Core Information Model (CoreModel)

TR-512.10

Operation Patterns

Version 1.3.1
January 2018

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.3.1

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
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

©2018 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.

Table of Contents

 © 2018 Open Networking Foundation

List of Figures

 © 2018 Open Networking Foundation

Document History

Version	Date	Description of Change
		This document was not published prior to Version 1.3
1.3	September 2017	Version 1.3 [Published via wiki only]
1.3.1	January 2018	Addition of text related to approval status.

© 2018 Open Networking Foundation

1 Introduction

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.

1.1 References

For a full list of references see TR-512.1.

1.2 Definitions

For a full list of definition see TR-512.1.

1.3 Conventions

See TR-512.1 for an explanation of:

- UML conventions
- Lifecycle Stereotypes
- Diagram symbol set

1.4 Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%), 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).

1.5 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 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.

2 Introduction to the Operation Patterns

The focus of this document is the modeling of generalized operations.

2.1 Background to the work

The work has been carried out with the assumption that the future is cloud oriented such that the controllers are an interconnected system of cloud-based components and such that inter-

© 2018 Open Networking Foundation

component (inter-controller and intra-controller communications) will be via native cloud interface encodings. Hence, the interface encodings are not relevant as they can be generated automatically using tooling from some semantic definition of the required exchange[1]. Necessary mapping to the communication infrastructure is provided by capabilities in the cloud environment.

The key then is the semantics presented at the programmers interface[2]. The programmers' interface is presented in a coding language (such as Python, Java etc.) but the semantics are consistent independent of language and hence the language of encoding is not relevant[3].

Interface interaction will benefit from normalization of interaction patterns and a messaging grammar that unifies the variety of interaction complexities (CRUD, Intent etc.) in a single sophisticated structure[4]. The key is modelling the message content structure. It is modelling of the structuring of the content that is the focus of this document.

The structure definition discussed in this document enables the folding away of capability that is unnecessary for any specific interaction. The structure is efficient and simple for simple operations but sophisticated and versatile for complex interactions[5].

It is assumed that in a cloud environment the operations will be "outcome-oriented" interaction[6] where the focus is on stating the constraints that form a boundary that defines the desired target. In outcome-oriented interaction the operations/methods/activities/tasks used to achieve the desired outcome are firmly in the domain of the provider. The client simply provides information about the desired outcome in the context of what has been agreed as possible[7]. Hence the essential need of any interaction is the provision of information about the desired outcome in terms of constraints and potentially in the context of some expected initial system state. Whilst the content of any message will differ per interaction the structure will be consistent[8].

The content of any message will differ per viewpoint and per interaction. The key to content opportunity for any message is the viewpoint context shared between the parties that are interfacing. The viewpoint context relates to the shared semantics about which the parties need to communicate. The allowable content in an exchange is determined by the properties of the attributes in that viewpoint (read only, read-write etc.)

The information about any particular context or outcome can be a flat structure of statements about things and their properties that is as simple as possible for the specific interaction. The content and its semantic structure detail can vary from interaction to interaction. The semantic

[1] The language of exchange is generated by a compiler from the semantic model. Depending upon the sophistication of the environment the compilation may be run-time and dependent upon operational parameters. The same semantics may be encoded in bit form for one exchange and in verbose XML for another exchange. The change in encoding may take place between two operations between the same two control elements.
[2] Traditional definition of API
[3] The programmers' language is generated by a compiler from the semantic model.
[4] Increasing consistency of the interface interaction reduces training and the broadens the applicability of any particular related software.
[5] A good analogy for the sort of structure is human language grammar.
[6] Intent is an outcome-oriented form of interaction.
[7] As defined in terms of a shared model of semantics
[8] Again, human language is a good analogy. The grammar remains constant, simple and repeating but the vocabulary is broad and changes/grows often rapidly.

 © 2018 Open Networking Foundation

structure is conveyed in one or more interrelated information models. TR-512.7 provides more information on the combining of models to provide a per-case definition of properties etc.

As understood from various activities over the past decade or so the concept of an NE as a thing is broken. In this release a Controller model has been developed (see TR-512.8). The controlling essence of the NE is now modelled in the same way that any other manager-controller is. It is just another controller. The expectation is that the interfaces at the "NE" will migrate to the same form as those from any other and hence will become "outcome oriented". The emergence of white box NE (from a control perspective) also points to "cloudification" of the controller functions in the NE which further emphasizes the evolution to "outcome oriented" interfaces.

