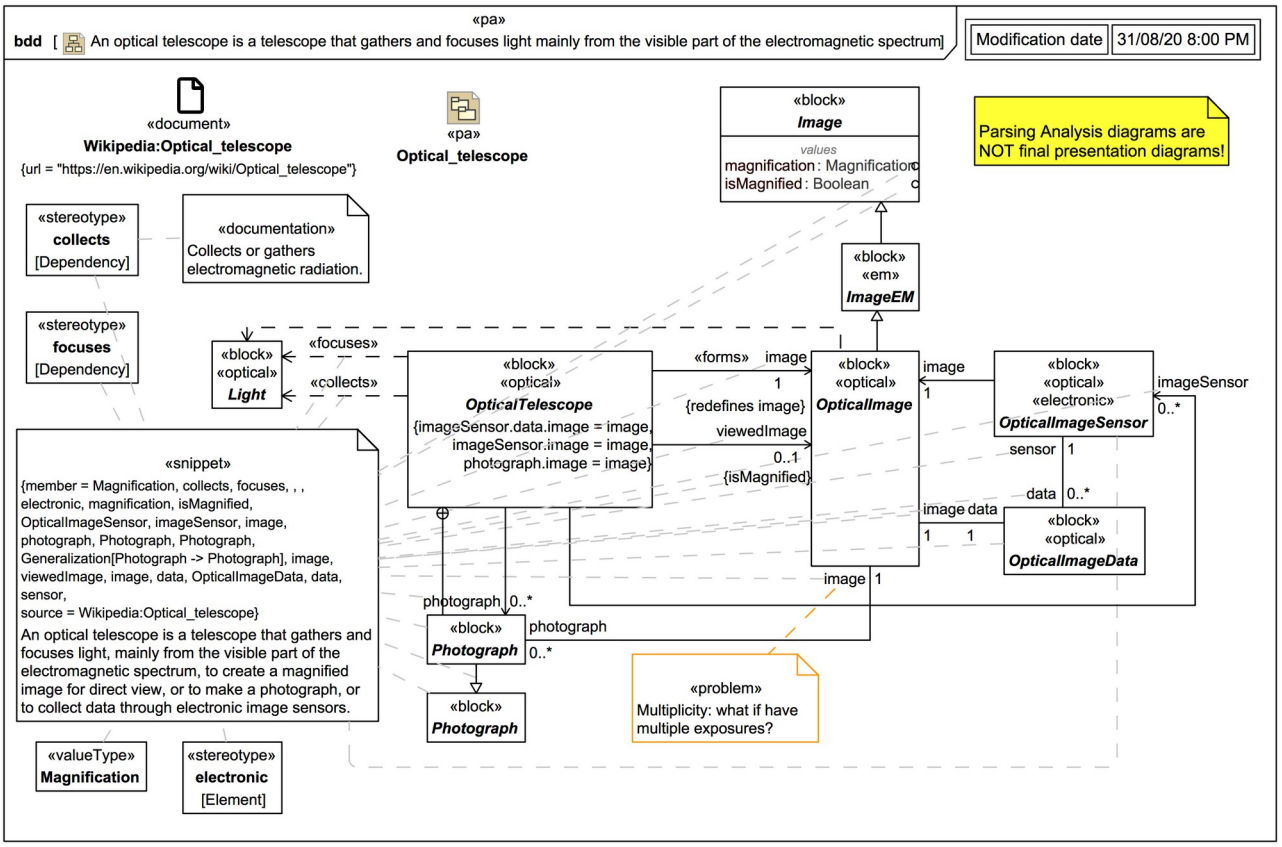

Figure 27: PAD with Dependency stereotypes as pseudo-semantic triples

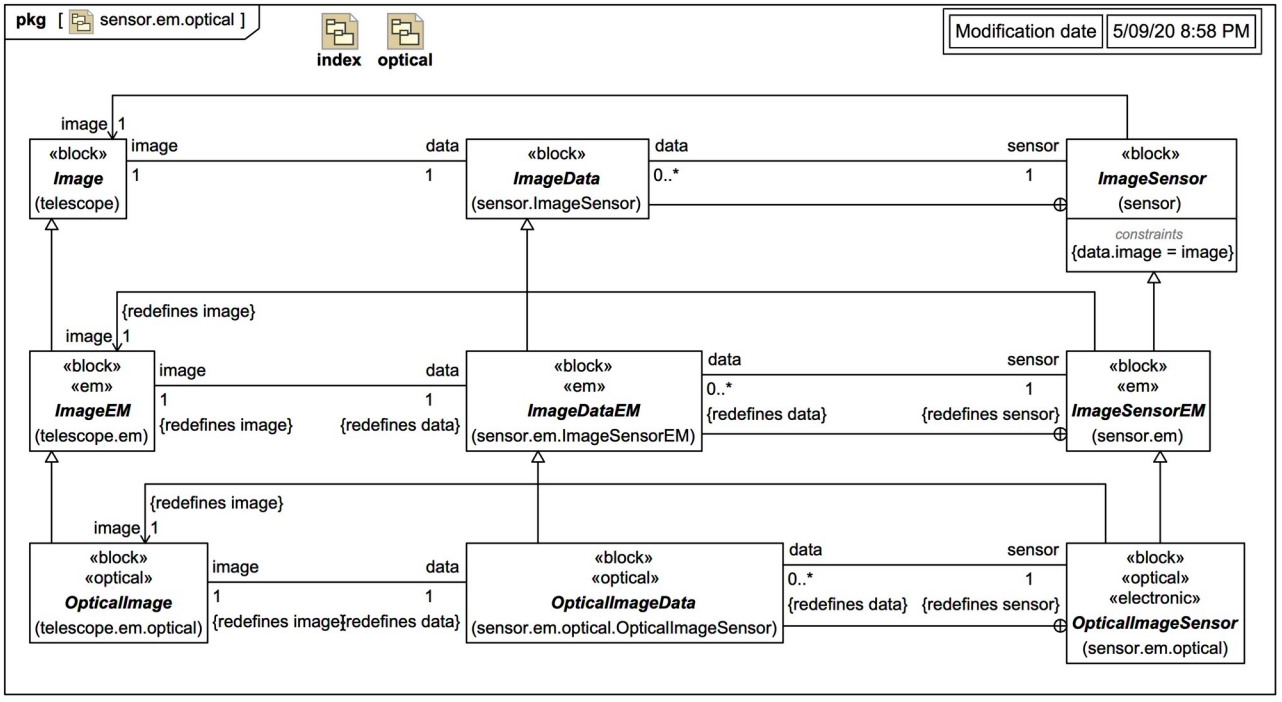

Figure 28: Package Diagram with elicited Block hierarchy and redefinition sequence

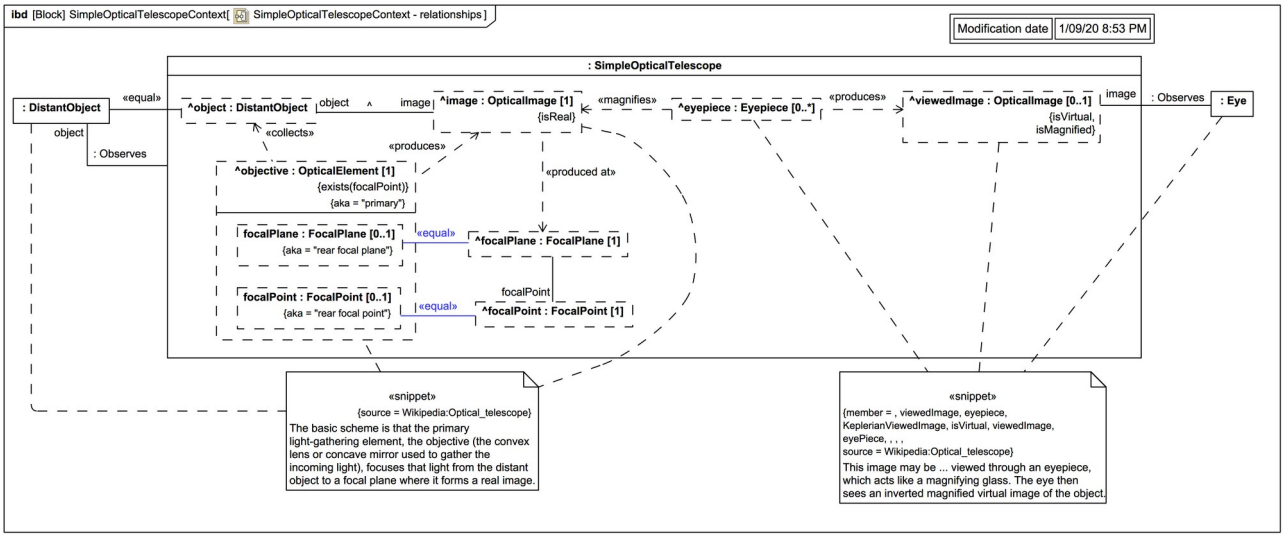

Figure 29: IBD of simple optical telescope context with Snippets as commentary. Includes illustrative Dependency stereotypes a pdeudo semantic triples.

