<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.18\_OnfCoreIm-TemporalExpression.docx' />  
</config>

<context model=’C:\Users\ndavis\git\ONFInfoModel\OnfModel\TemporalExpression.uml' element=’{0}’ importedBundles='gmf;papyrus' searchMetamodels='true'/>

<gendoc><drop/>

Change path substrings above from “{path for output files}\” to your local path for the output files and “{path for CoreModel}\” to your local path for the Core Model. <drop/>

DELETE: Prior to publishing this –gd.docx (including for review), change path substrings above from “C:\Users\ndavis\git\OnfInfoModelOutput\” to “{path for output files}\” and from “C:\Users\ndavis\git\ONFInfoModel\OnfModel\” to “{path for CoreModel}\” <drop/>



Core Information Model (CoreModel)

TR-512.18

Temporal Expression

Version 1.6

January 2024

ONF Document Type: Technical Recommendation

ONF Document Name: Core Information Model version 1.6

**Disclaimer**

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Any marks and brands contained herein are the property of their respective owners.

Open Networking Foundation  
1000 El Camino Real, Suite 100, Menlo Park, CA 94025  
[www.opennetworking.org](http://www.opennetworking.org)

©2024 Open Networking Foundation. All rights reserved.

Open Networking Foundation, the ONF symbol, and OpenFlow are registered trademarks of the Open Networking Foundation, in the United States and/or in other countries. All other brands, products, or service names are or may be trademarks or service marks of, and are used to identify, products or services of their respective owners.

**Important note**

This Technical Recommendations has been approved by the Project TST, but has not been approved by the ONF board.  This Technical Recommendation is an update to a previously released TR specification, but it has been approved under the ONF publishing guidelines for ‘Informational’ publications that allow Project technical steering teams (TSTs) to authorize publication of Informational documents.  The designation of ‘-info’ at the end of the document ID also reflects that the project team (not the ONF board) approved this TR.

Finalizing this document once generated… delete this text prior to publication:

* Replace “{{..}}” with square brackets (which trip up Gendoc)
* Select text in document from beginning of table of contents (first line) to end of document
  + Click menu item “Update Field” (on this large block of text)
    - if “Update Table…” dialogue appears select “Update entire table”
  + Repeat “update fields” 2 more times (on the same large block of text)
    - if “Update Table…” dialogue appears select “Update entire table”
* Remove reviewer comment

Note that the table of contents and figures need to be updated several times as the table length changes the page numbering and the cross references will need to be re-updated.

Table of Contents

[Disclaimer 2](#_Toc456706149)

[Open Networking Foundation 2](#_Toc456706150)

[Document History 4](#_Toc456706151)

[1 Introduction 4](#_Toc456706152)

[2 Model Overview 5](#_Toc456706153)

[2.1 Model Focuses 5](#_Toc456706154)

[2.1.1 Core Network Model Overview 5](#_Toc456706155)

[2.1.2 Core Foundation Model 6](#_Toc456706156)

[2.1.3 Layered Topology 8](#_Toc456706157)

[2.2 Other documents 10](#_Toc456706158)

[2.3 Supporting Guidelines 10](#_Toc456706159)

[2.4 Key reference material 10](#_Toc456706160)

[3 Summary of main changes between version 1.1 and 1.2 11](#_Toc456706161)

[4 References 11](#_Toc456706162)

[5 Definitions 12](#_Toc456706163)

[5.1 Terms defined elsewhere 12](#_Toc456706164)

[5.2 Terms defined in this TR 12](#_Toc456706165)

[6 Abbreviations and acronyms 13](#_Toc456706166)

[7 Conventions 15](#_Toc456706167)

[7.1 Lifecycle Stereotypes 15](#_Toc456706168)

[7.2 Diagram Keys 15](#_Toc456706169)

[8 Future CoreModel areas 17](#_Toc456706170)

[9 Terminology Translation table 18](#_Toc456706171)

[10 Back matter 19](#_Toc456706172)