In the environment sketched above it is expected that interfaces structured as described in this document and generated from canonical models via tooling will be the norm, where the interface structure and interactions will be independent of model of things being discussed and of coding of interfaces.

2.2 Goals

The interface will:

- Aim to maximise integration without over constraining Interaction Pattern
- Deal with ecosystem that forms an Agile Value Fabric (AVF)
- Support Dynamic API/Interface requirements
- Stabilize the definition of infrastructural considerations so as to free up effort to deal with the more relevant interaction value

The common pattern should:

- Be such that all traditional interface interaction patterns could be derived by "pruning & refactoring" the common pattern.
- Capture all semantics of machine interfacing

The underpinning pattern should provide the opportunity for expression of:

- Outcome oriented constraint based sub-graph for creation/modification/deletion (this is discussed here)
- Constraint based sub-graph query with sub-graph response (not discussed in this release of this document)
- Notification of sub-graph changes (not discussed in this release of this document)

2.3 Aspects of the work

- Basic Message Exchange Patterns – Simple exchanges
    - E.g. message/response, notification etc.
- Complex Message Exchange Patterns – Sequence of outcomes and potentially sequence of exchangs
    - E.g. for long lived operations with pause/resume
- Encapsulation grammar
    - E.g. content framing

© 2018 Open Networking Foundation

- Statement style
  - E.g. action verb oriented, outcome oriented
- Statement definition specificity
  - E.g. pictorial phrase v SVO (Subject, Verb, Object)
- Definition fluidity
  - E.g. Dynamic API/Interface

2.4 Some terminology

2.4.1 Agile Value Fabric (AVF)

- Is an evolution of the term value chain
  - Recognises that the term chain does not reflect the complexity
    - Fabric implies both a complex mesh and also some intentional structure
  - Recognises that the structure is not static
    - Agile emphasises that the fabric needs to change in a particularly efficient way to deal with the environment into which it plays
- Is the arrangement of interacting parties/components that is in place to offer solutions to particular needs in the environment
  - These needs may be known up-front and contracted prior to the formation of the AVF
    - This is like a "civil engineering project"
  - The AVF may form to offer a speculative capability where there is assumed to be a need that has not yet emerged
    - This is like a "market stall"
  - The needs may be partially known
    - And hence is a hybrid of the two above
- Note: The "up-front" case suggest a known arrangement of components where both (all) ends of an interaction (client/server/peer) are known and stated BUT may evolve independently.

2.4.2 Dynamic API/Interface

An API could be considered as dynamic if it offers some of the following:

- On-the-fly ecosystem-asynchronous changes to fragments of run time schema whilst maintaining on-going undisrupted compatibility
- Extension by augmenting compile time schema with runtime schema supporting on the fly changes to the extensions
- On the fly rebalancing of compile time and run time schema usage
  - What was compile time becomes run time and vice-versa
- Varying model per operation and morphing model per case of use and per instance of use based upon rules
  - Including per interaction based folding/compacting of model
- Outcome oriented constraint based interactions with grammar that enables seamless interaction across the range from "phrase-book-user to orator"

 © 2018 Open Networking Foundation

- Published machine interpretable uniquely identified schema, for all of the above, referenced in the interaction and acquired from an on-line library

2.5 Introduction to this document

This document:

- Introduces the operations pattern
- Illustrates the fold-away nature

The model relates to all other models but especially to:

- ONF Specification in TR-512.7

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

3 Purpose and essentials of the Operation Patterns

3.1 Background

This document covers aspects of each of the items listed below.

- Basic Message Exchange Patterns – Simple exchanges
  - E.g. message/response, notification etc.
- Complex Message Exchange Patterns sequences
  - E.g. for long lived operations with pause/resume
- Encapsulation grammar
  - E.g. content framing
- Statement style
  - E.g. action verb oriented, outcome oriented
- Statement definition specificity
  - E.g. pictorial phrase v SVO (Subject, Verb, Object)
- Definition fluidity
  - E.g. Dynamic API/Interface