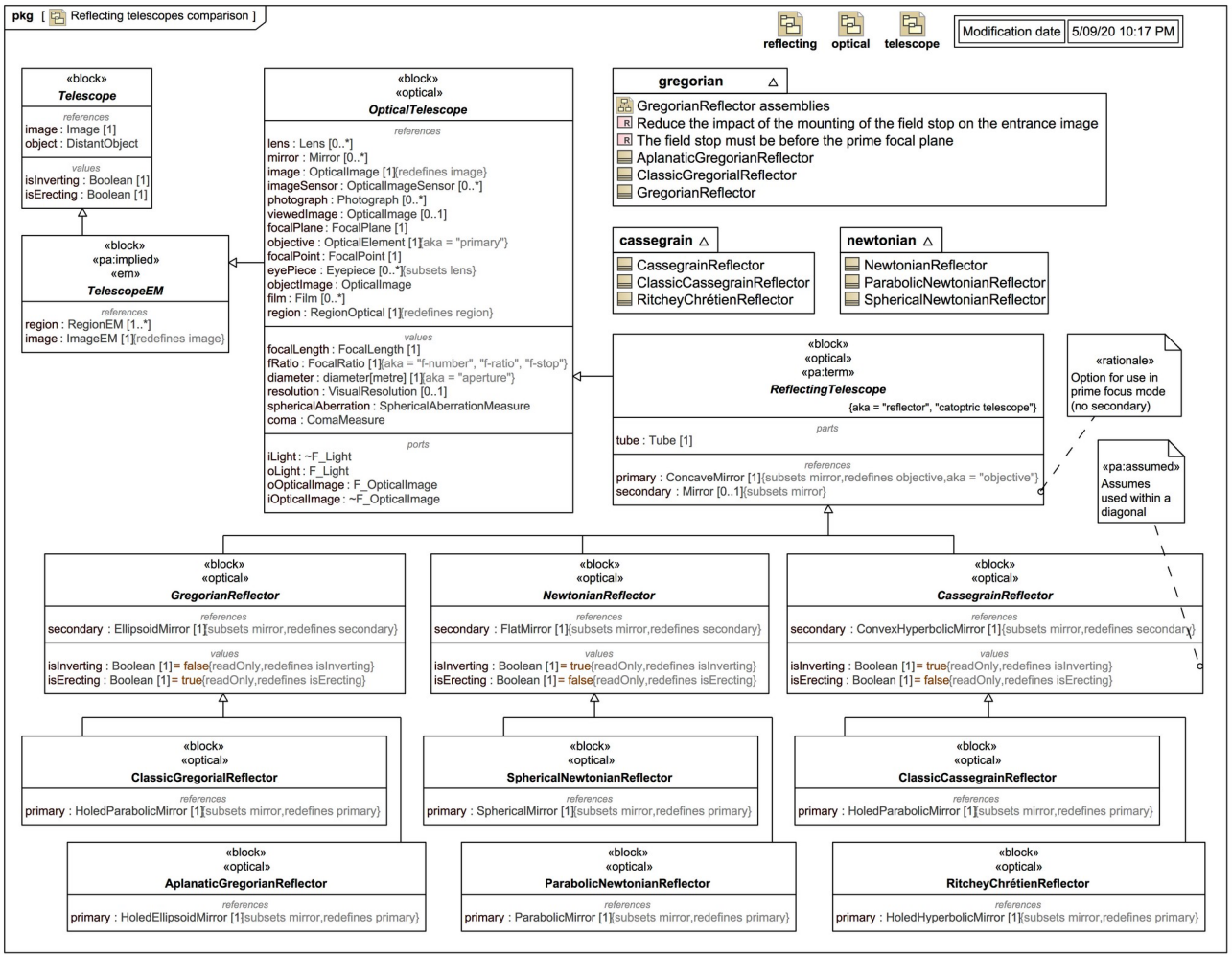

Figure 30: Package Diagram overview of hierarchy of elicited reflecting telescope Blocks

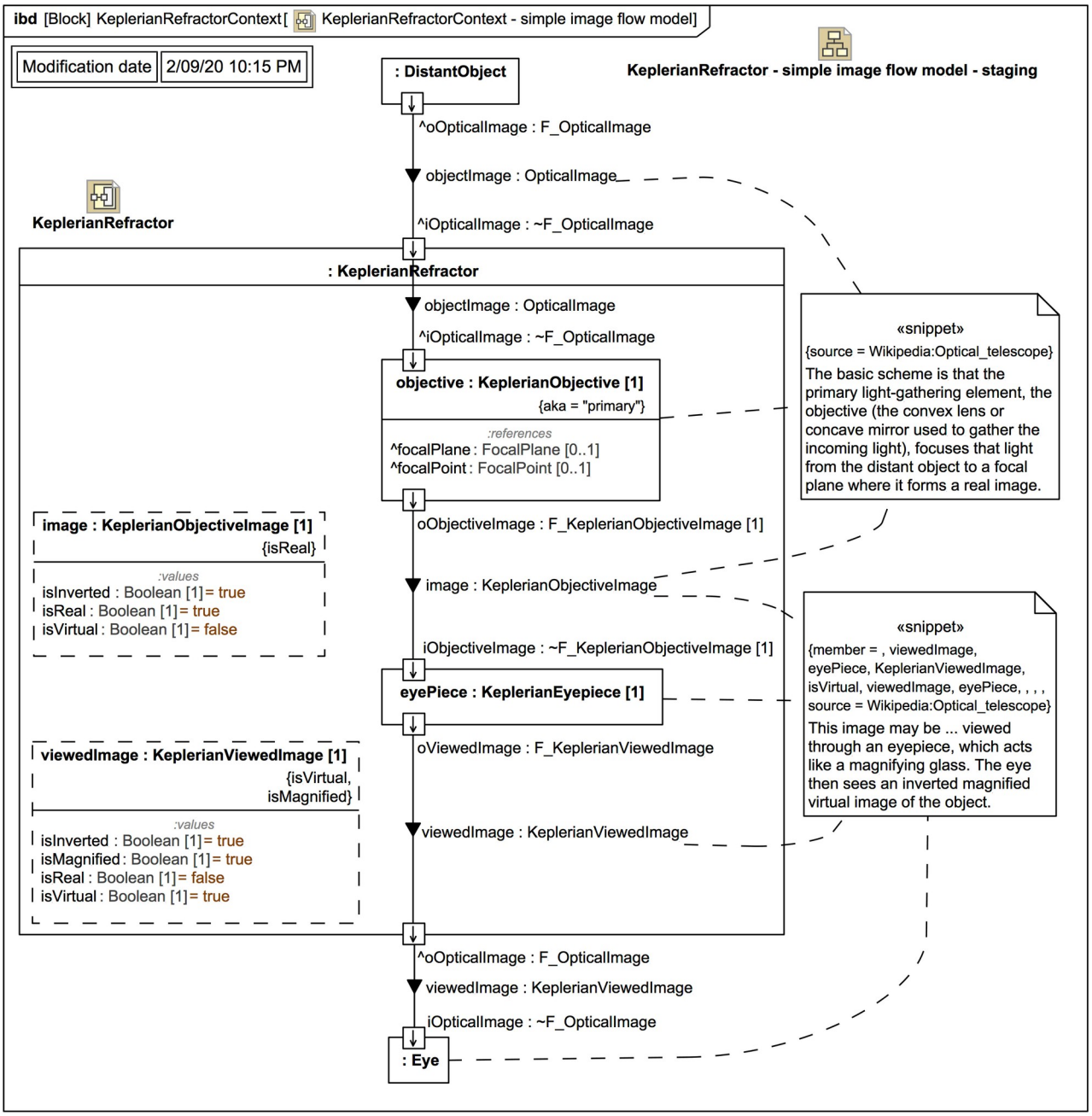

Figure 31: IBD with simplified port-based light flow model and Snippets as commentary

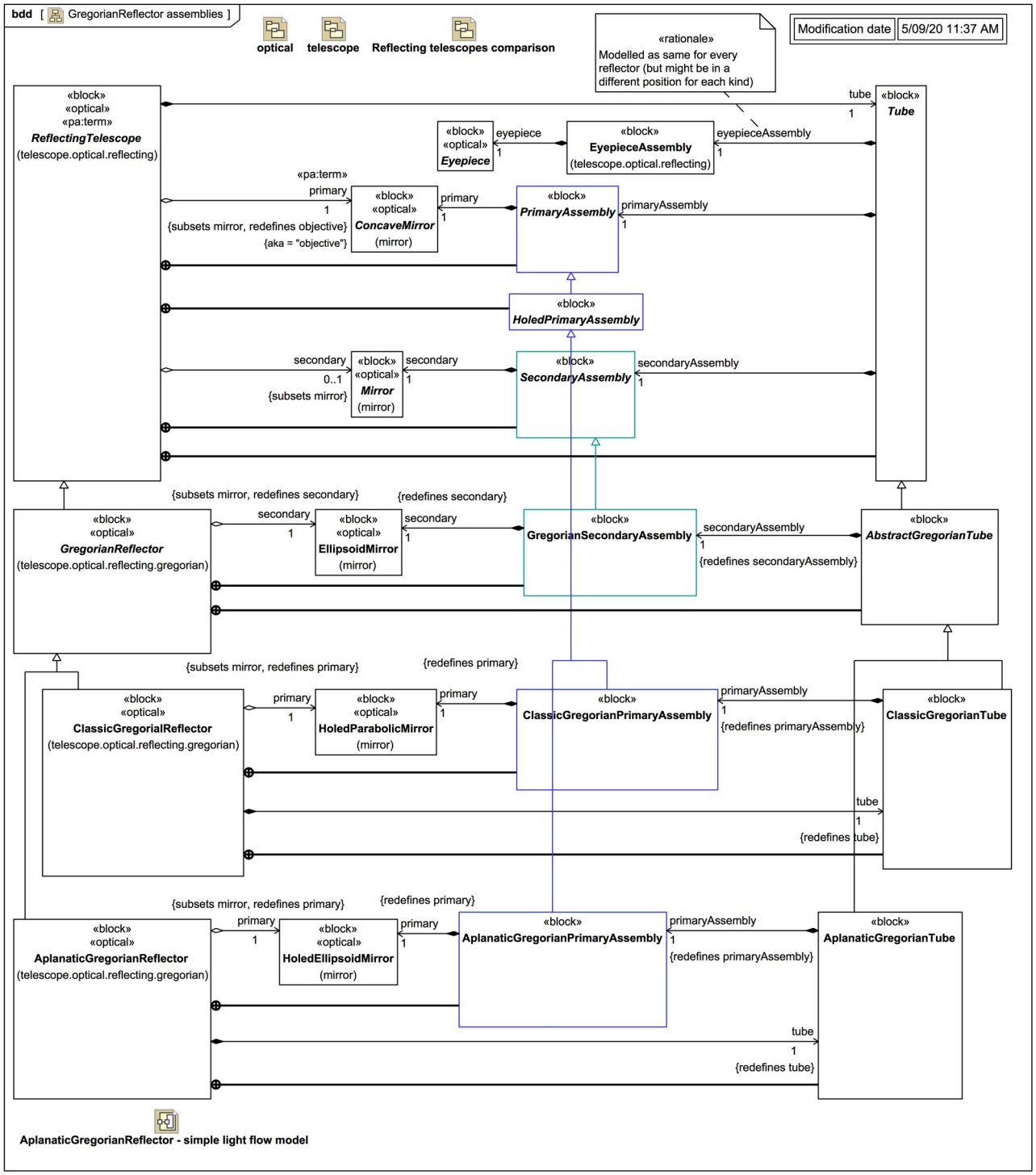

Figure 32: BDD of hierarchy of assemblies for a Gregorian reflector with redefinitions