[10.1 Editors 19](#_Toc456706173)

[10.2 Contributors 19](#_Toc456706174)

List of Figures

[Figure 1-1 Methodology of IM and DS Development 6](#_Toc430780029)

# List of Tables

[**Table 1: periodDurationInDays property interaction** 9](#_Toc148004173)

Document History

| **Version** | **Date** | **Description of Change** |
| --- | --- | --- |
| 1.0 | January 2024 | Initial Version |

# Introduction to the document suite

This document is an addendum to the TR-512 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## References

For a full list of references see [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## Definitions

For a full list of definition see [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## Conventions

See [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf) for an explanation of:

* UML conventions
* Lifecycle Stereotypes
* Diagram symbol set

## Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%) or open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

## Understanding the figures

Figures showing fragments of the model using standard UML symbols as well as figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf) for diagram symbol sets). All UML diagrams depict a subset of the relationships between the classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some UML diagrams also show further details of the individual classes, such as their attributes and the data types used by the attributes.

# Introduction to Temporal Expression

The focus of this document is on a model of time periods. Application of this model to the other Core Model entities will allow for representation of the history of actual lifecycle changes and representation of future possibility/intention. For the latter some of the multiplicities in the model may need to be refined.

There are many models/representations of temporal expression. These have been explored to inform the development of the model described in this document.

A data dictionary that sets out the details of all classes, data types and attributes is also provided ([TR-512.DD](TR-512.DD_OnfCoreIm-DataDictionary.pdf)).

# Temporal Expression model detail

The model:

* Represents time using a combination of absolute calendar statements, statements of periodicity, statements of duration, statements of iteration and statements of phase.
* Provides temporal expressions in terms of combinations of minutes, hours, days, weeks, months and years.
* Accounts for time zone and DST
* Is formed around a TemporalExpression class that is built from a union of TeElements where each TeElement is formed by an intersection of definitions where each definition is applied via «specify» abstraction (such that the definition augments the TeElement).
* Supports the combination of expressions where, in general, specifications of the same sort combine by union and of different sort combine by intersection. Following the model explanation there is a section of examples to illustrate this.
* Enables a TemporalExpression to incorporate other previously defined TemporalExpressions where the incorporation method can be union, intersection or intersection with the complement of the incorporated TemporalExpression
* Has been designed such that all combinations give a defined output (i.e., no combinations are “illegal”), although some may result in no period

The diagram below shows the model.

[for(p:Package|Package.allInstances())]<drop/>

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘TemporalExpression-Overview’, ’Temporal Expression Model’)/]

[/for]<drop/>

The diagram below shows the usage of data types in the model and exposes the detail of the model.

[for(p:Package|Package.allInstances())]<drop/>

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘TemporalExpression-DataTypes’, ’Temporal Expression Data Types’)/]

[/for]<drop/>

The following sections provide definitions of the structures depicted in the diagrams above.

## Temporal Expression Class Model

This section provides the temporal expression classes.

[for (cl:Class | Class.allInstances()->sortedBy(name))]<drop/>

[if (cl.qualifiedName.contains(‘TemporalExpression’))]<drop/>

### [cl.name/]

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[cl.insertClass(cl.name,’TemporalExpression’)/]

Inserts the attributes of the class <drop/>   
[cl.insertAttributeTableBrief ()/]

[/if]<drop/>

[/for]<drop/>

## Temporal Expression Data Types

This section provides the temporal expression data types.

[for (dt:DataType | DataType.allInstances()->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(DataType)]<drop/>

[if (dt.qualifiedName.contains(’TemporalExpression’))]<drop/>

### [dt.name/]

Inserts the details of the data type in first quotes from the package in second quotes <drop/>

[dt.insertDataType(dt.name,’TemporalExpression’)/]

Inserts the attributes of the data type <drop/>

[dt.insertDataTypeAttributeTableBrief ()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

## Enumeration Types

[for (dt:DataType | DataType.allInstances()->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(Enumeration)]<drop/>

[if (dt.qualifiedName.contains(‘TemporalExpression’))]<drop/>

### [dt.name/]

Inserts the details of the enumeration in first quotes from the package in second quotes <drop/>

[dt.insertEnums ()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Further detail

Some of the structures use in the model have a somewhat complex interaction. These are discussed in the sections below.