3.2 The model

The model here is an early experimental sketch

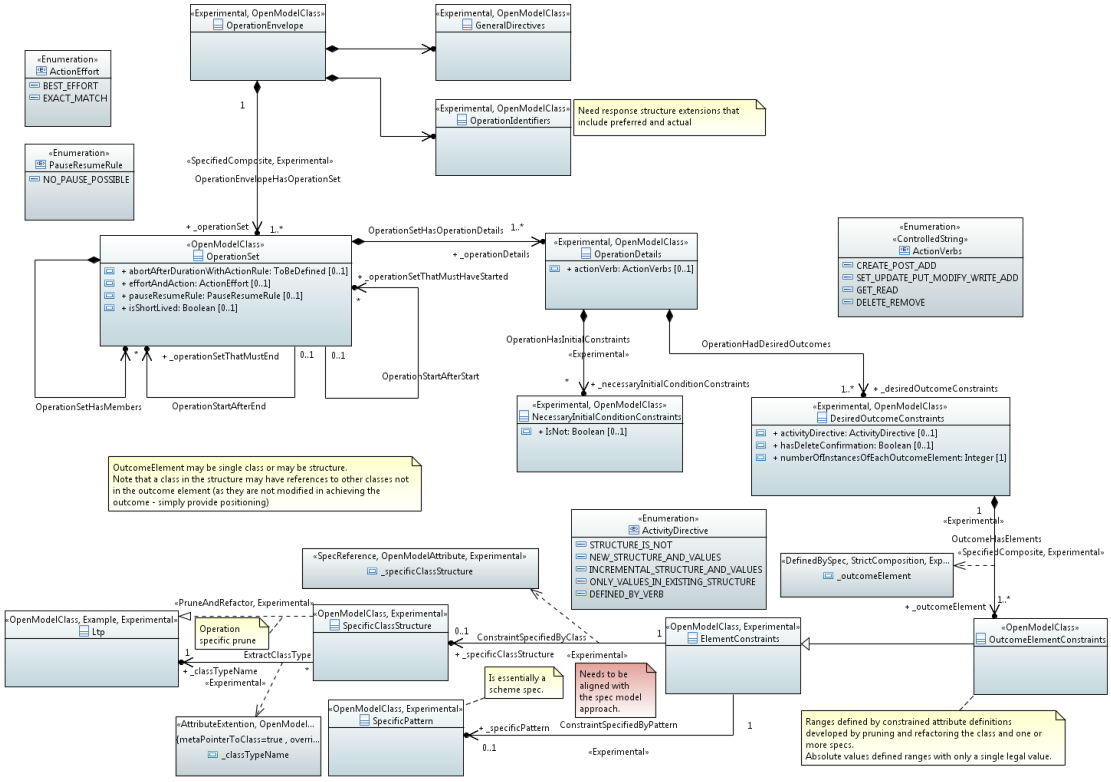
- The model is intended to provide a dynamic sophisticated structure that has "foldaway" parts

© 2018 Open Networking Foundation

    - This allows for the construction of sub-set schema for different degrees of sophistication where each of the sub-set schema is fully compatible with the full schema
- The aim is to provide one structure:
    - For all outcome oriented constraint based forms including intent
    - Supports traditional Verb driven forms
        - with constrained valued
        - with absolute values
    - Enables operations that:
        - Act on multiple separate independent things
        - Have sequence and interdependency between parts and with other separate interactions
        - Are long lived or short lived (where the life may depend upon the case and may not be knowable before the request
- There are almost certainly errors in the model and it is only presented at this point to stimulate discussion
- The aim is that the model will be used to generate schema where there is a continuum of compatible schema from the most basic simple CRUD forms to the most sophisticated forms such that he CRUD form can be seen as a tiny subset of the sophisticated form

The following figure shows the model of the request.

© 2018 Open Networking Foundation



CoreModel diagram: Operation-Structure

Figure 3-1 The structure of an operation (request)

In the figure highlights the relationship between the approaches used in this model and the specification model (see TR-512.7). The model here has not been fully aligned with the specification model.

3.3 The structure

As usual, the classes in the model represent structure that will appear in the interface schema. The classes in the model described in the following section. Note that at this early stage of development limited detail has been provided. The descriptions will be enhanced in subsequent releases as the model is refined.

3.3.1 DesiredOutcomeConstraints

Qualified Name:
CoreModel::CoreOperationsModel::ObjectClasses::DesiredOutcomeConstraints

This class is Experimental.