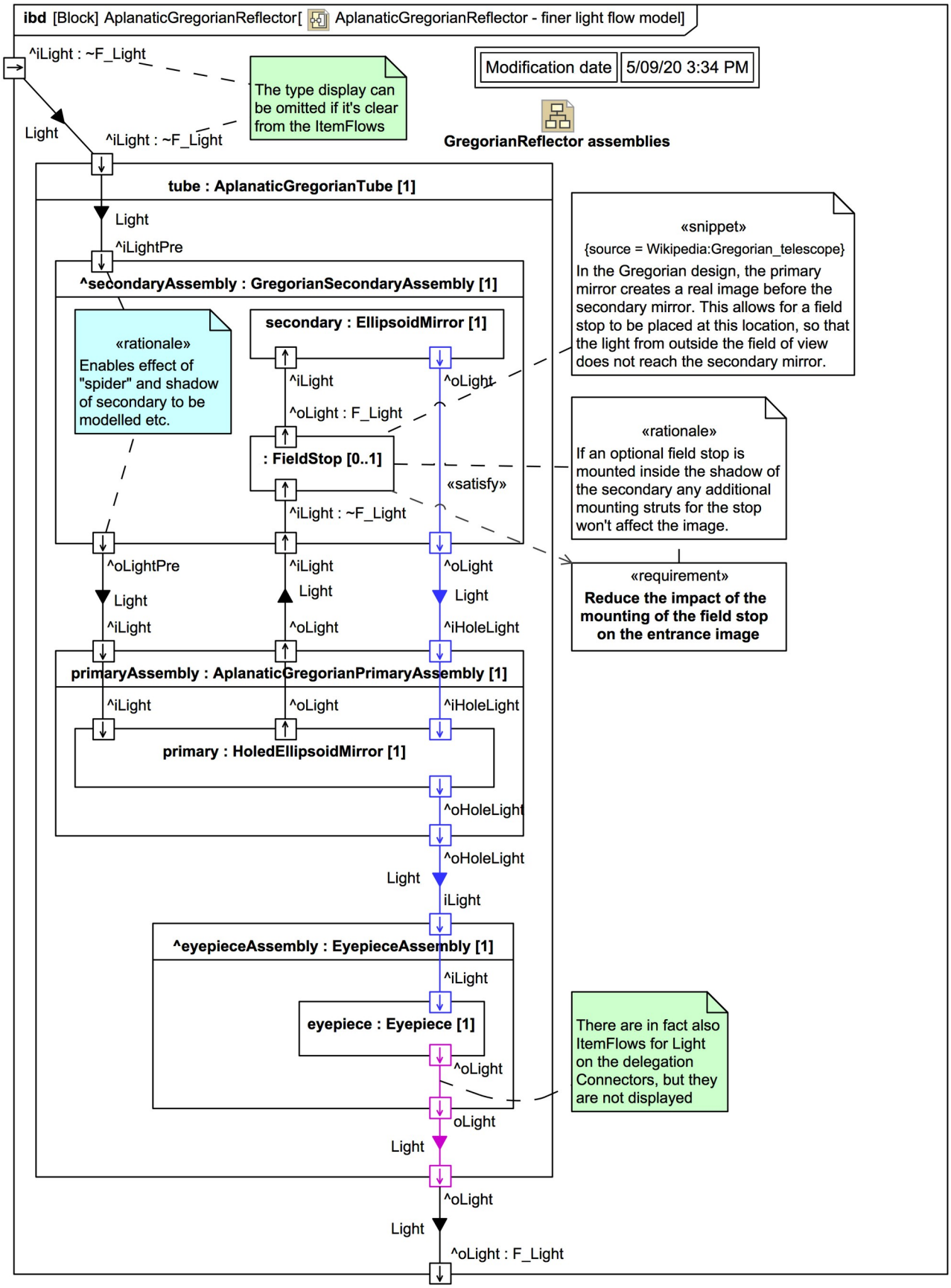

Figure 33: IBD of light flow for an aplanatic Gregorian reflector with supporting Snippet

CASE STUDY: Mars Society Rover Challenge rules

Selected diagrams from a PDF slideset (100 pages) of collated exported SysML diagrams with a preface and some explanations available for public download:

https://www.webel.com.au/node/2599

Domain source document: ‘The Mars Society University Rover Challenge 2020’

http://urc.marssociety.org/home/requirements-guidelines

Visit also: Screencast video of simulation of some Activities and StateMachines:

https://www.webel.com.au/node/3205

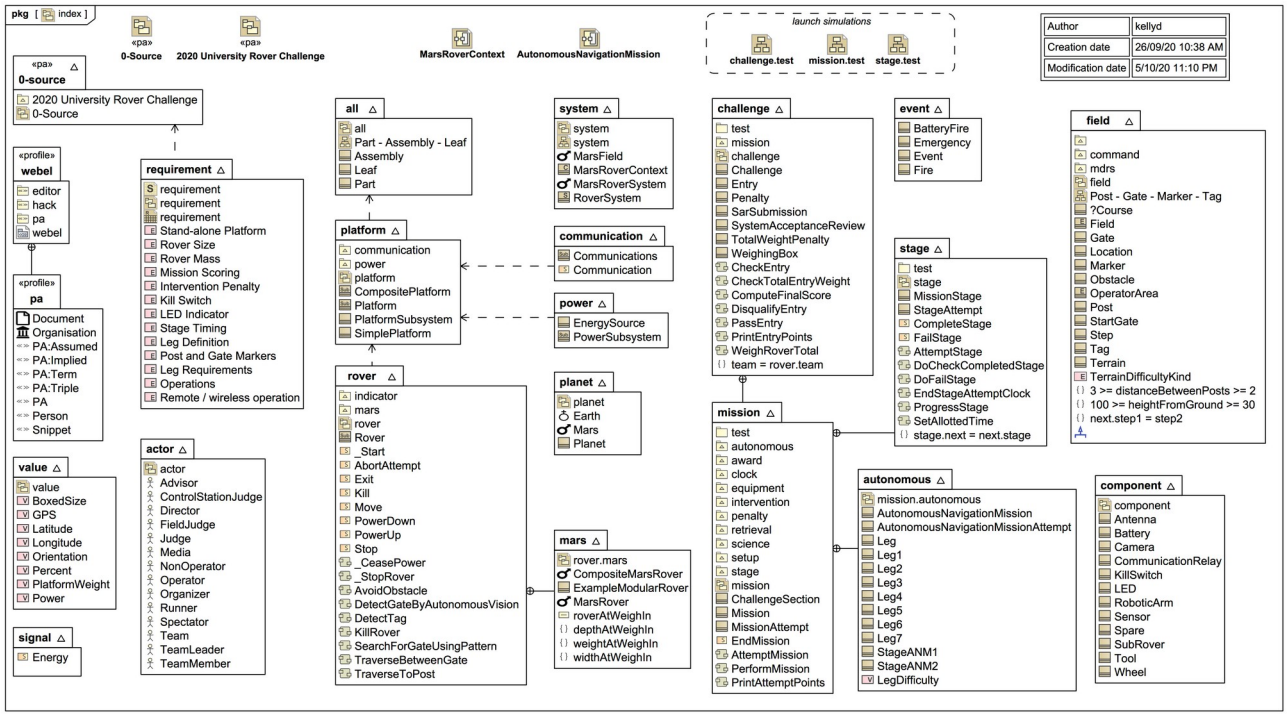

Figure 34: Mars Rover Challenge Package Diagram index with most elicited elements