### Period duration

The periodDurationInDays has several interacting properties. The interaction is explained below.

**Table 1: periodDurationInDays property interaction**

| startTime explicit | endTime explicit | periodDuration explicit | daySpan explicit | Meaning |
| --- | --- | --- | --- | --- |
| Don’t care | Don’t care | None | 0 | No period |
| None | None | None | None | Starts at beginning of listed day and ends at end of listed day. |
| None | None | None | 1 | Starts at beginning of listed day and ends at end of listed day |
| None | None | None | 2+ | Starts at beginning of listed day and ends at end of day 1+ days later |
| Specific | None | None | 1..\* | Starts at specific time in day listed and ends at end of day as per daySpan |
| None | Any specific | None | 1..\* | Starts at start of day listed and ends at specific time as per daySpan. |
| Before end | After start | None | 1 | Simple duration within each day of week listed |
| After end | Before start | None | 1 | Starts at beginning of day of week ends at end time starts again at start time and ends at end of day |
| Specific | Specific | None | 2+ | Starts at start time in day of week listed ends 2+ days layer at end time. |
| Specific | Don’t care | Specific | Don’t care | Starts at start time in day of week listed and stops after duration. |
| None | Specific | Specific | Don’t care | Starts duration before end time in day listed and stops at end time in day listed. |
| None | None | Specific | Don’t care | Starts at start of day listed and ends after duration |

# Examples of Combination

The followings sections illustrate usage of the structure for an arbitrary set of definitions.

## Union within a property

Consider the following examples:

* dayOfWeekUnion (in DaysOfWeekInMonthInYearIntersectionTe): if the list has MONDAY and TUESDAY then this means for both Monday and Tuesday.
* periodUnion (in BoundingPeriodIntersectionTe): If there are two members of the list:
  + Start: 1 January 2023 00:00:00 (Sunday) End: 31 March 2023 23:59:59 (Friday)
  + Start 1 June 2023 00:01:00 (Thursday) and End: 30 June 2023 22:59:59 (Friday)

Then the expression is “active” for both periods, but clearly not between 1 April 2023 00:00:00 to 1 June 2023 00:00:59

## Intersection between properties in augments

Take the properties above. If these were applied to a TeElement, the intersection results in every Monday and Tuesday between 1 January 2023 00:00:00 (Sunday) End: 31 March 2023 23:59:59 as well as between 1 June 2023 00:01:00 (Thursday) 30 June and 2023 22:59:59 (Friday).

So, this would include Monday 2 January, Tuesday 3 January, Monday 9 January etc. as well as Monday 5 June, Tuesday 6 June etc.

Now consider properties from PhasedPeriodIntersectionTe. If hourPhasing is set to 2 and periodDurationInMinutes is set to 15, the intersection of these two means the first 15 minutes every other hour throughout each continuous period starting at the beginning of the period[[1]](#footnote-1).

If this is also applied to the TeElement this would result in the TeElement being valid for Monday 1 January 00:00:00 to 00:14:59 (last moment of), then 02:00:00 to 02:14:59 through Monday and similarly through Tuesday as that is part of the continuous period. This would be the same for each following Monday/Tuesday pair.

All other properties intersect in a similar way.

## Union of TeElements ino a TemporalExpression

Consider two TeElements, one as defined above and another that was built similarly but was for Tuesday and Wednesday from 1 January 2023 00:00:00 to 31 January 2023 23:59:59 with a PhasedPeriodIntersectionTe of every hour for 30 minutes. If these two TeElements are applied to the same TemporalExpression, TeA, via TeHasUnionOfPeerElements association, this would result in Monday being exactly as previously defined in the previous section.