Table 1: Attributes for DesiredOutcomeConstraints

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
activityDirective	Experimental	To be provided

 © 2018 Open Networking Foundation

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
hasDeleteConfirmation	Experimental	To be provided
numberOfInstancesOfEachOutcomeElement	Experimental	To be provided
_outcomeElement	DefinedBySpec StrictComposition Experimental	See referenced class

3.3.2 ElementConstraints

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::ElementConstraints

This class is Experimental.

Table 2: Attributes for ElementConstraints

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_specificClassStructure	SpecReference Experimental	See referenced class
_specificPattern	Experimental	See referenced class

3.3.3 GeneralDirectives

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::GeneralDirectives

This class is Experimental.

3.3.4 Ltp

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::Ltp

This class is Example.

This class is Experimental.

3.3.5 NecessaryInitialConditionConstraints

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::NecessaryInitialConditionConstraints

This class is Experimental.

© 2018 Open Networking Foundation

Table 3: Attributes for NecessaryInitialConditionConstraints

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
IsNot	Experimental	To be provided

3.3.6 OperationDetails

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::OperationDetails

This class is Experimental.

Table 4: Attributes for OperationDetails

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
actionVerb	Experimental	To be provided
_necessaryInitialConditionConstraints	Experimental	See referenced class
_desiredOutcomeConstraints	Experimental	See referenced class

3.3.7 OperationEnvelope

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::OperationEnvelope

This class is Experimental.

Table 5: Attributes for OperationEnvelope

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
generalDirectives	Experimental	To be provided
operationIdentifiers	Experimental	To be provided
_operationSet	Experimental	See referenced class

3.3.8 OperationIdentifiers

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::OperationIdentifiers

This class is Experimental.

© 2018 Open Networking Foundation

3.3.9 OperationSet

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::OperationSet

Table 6: Attributes for OperationSet

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
abortAfterDurationWithActionRule	Experimental	To be provided
effortAndAction	Experimental	To be provided
pauseResumeRule	Experimental	To be provided
operationSet	Experimental	To be provided
isShortLived	Experimental	To be provided
_operationSetThatMustHaveStarted	Experimental	See referenced class
_operationSetThatMustEnd	Experimental	See referenced class
_operationDetails	Experimental	See referenced class

3.3.10 OutcomeElementConstraints

Qualified Name:
CoreModel::CoreOperationsModel::ObjectClasses::OutcomeElementConstraints

Inherits properties from:

- ElementConstraints

3.3.11 SpecificClassStructure

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::SpecificClassStructure

This class is Experimental.

Table 7: Attributes for SpecificClassStructure

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_classTypeName	AttributeExtention Experimental	See referenced class

3.3.12 SpecificPattern

Qualified Name: CoreModel::CoreOperationsModel::ObjectClasses::SpecificPattern

This class is Experimental.

© 2018 Open Networking Foundation

3.4 "Foldaway complexity" explained

The following sections progress through foldaway rationale that, in a sequence of steps, takes the complex structure presented above and transforms it to a simple Create operation of a basic CRUD form. This is simply an illustrative example of how to arrive at an interface expression that is no more complex than required and in this case is no more complex than that used for current interfaces.

3.4.1 Single Envelope

The generalized structure offers the opportunity to have many outcome structures in separate sets. If only one structure is required only one set is required and the nesting of composition can be removed. The basic principle is that 1 of 1 can be collapsed (the subordinate is folded away (mered) into the superior). The general idea is that if an application:

- Is coded with the "collapse" rules it will be able to interpret the structure resulting from the collapse
- Does not need anything other than one envelope it can be coded to only deal with the collapse and need have no knowledge of the expanded form

This general principle applies to all of the following sections.