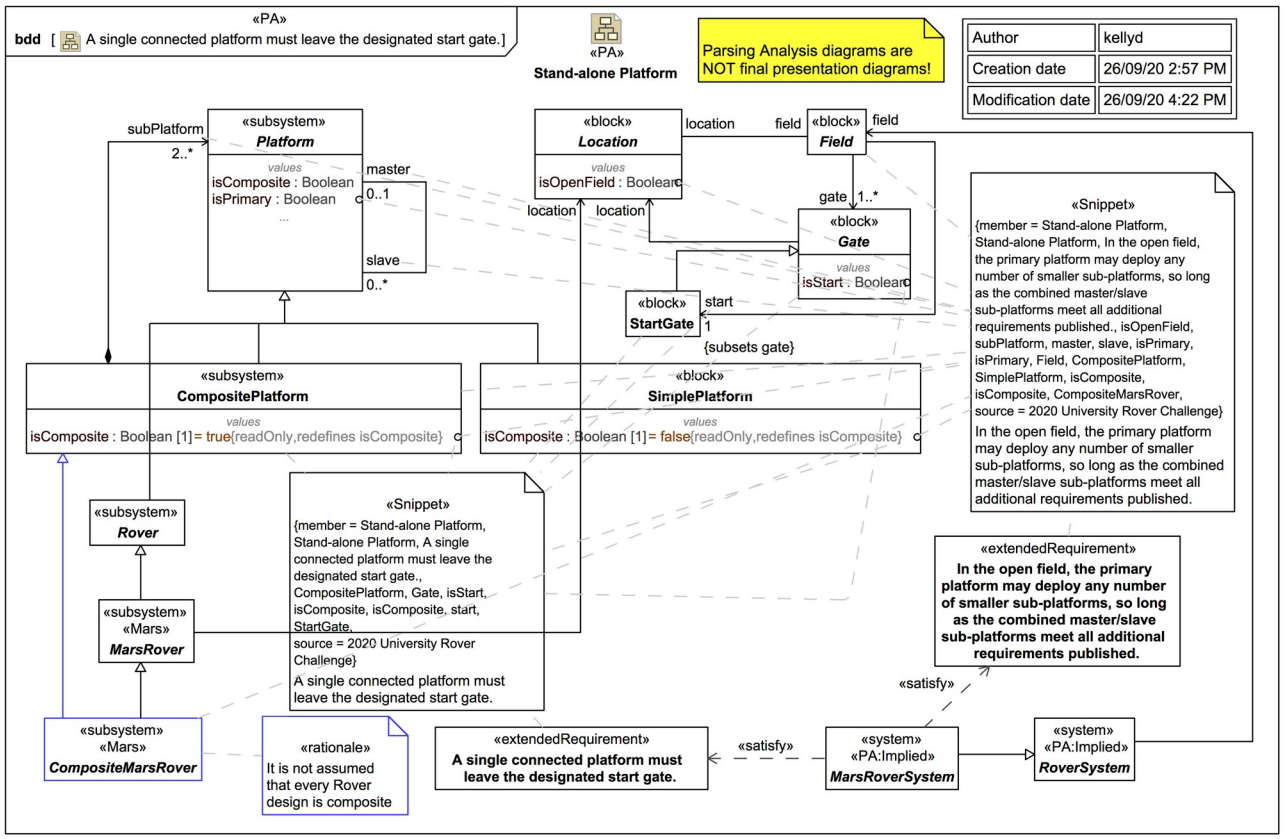

Figure 35: PAD with elicited SysML Requirements and related Blocks and Associations

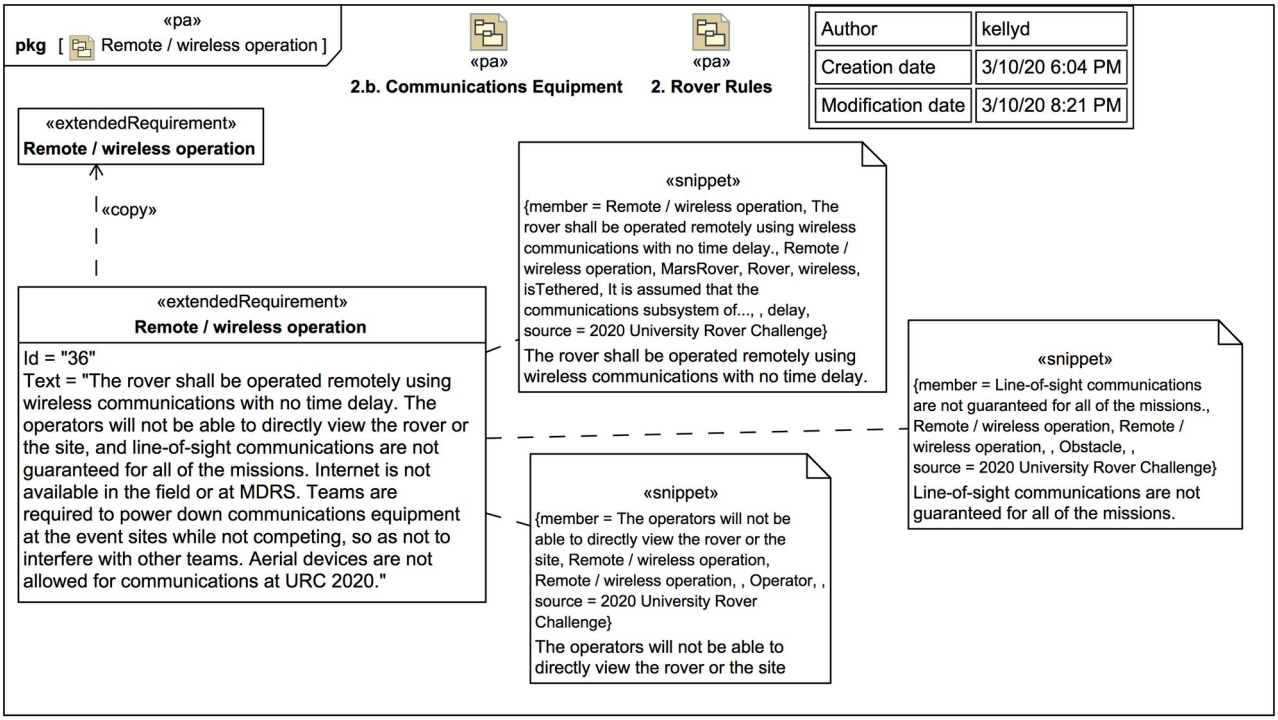

Figure 36: Breakdown of large imported requirement using Snippets with elicited elements

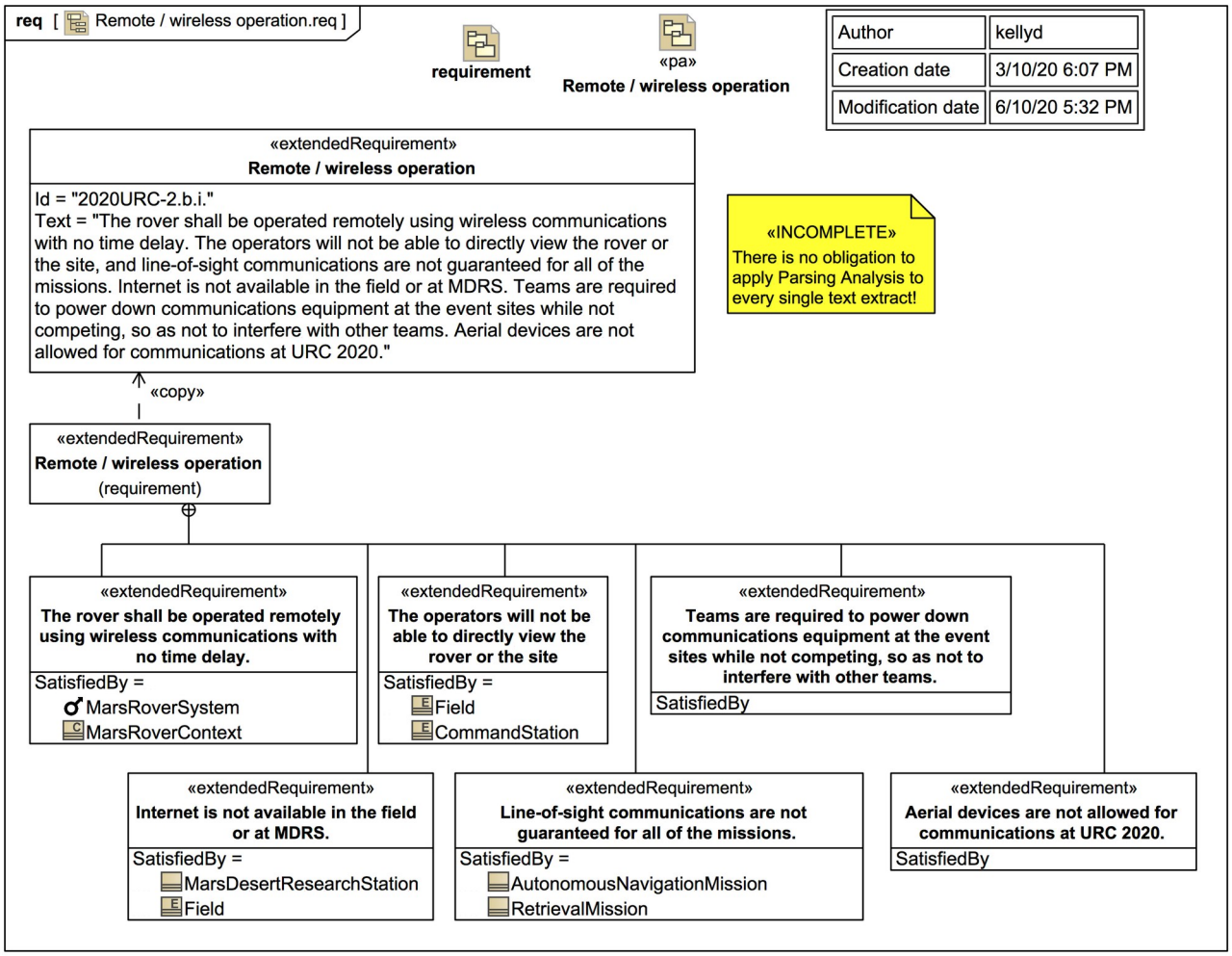

Figure 37: Breakdown of composite Requirement with SatisfiedBy tracking to Blocks