But the Tuesdays in January would be active for the first 30 minutes of every hour as would the Wednesdays. From the end of January only Monday and Tuesday would be active for the first 15 minutes of every other hour as before.

Again, all other properties can be treated in this way.

## Applying an IncorporatedTe

Consider a further TemporalExpression (TeB) that is defined with one TeElement and no BoundingPeriodIntersection so that it applies for all time. If that TeElement has (in DaysOfWeekInMonthInYearIntersectionTe):

* specificMonthUnion of JANUARY
* dayOfWeekUnion indicating TUESDAY
* namedDayPositionInMonthUnion indicating SECOND

This TemporalExpression (TeB) is active for the whole of the second Tuesday (i.e., the first moment of 00:00:00 to the last moment of 23:59:59) in January (for any year).

Consider another TemporalExpression (TeC) that has a TeElement that is defined for all time (i.e., has no constraints), but also has two IncorporatedTes.

The first has incorporationMethod set to INTERSECTION and refers to TeA and the second has incorporationMethod set to INTERSECT\_COMPLEMENT and refers to TeB.

The complement of TeB is active for all time other than the whole of the second Tuesday in January (when it is inactive). When the intersection of this with TeA is formed this results in the same definition as the raw TeA other than for the second Tuesday in January (10 January) when the result is inactive.

So TeC would be active as defined for TeA for each Tuesday in January other than 10 January when it would be inactive.

# Using the temporal model

## Application to elements of the domain model

Temporal variations can occur at any level of a model:

* An instance may be created at a particular time then deleted at another
* An instance may become visible at a particular time then vanish at another time and then reappear later etc.
* A property of an instance can take a new value at a particular time.
* A property can appear in an instance at a particular time
* A member of a list can change position at a particular time
* Etc.

Distilling from the above, there appear two distinct considerations:

* Presence/existence (at a place)
* Value

Value at a time can be stated in terms of solely the value that has changed in the context of a sparce positional structure or in the context of other unchanging close values. For example, the value of one element of a list may change, this could be expressed in isolation as a new value against the list position or could be expressed as part of the whole list of values (in a sparce positional structure).

Presence is more complex as it requires:

* create and remove (and potentially hide and expose) of an identified unit, including structure and creation of an identifier, at a place in an existing structure
* move and replication of an identified unit (and potentially change of identifier of a unit, e.g., where the identification is the position and there is a requirement to change the position).

As a consequence, in the most complex cases, the temporal model statements will apply at many/all levels of the model. It may also be necessary to separate presence from value for all levels

The figure below shows a generalized representation of the application/positioning of the temporal expression.

It is expected that at each application there will be a single referenced temporal expression that will, as necessary, incorporate other temporal expressions to form the appropriate complexity of temporal expression.

There may be several distinct changes over time such that there are several distinct representations of the structure/values, each with its own temporal expression.

There may be a statement of value/structure that applies to any time where stated temporal expressions are not true.

It appears that there is a need for an idempotent add (i.e., “add if not already present” as opposed to “add another”) and corresponding remove.

There is a clear challenge adding and removing by list position as opposed to invariant id as there may be race conditions with competing adds and removes.

The expression could be in terms of presence/absence after a specific time as opposed to add/remove at a moment in time.

The model could be extended with considerations of time to achieve and an indication of jeopardy if a temporal expectation cannot be met.

* E.g., If a connectivity-service is not realized for a period then is to be realized for a period and the time to realize is longer than the time left before the service should be realized.
* E.g., if a future realization (with resources already acquired) is failed (operational state disabled) and the time to repair is longer than the time remaining before the schedule becomes active.



Figure 6-1 TemporalExpresion being applied to an entity – Pattern



Figure 6-1 TemporalExpresion being applied to an entity – Rough Expansion

## Application to deal with plan deviations and plan alternatives

The temporal model is applied to the modeling of both actual history and possible future. Considering possible future, there are several degrees:

* Committed future
* Temporary deviation from commitment
* Alternative futures

The following subsections consider each of the above.