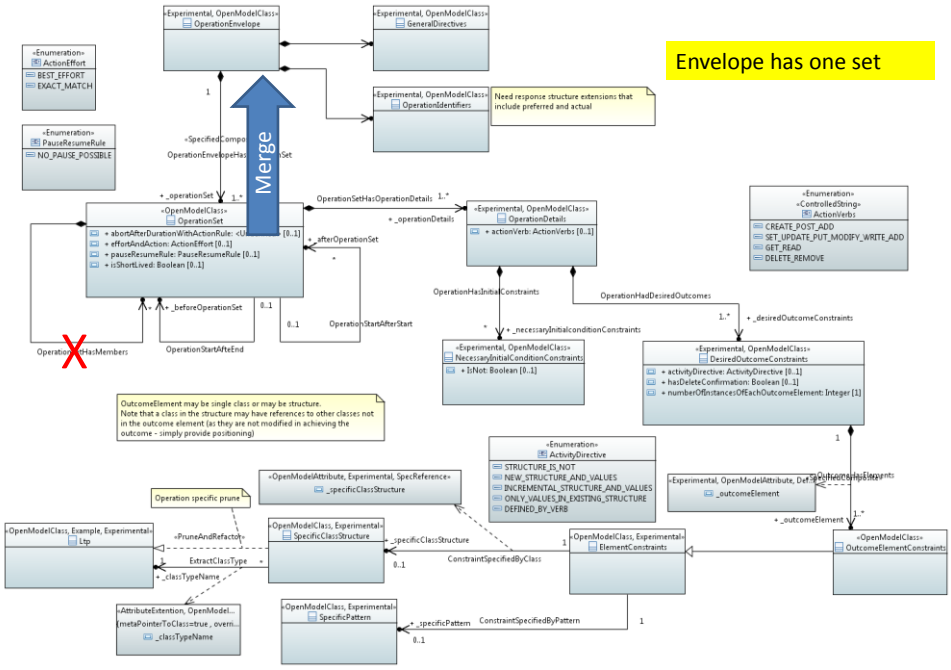


Figure 3-2 Only a single operation set

© 2018 Open Networking Foundation

3.4.2 One Operation

In the figure below only one operation is required. This allows the OperationDetails to be folded into the Operation Envelope. The operation has members can also be removed (as this is a nesting for multiple contained operations).

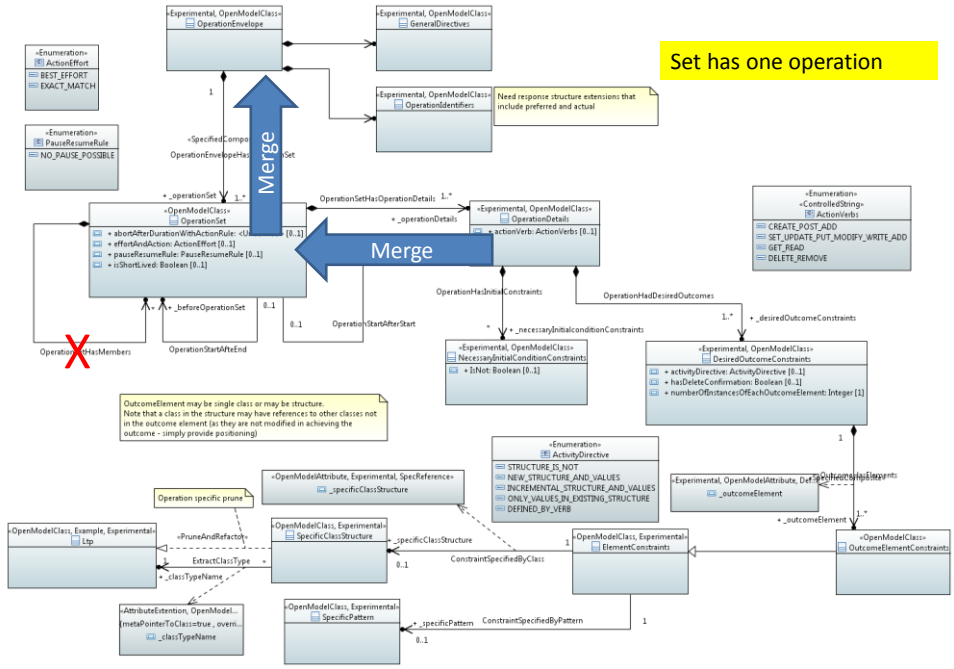


Figure 3-3 One operation

3.4.3 No start/finish dependencies

If the operation is standalone in relation to the start or end of other operations, the associations supporting operation interrelationship can be removed.