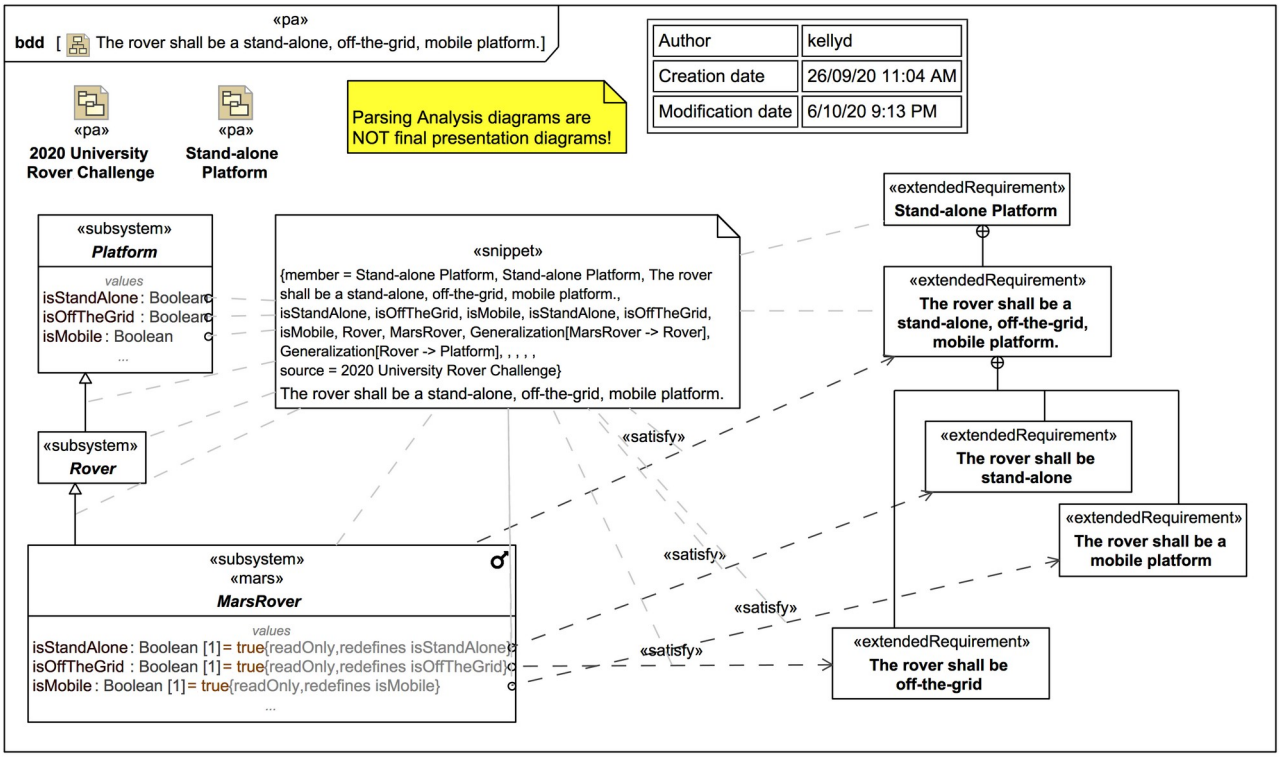

Figure 38: PAD with Snippet for a Requirement and some other elicited model elements

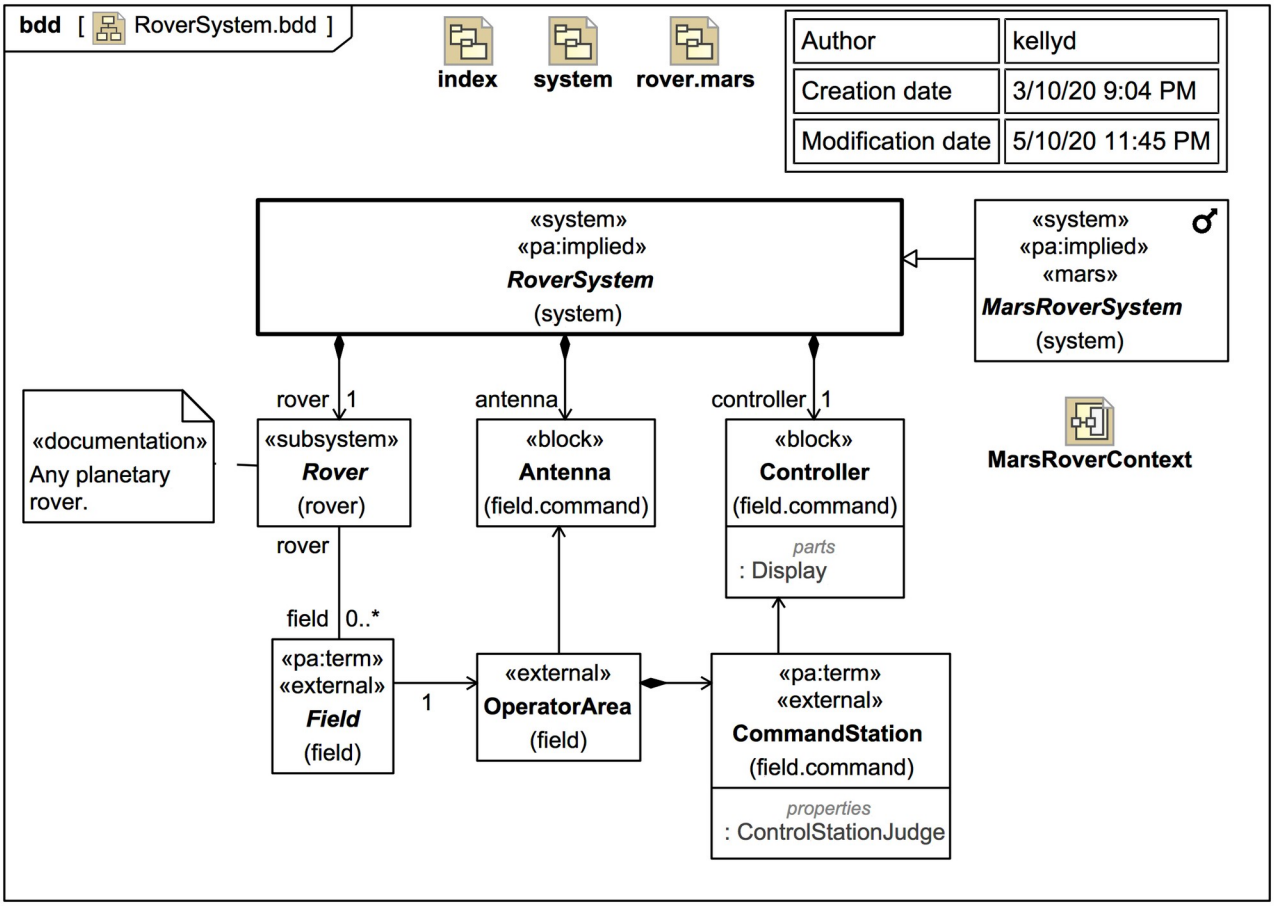

Figure 39: BDD of general RoverSystem (from elements elicited in PADs elsewhere)