### Committed future

This is a single thread of time where, at any particular point in time, there is no conflict between resource usages in the same way there is no conflict in usage in live solution, i.e., each unit of resource can be used only once.

Unlike the live solution, some resources may not have a defined state (distinct from known state, the resource must be in a state even if it is not known). The representation of committed future provides a set of constraints that restrict what is intended.

The committed future is a statement of a progression of outcomes that is a realization solution to the outcomes requested, usually with more detail that was provided in the request. It is possible to:

* delay the specification of resource configuration state to the moment when it is required
* plan explicit resource configuration state ahead of time

In both cases there is a chance that the resources will not be available at the moment they are required to support the desired outcome, however, in many cases an appropriately detailed plan that is continually evaluated for achievability will provide a greater likelihood of success than a delayed specification where resources may just not be available. Clearly, continual evaluation comes at a cost.

### Temporary deviation from commitment

This can apply:

* “Now”
* From some point in the future

The deviation may have a clear end date/time or may have uncertainty in its end (and possibly even its start). The deviation may be fully defined or may have temporal variation with uncertainty etc. Clearly, there may be a deviation from the deviation etc.

Any deviation will have some associated definition of progression from non-deviated state to deviated state and also back to non-deviated state.

### Alternative futures

Beyond any commitment, there may be some knowledge of potential futures and there may be several potential futures each of which may only be partially defined and each of which will probably have temporal variation with uncertainty.

As time progresses, some part of the alternative futures will become a commitment “collapsing” much of the uncertainty into a specific set of details. Eventually that committed future will become the present (and then the past) and will be fully resolved into real instances.

# Specific examples

This section sets out examples of use of the temporal expression for common everyday activities (as opposed to telecoms scheduling) to help clarify the model usage in a familiar context.

## Garden Waste Collection

This example uses several of the structures to show how a regular waste collection activity may be suspended for a week due to a public holiday. It is likely that the waste collection would be reschedules in the week of the holiday. This is not detailed in the example, but the opportunity is highlighted.

The figure below sets an instance model.