© 2018 Open Networking Foundation

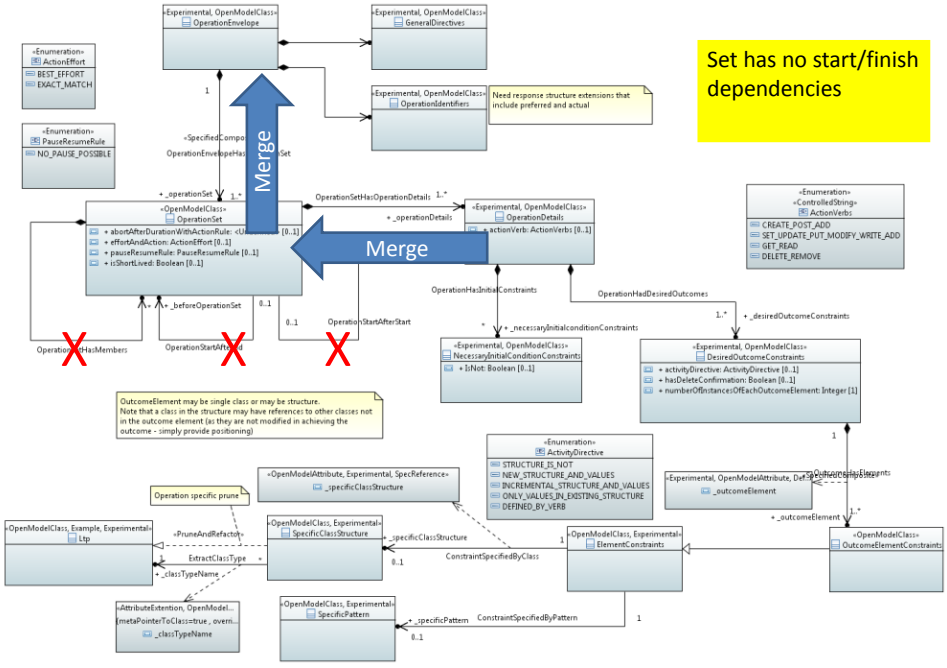


Figure 3-4 No start/finish dependencies

3.4.4 Operation is not idempotent

If the operation is such that a repeat of the same operation will cause failure then the operation is not idempotent, default initial condition constraints can be assumed (where the default is that the entities identified in the outcome should not exist prior to the request) and hence the initial condition structure can be removed.

© 2018 Open Networking Foundation

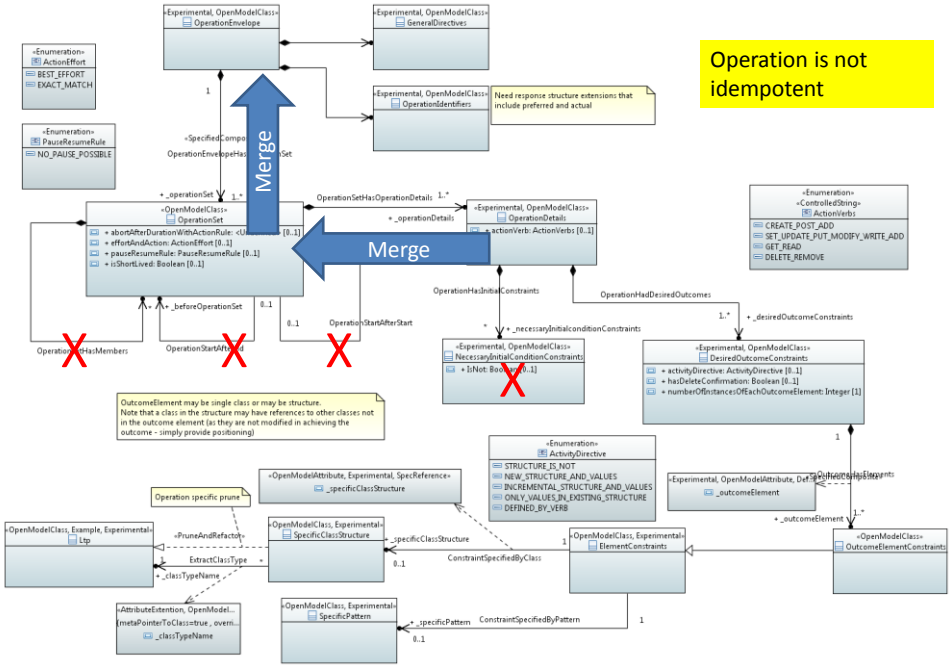


Figure 3-5 Operation not idempotent

3.4.5 Operation is short-lived

If the operation is short-lived the pause/resume capability is not meaningful.