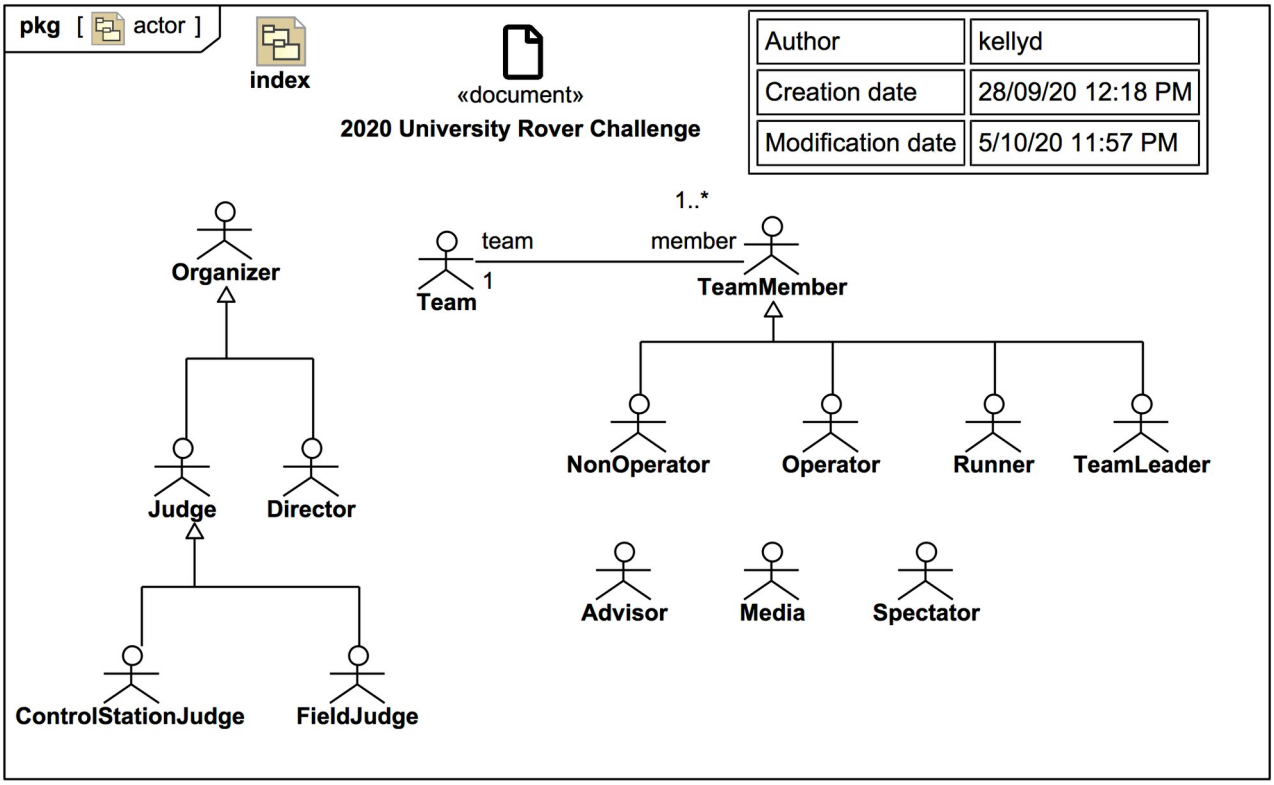

Figure 40: Elicited Actor roles collated in a Package Diagram

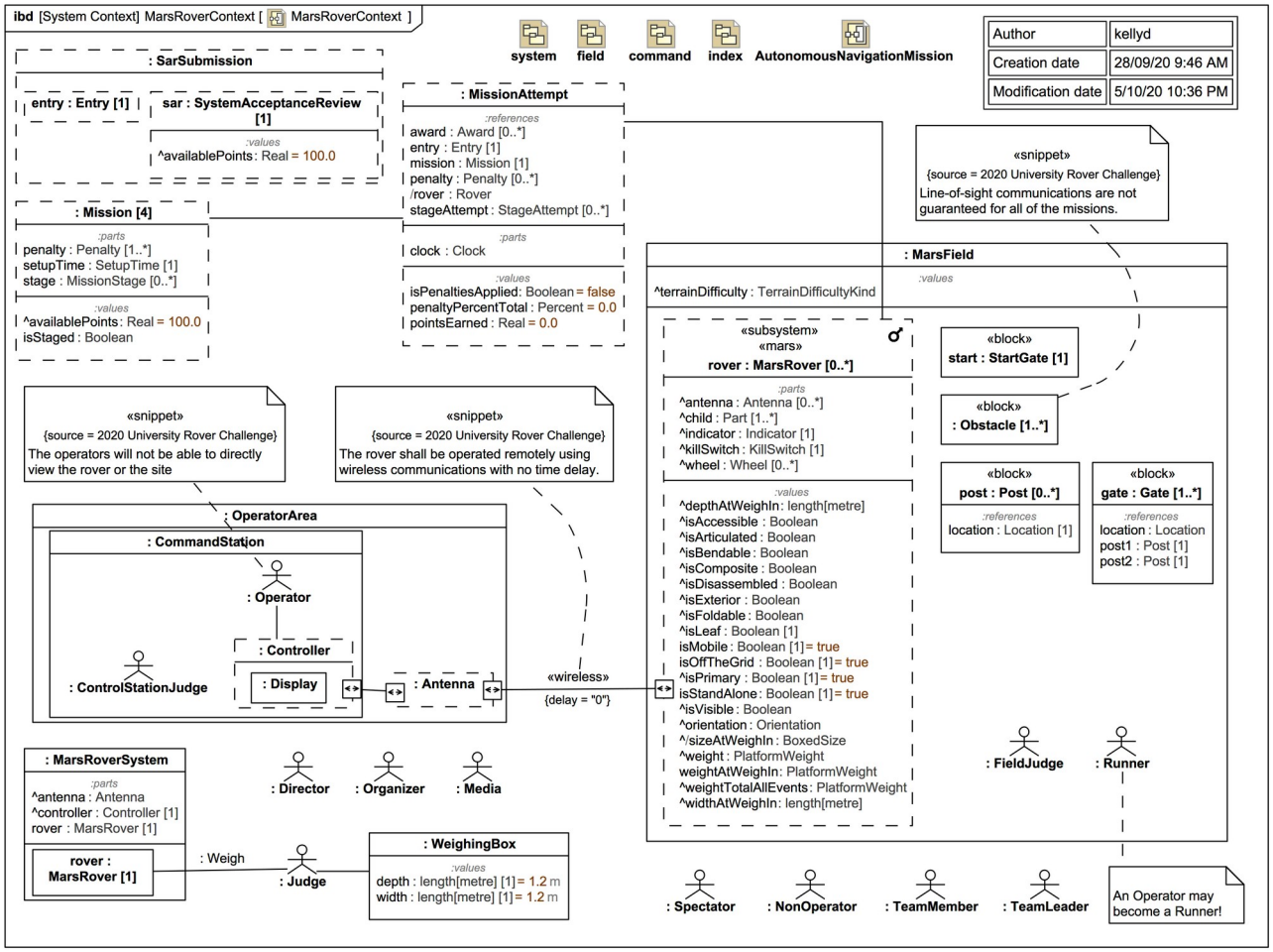

Figure 41: IBD as MarsRoverContext (including Snippets as commentary)

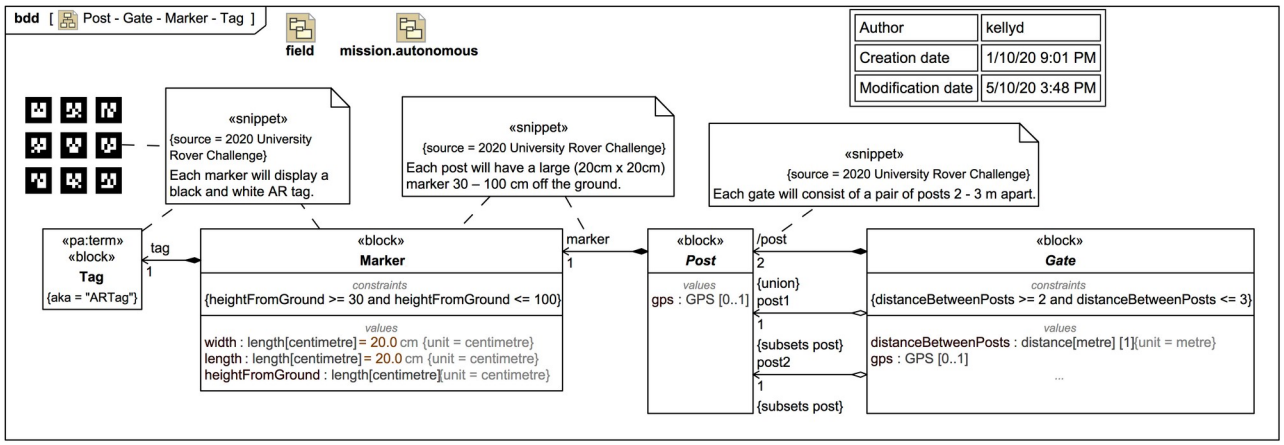

Figure 42: BDD with Constraints elicited from Snippets

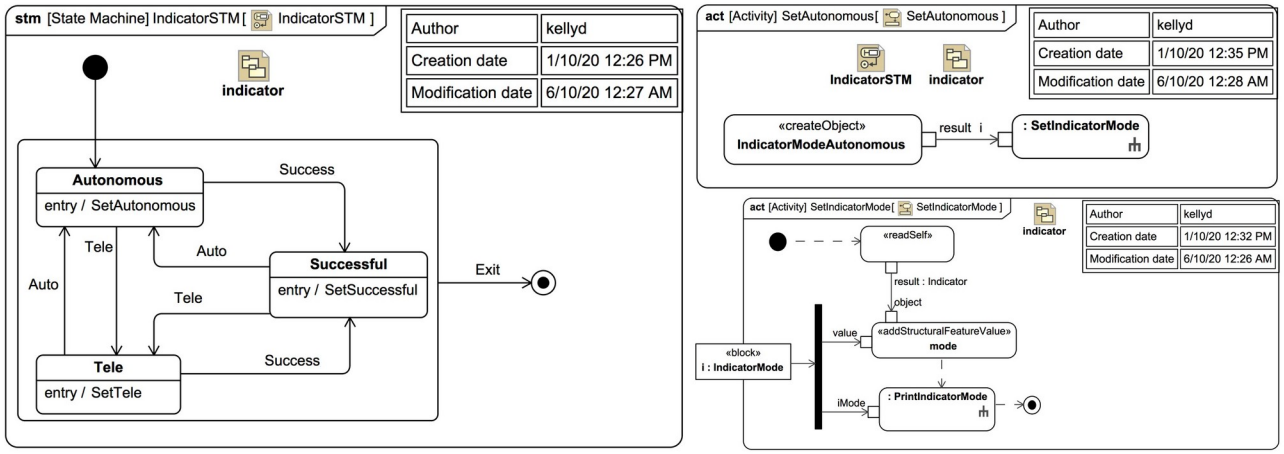

Figure 43: Elicited Behaviors: StateMachines and Activities

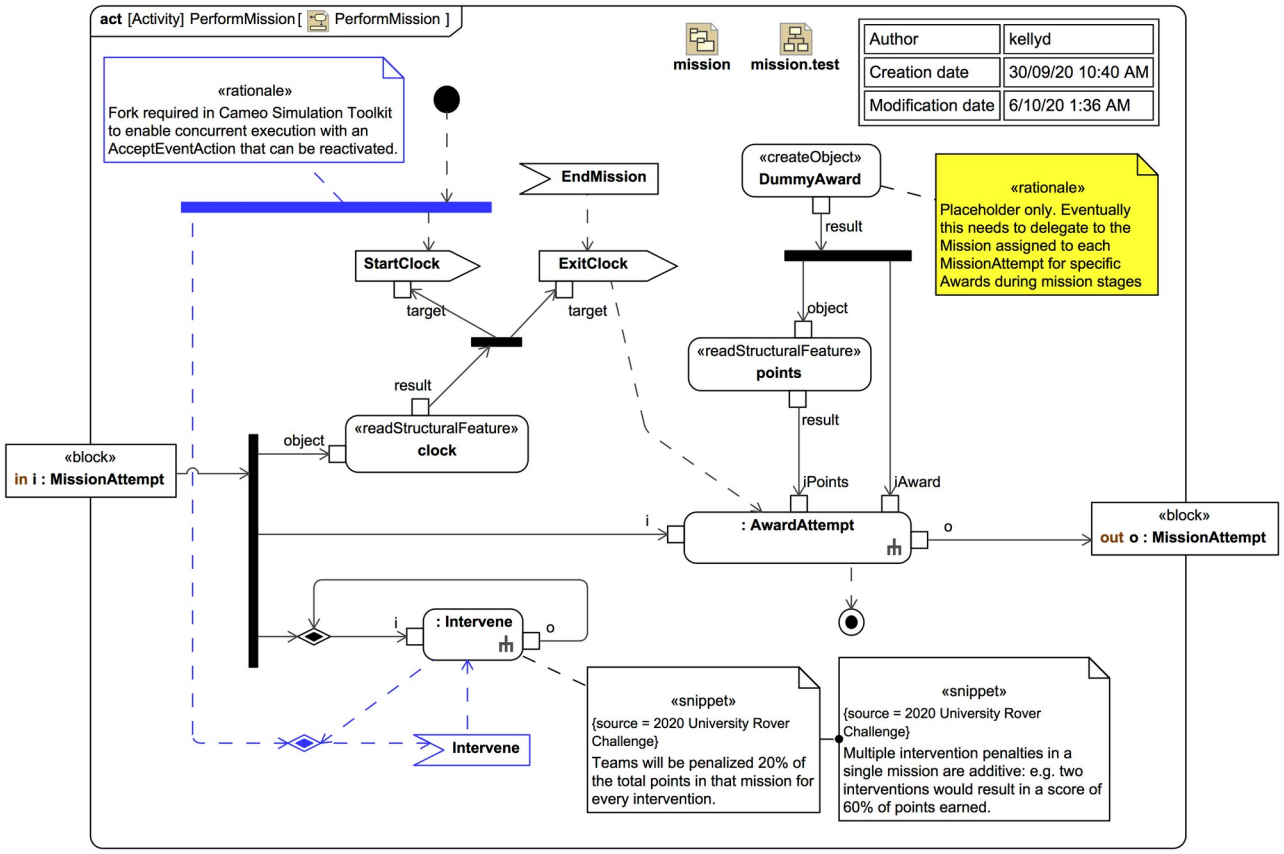

Figure 44: PerformMission Activity for MissionAttempt (with Snippets as commentary)