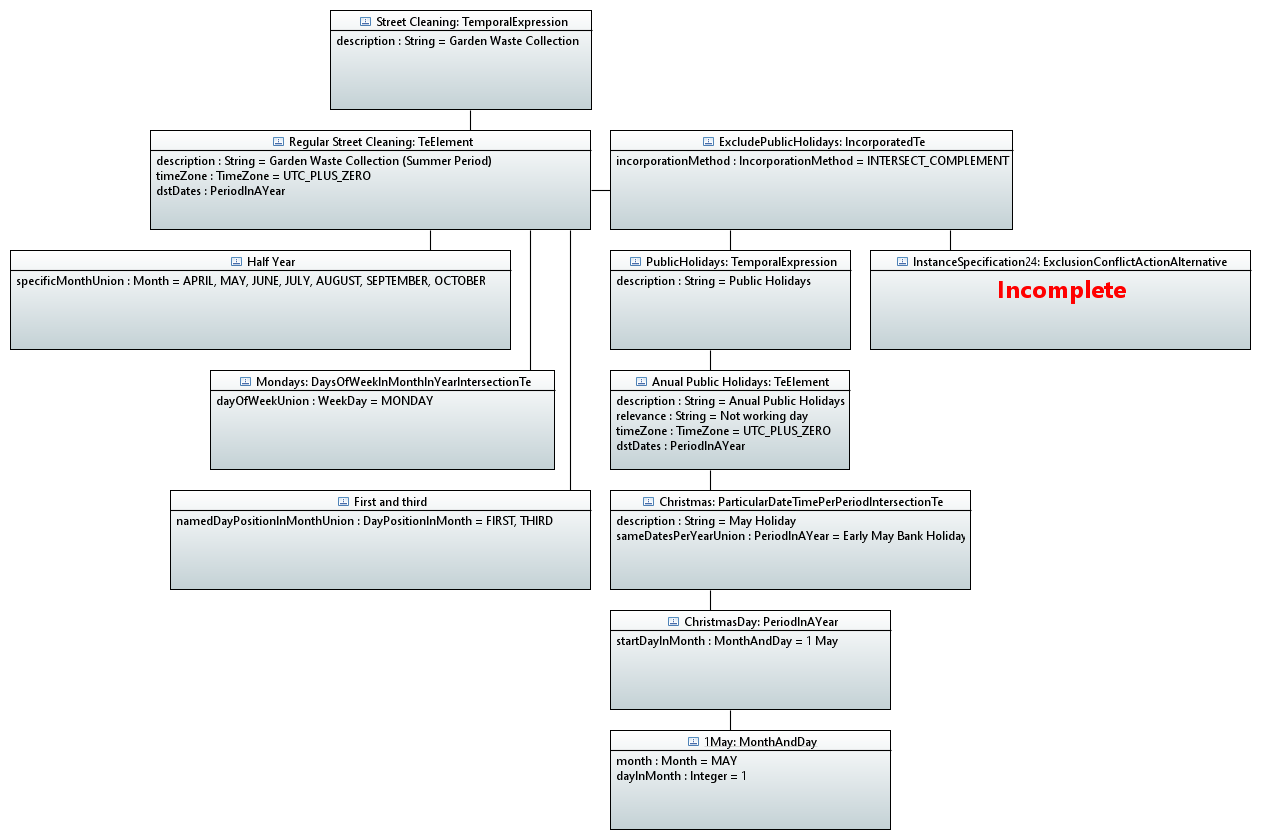


Figure 6-1 Garden waste collection example

# Further work

Further work will be carried out in this area in a subsequent release of the model.

**End of Document**

</gendoc><drop/>

To take latest template: <drop/>

* delete text from “Template version…” to end of file <drop/>
* insert a line in “Normal” style<drop/>
* insert text (Insert 🡪 Object 🡪 Text from File… (alt njf)) from: <drop/>

TR-512.GT\_OnfCoreIm-CommonGendocTemplate-Fragments.docx <drop/>

Template version 0.0.11 1 June 2018 <drop/>

# Fragment: Insert class <drop/>

<fragment name=’insertClass’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>  
<arg name=’className’ type=’String’/><drop/>  
<arg name=’packageName’ type=’String’/><drop/>  
[if (not cl.qualifiedName.contains(packageName))]<drop/>  
[else] <drop/>  
[if(cl.name.contains(className))]<drop/>

Qualified Name: [cl.qualifiedName/]

[if cl.ownedComment->notEmpty()]<drop/>

[for (co:Comment | cl.ownedComment)] <drop/>

<dropEmpty>[cleanAndFormat(co.\_body.clean())/]</dropEmpty>

[/for]<drop/>

[else]To be provided

[/if]<drop/>

[if (cl.isAbstract)]<drop/>

This class is abstract.

[/if]<drop/>

[if (cl.oclAsType(uml::Class).general ->notEmpty())]<drop/>

Inherits properties from:

[for (gen:Class | cl.oclAsType(uml::Class).general)]<drop/>

* [gen.name/]

[/for]<drop/>

[/if]<drop/>

[for (st:Stereotype | cl.getAppliedStereotypes())]<drop/>  
[if(not st.name.contains(‘OpenModelClass’))]<drop/>

This class is [st.name/].

[else] <drop/>  
[/if]<drop/>  
[/for]<drop/>  
[else] <drop/>  
[/if]  
[/if]  
</fragment><drop/>

# Fragment: Insert standard diagram <drop/>

<fragment name=’insertStandardDiagram’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Package’/><drop/>  
<arg name=’diagramName’ type=’String’/><drop/>  
<arg name=’diagramTitle’ type=’String’/><drop/>

[for (d:Diagram|p.getPapyrusDiagrams())]<drop/>

[if d.name.contains(diagramName)]

<drop/>

<image object='[d.getDiagram()/]' maxW='true' keepH='false' keepW = ‘false’></image>