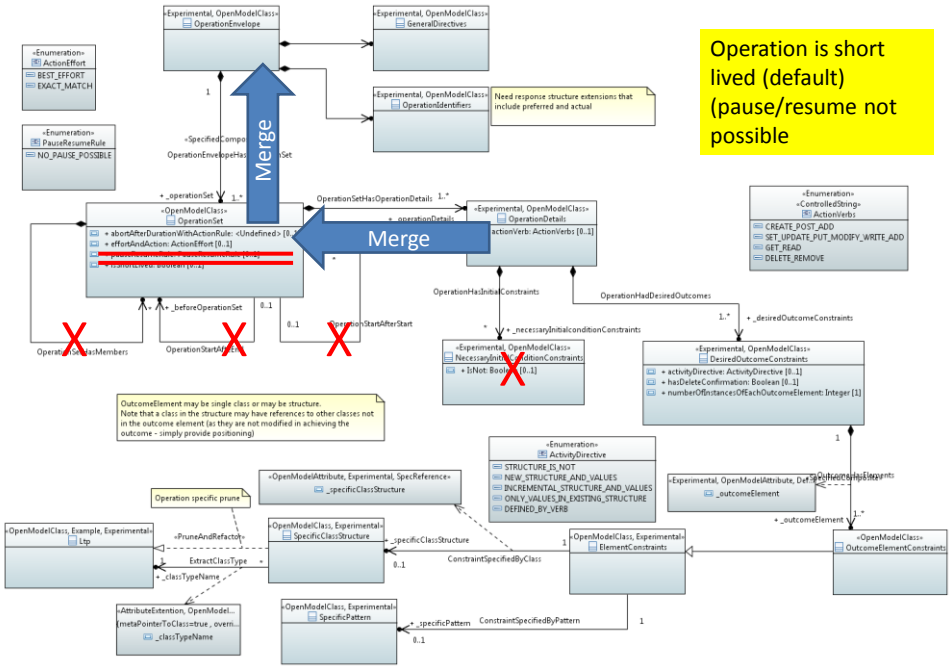


Figure 3-6 Short-lived operation

© 2018 Open Networking Foundation

3.4.6 Operation has no relevant abort behavior

It may not be possible to abort the operation (especially true for short-lived operations).

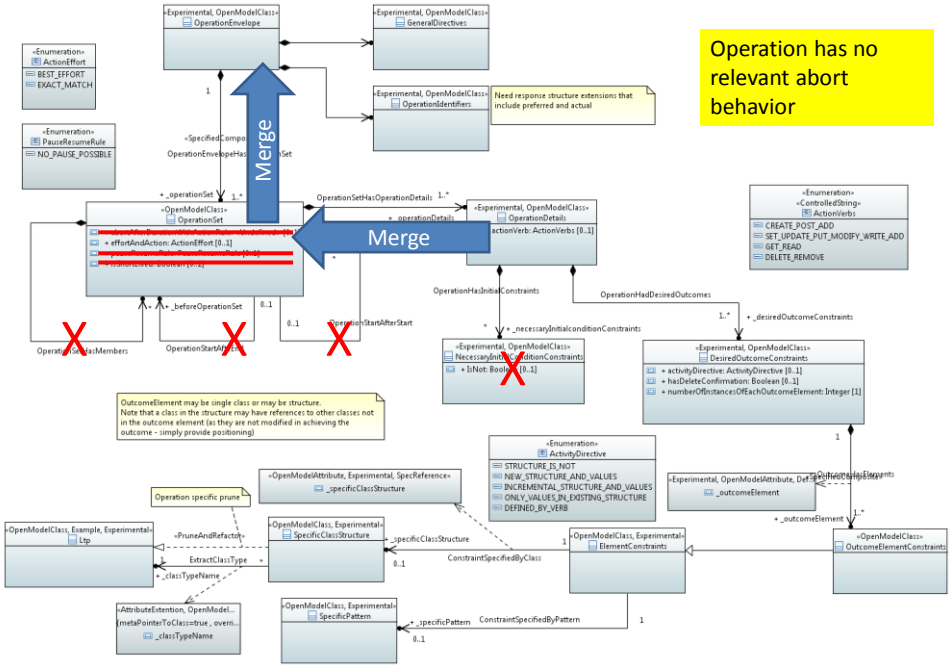


Figure 3-7 No abort

3.4.7 Operation is EXACT_MATCH

If the specification of outcome constraints is such that all properties are defined precisely then the default EXACT_MATCH need not be stated.