CASE STUDY: Digital Twin lifecycles vs ANZLIC2019

Selected slides from:

> ‘TRAIL: A SysML Pattern for Digital Twinning:’
>
> https://www.webel.com.au/sysml/trail/twin

Domain source document:

> ‘ANZLIC 2019 – Principles for Spatially Enabled Digital Twins
> of the Built and Natural Environment in Australia’
>
> https://www.anzlic.gov.au/resources/principles-spatially-enabled-digital-twins-built-and-natural-environment-australia

[CAVEAT: Some diagrams in this example series include introduced elements that were not traceably elicited and are not explicitly marked as «pa:implied» or «pa:assumed»]

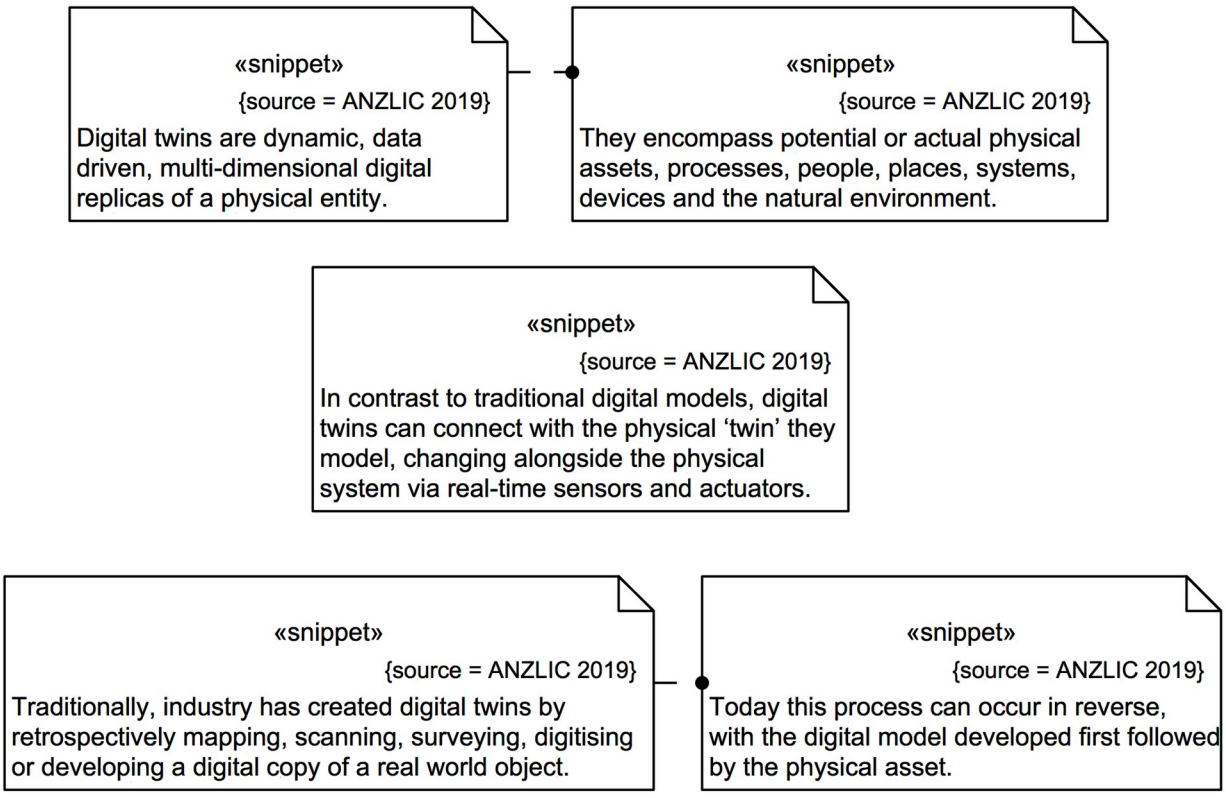

Figure 45: Some Snippets about Digital Twins from ANZLIC2019 (without /member lists)

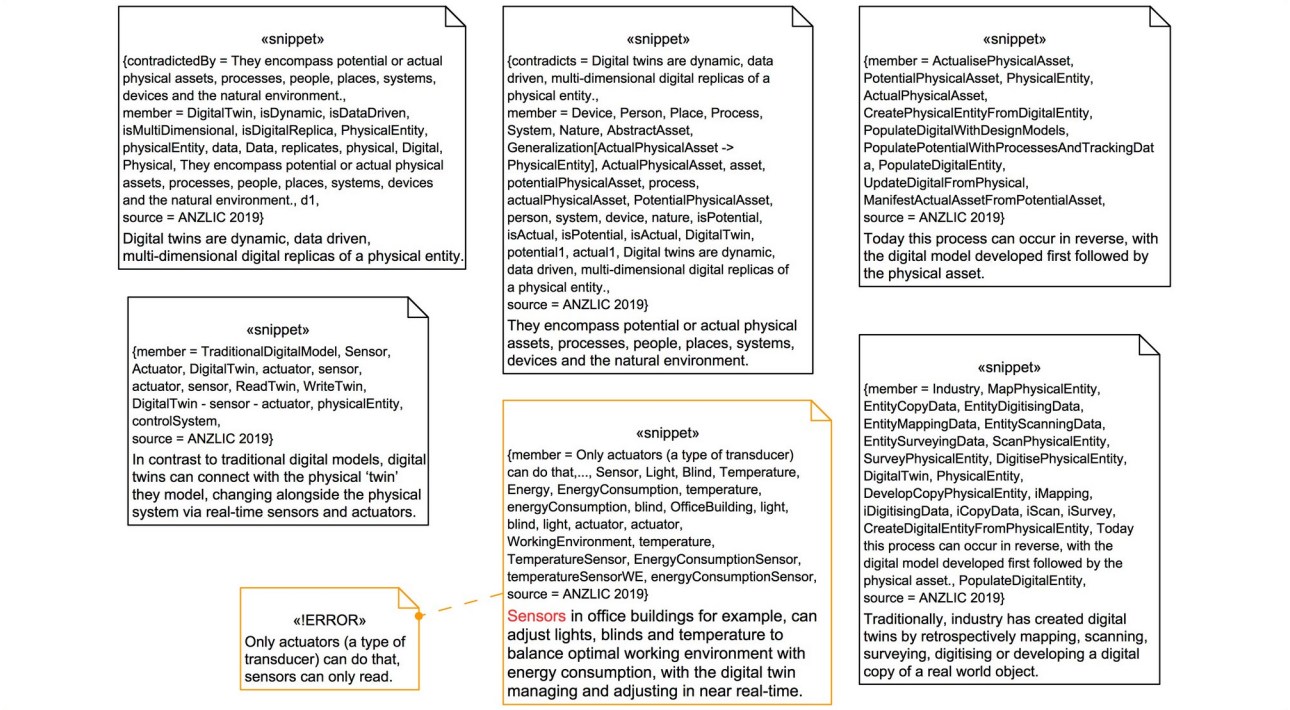

Figure 46: Snippets with elicited model elements /member lists (with error commentary)