CoreModel diagram: [d.name/]

Figure 12-1 [diagramTitle/]

[else]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert small diagram <drop/>

<fragment name=’insertSmallDiagram’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Package’/><drop/>  
<arg name=’diagramName’ type=’String’/><drop/>  
<arg name=’diagramTitle’ type=’String’/><drop/>

[for (d:Diagram|p.getPapyrusDiagrams())]<drop/>

[if d.name.contains(diagramName)]

<drop/>

<image object='[d.getDiagram()/]' maxW='true' keepH='false' keepW = ‘false’></image>

CoreModel diagram: [d.name/]

Figure 13-1 [diagramTitle/]

[else]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert attribute row brief not Obsolete<drop/>

<fragment name=’insertAttributeRowBriefNotObsolete’ importedBundles=’commons;gmf;papyrus’><drop/>

Does not work unless we have Mature stereotype… <drop/>  
<arg name=’p’ type=’uml::Property’/><drop/>

[for (st:Stereotype | p.getAppliedStereotypes())]<drop/>

[if(not st.name.contains(‘OpenModelAttribute’))]

[if(not st.name.contains(‘Obsolete’))]

| [p.name/] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [if(not st.name.contains(‘OpenModelAttribute’))] [st.name/]  [/if]<drop/>  [/for]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [cleanAndFormat(c.\_body.clean())/]  [/for]  [else] [if (p.name.contains (‘\_’))]See referenced class  [else]To be provided  [/if]<drop/>  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> |
| --- | --- | --- |

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert attribute row brief <drop/>

<fragment name=’insertAttributeRowBrief’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Property’/><drop/>

| [p.name/] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [if(not st.name.contains(‘OpenModelAttribute’))] [st.name/]  [/if]<drop/>  [/for]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [cleanAndFormat(c.\_body.clean())/]  [/for]  [else] [if (p.name.contains (‘\_’))]See referenced class  [else]To be provided  [/if]<drop/>  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> |
| --- | --- | --- |

</fragment><drop/>

# Fragment: Start attribute table brief <drop/>

<fragment name=’insertAttributeTableHeader’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Lifecycle Stereotype (empty = Mature)** | **Description** |

</fragment><drop/>

# Fragment: Insert Attribute table brief <drop/>

<fragment name=’insertAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>  
[if cl.ownedAttribute->notEmpty()]<drop/>

Table 2: Attributes for [cl.name/]

<table><drop/>

[cl.insertAttributeTableHeader ()/]

[for (p:Property|cl.ownedAttribute)]<drop/>

[if (not p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/for]<drop/>

[for (p:Property|cl.ownedAttribute)]<drop/>

[if (p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert Ten Specified Attribute table brief <drop/>

<fragment name=’insertTenSpecifiedAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>

<arg name=’p1’ type=‘String’/><drop/>

<arg name=’p2’ type=‘String’/><drop/>  
<arg name=’p3’ type=‘String’/><drop/>  
<arg name=’p4’ type=‘String’/><drop/>  
<arg name=’p5’ type=‘String’/><drop/>  
<arg name=’p6’ type=‘String’/><drop/>  
<arg name=’p7’ type=‘String’/><drop/>  
<arg name=’p8’ type=‘String’/><drop/>  
<arg name=’p9’ type=‘String’/><drop/>  
<arg name=’p10’ type=‘String’/><drop/>  
[if cl.ownedAttribute->notEmpty()]<drop/>

Table 3: Attributes for [cl.name/]

<table><drop/>

[cl.insertAttributeTableHeader ()/]

[for (p:Property|cl.ownedAttribute)]<drop/>

[if (p.name.contains(p1) or p.name.contains(p2) or p.name.contains(p3) or p.name.contains(p4) or p.name.contains(p5) or p.name.contains(p6) or p.name.contains(p7) or p.name.contains(p8) or p.name.contains(p9) or p.name.contains(p10))]<drop/>