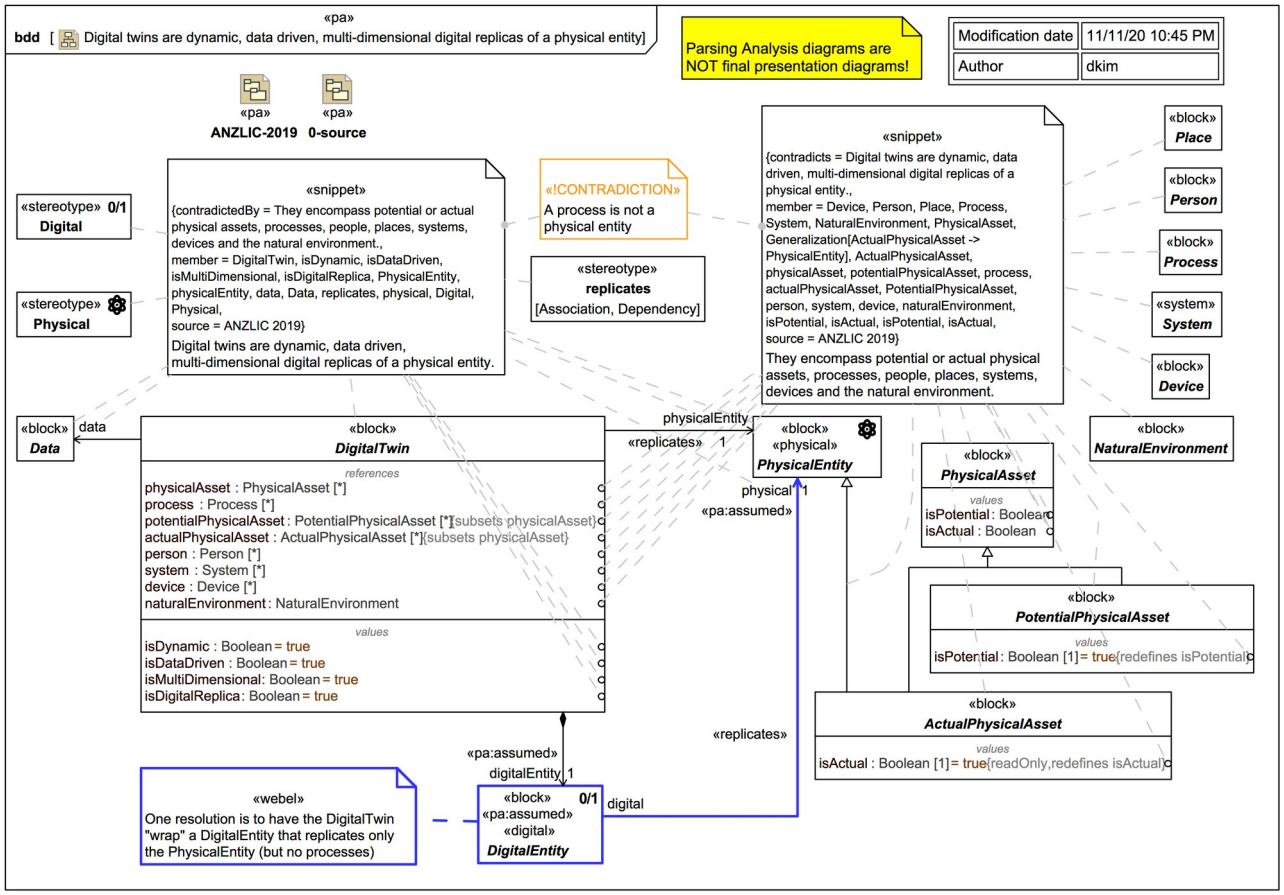

Figure 47: Focus PAD for two Snippets about Digital Twins from ANZLIC2019

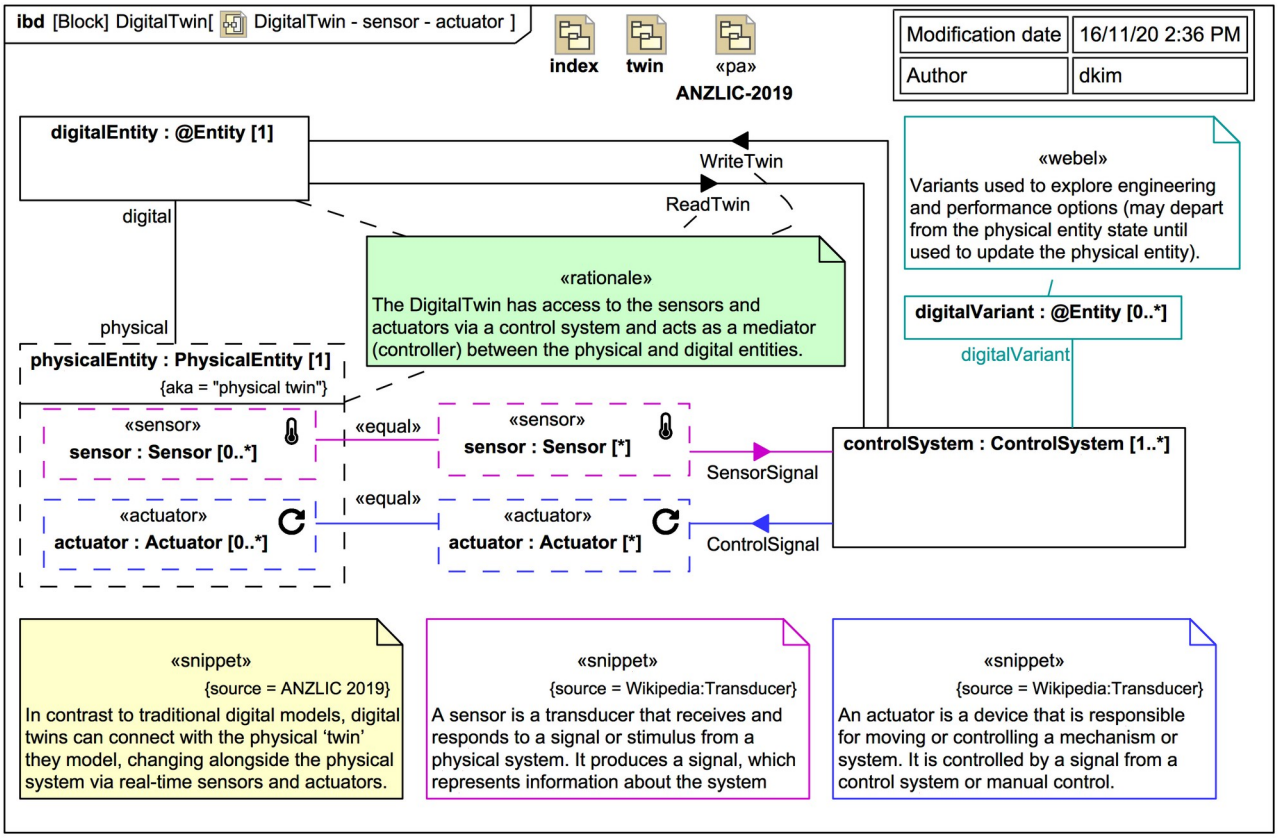

Figure 48: IBD on Digital Twin control loops with Snippets from ANZLIC2019 and Wikipedia

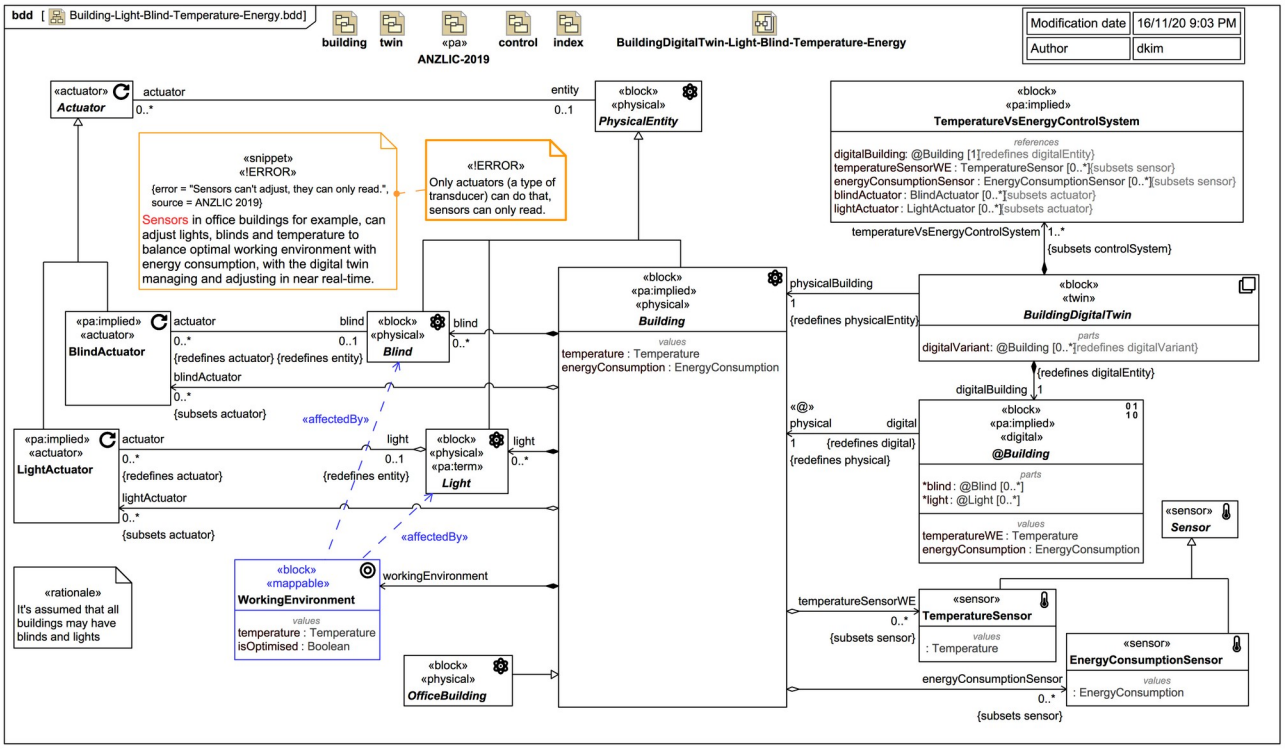

Figure 49: BDD for building DigitalTwins with Snippet from ANZLIC2019 and commentary

Appendix C – Additional examples and resources

The following public online tutorial trails and videos demonstrate realistic example applications of the Webel Parsing Analysis recipe for SysML. (Please note that some of the best examples of substantial real-world applications of the recipe are subject to commercial restrictions.)

TRAIL: Theory and best practices for the Webel Parsing Analysis recipe for SysMLv1.6+

https://www.webel.com.au/sysml/parsing_analysis/theory

TRAIL: Webel SysML Parsing Analysis example: A particle physics taxonomy from Wikipedia

https://www.webel.com.au/sysml/parsing_analysis/particles

TRAIL: Webel SysML Parsing Analysis example: Optical telescopes from Wikipedia: Structure and port-based light flow model

https://www.webel.com.au/sysml/parsing_analysis/telescopes

DEMO: Webel SysML Parsing Analysis: The Mars Society University Rover Challenge 2020 (PDF slides and simulation video)

https://www.webel.com.au/sysml/parsing_analysis/mars_rover_challenge

---

TRAIL: A SysML Pattern for Digital Twinning

https://www.webel.com.au/sysml/trail/twin

Slide presentation: 'The Webel Parsing Analysis recipe for text-driven Model-Based Systems Engineering (MBSE) with Systems Modeling Language (SysML)', Kelly 2021

Integrated Project Engineering Conference (IPEC), 28th May 2021, Australia

https://ipecongress.com.au/