[if (not p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/if]<drop/>

[if (p.name.contains(p1) or p.name.contains(p2) or p.name.contains(p3) or p.name.contains(p4) or p.name.contains(p5) or p.name.contains(p6) or p.name.contains(p7) or p.name.contains(p8) or p.name.contains(p9) or p.name.contains(p10))]<drop/>

[if (p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert DataType <drop/>

<fragment name=’insertDataType’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>  
<arg name=’dataTypeName’ type=’String’/><drop/>  
<arg name=’packageName’ type=’String’/><drop/>  
[if (dt.qualifiedName.contains(packageName))]<drop/>  
[if(dt.name.contains(dataTypeName))]<drop/>

Qualified Name: [dt.qualifiedName/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[cleanAndFormat(co.\_body.clean())/]</dropEmpty>

[/for]<drop/>  
[if (dt.oclAsType(uml::DataType).general ->notEmpty())]<drop/>

Inherits properties from:

[for (tp:DataType | dt.oclAsType(uml::DataType).general)]<drop/>

* [tp.name/]

[/for]<drop/>

[for (gen:Class | dt.oclAsType(uml::DataType).general)]<drop/>

* [gen.name/]

[/for]<drop/>

[/if]<drop/>

[for (st:Stereotype | dt.getAppliedStereotypes())]<drop/>  
This class is [st.name/].

[/for]<drop/>  
[else] <drop/>  
[/if]  
[/if]  
</fragment><drop/>

# Fragment: Start Data Type attribute table brief <drop/>

<fragment name=’insertDataTypeAttributeTableHeader’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Lifecycle Stereotype (empty = Mature)** | **Description** |

</fragment><drop/>

# Fragment: Insert Data Type Attribute table brief <drop/>

<fragment name=’insertDataTypeAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertDataTypeAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>  
[if dt.ownedAttribute->notEmpty()]<drop/>

Table 4: Attributes for [dt.name/]

<table><drop/>

[dt.insertDataTypeAttributeTableHeader ()/]

[for (p:Property|dt.ownedAttribute)]<drop/>

[p.insertAttributeRowBrief ()/]

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert enums <drop/>

<fragment name=’insertEnums’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>

#### [dt.name/]

Qualified Name: [dt.qualifiedName/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[cleanAndFormat(co.\_body.clean())/]</dropEmpty>

[/for]<drop/>

Applied stereotypes:

[if dt.getAppliedStereotypes()->notEmpty()] <drop/>

[for (st:Stereotype | dt.getAppliedStereotypes())]<drop/>

* [st.name/]

[/for]<drop/>

[else] No stereotypes applied

[/if]<drop/>

[if (dt.oclAsType(uml::DataType).general ->notEmpty())]<drop/>

Inherits literals from:

[for (tp:DataType | dt.oclAsType(uml::DataType).general)]<drop/>

* [tp.name/]

[/for]

[/if]<drop/>

[if (dt.oclAsType(Enumeration).ownedLiteral->notEmpty())]<drop/>

Contains Enumeration Literals:

[for (e:EnumerationLiteral|dt.oclAsType(Enumeration).ownedLiteral)]<drop/>

* [e.name/]:
  + [for (co:Comment | e.ownedComment)]<drop/>
  + <dropEmpty>[cleanAndFormat(co.\_body.clean())/]
  + </dropEmpty>[/for]<drop/>
  + [if dt.getAppliedStereotypes()->notEmpty()] <drop/>
  + Applied stereotypes:
    - [for (st:Stereotype | e.getAppliedStereotypes())]<drop/>
    - [st.name/]
    - [/for]<drop/>
  + [/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

1. note that the current structure only allows the phasing to start at the beginning of a continuous period. The model allows for further augments of structures that allow greater flexibility (assuming that the general union/intersection pattern is followed). A structure could be developed similar to the current PhasedPeriodIntersectionTe that has offset structures that can “delay” the start after the continuous period boundary for each of the phasings. [↑](#footnote-ref-1)