© 2018 Open Networking Foundation

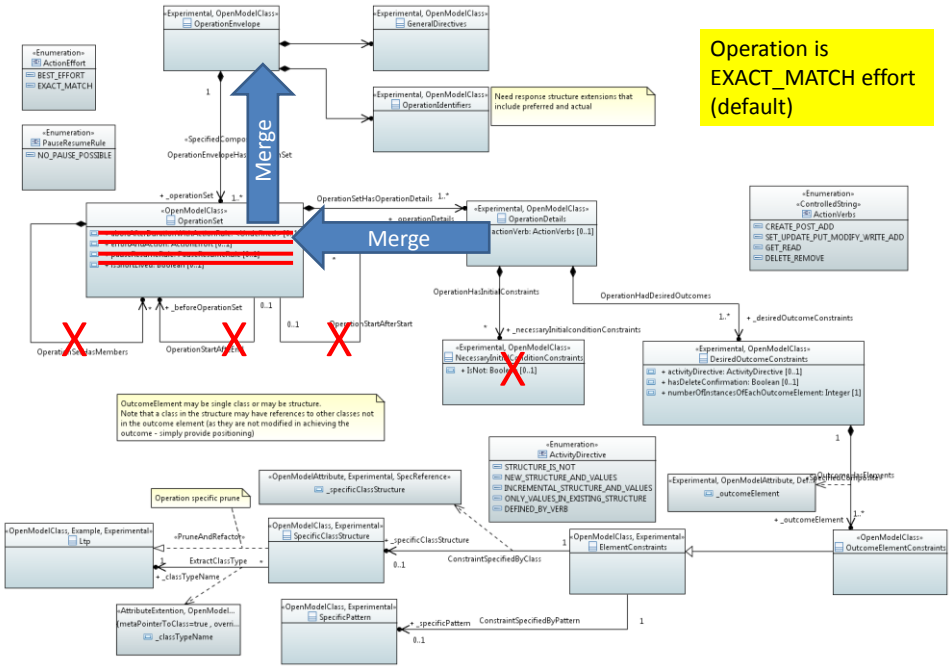


Figure 3-8 Exact match

3.4.8 ActivityDirective is DEFINED_BY_VERB

The action to be performed is defined by a verb as it is for traditional interface. The action Verb will be folded into the OperationEnvelope. An action verb CREATE_POST_ADD will be assumed here.

© 2018 Open Networking Foundation

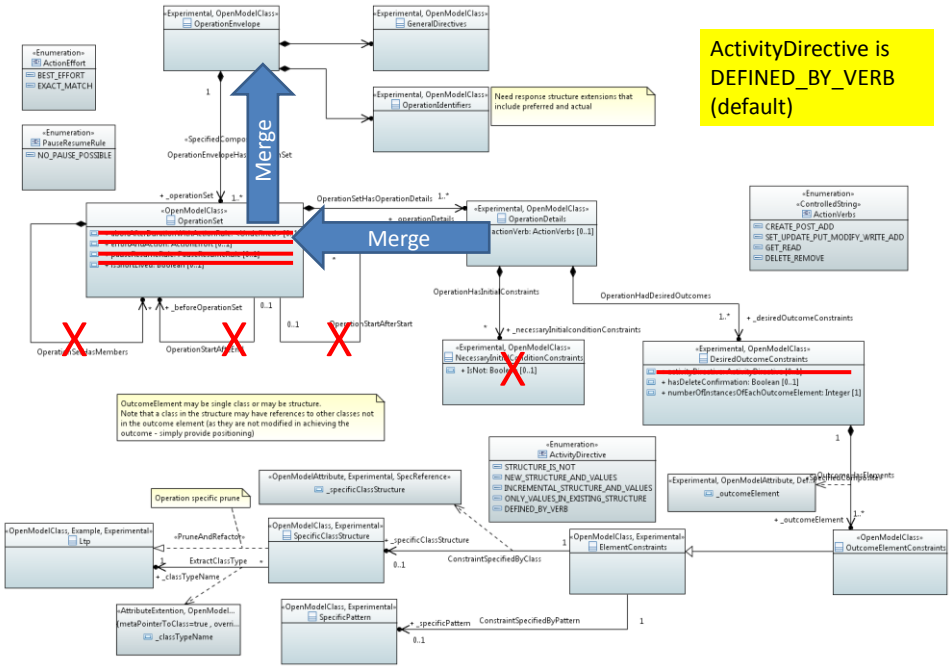


Figure 3-9 Defined by verb

3.4.9 Has delete confirmation

If the originator of the operation requires a delete confirmation then default behavior is required and the attribute supporting non-default responses need not be provided.

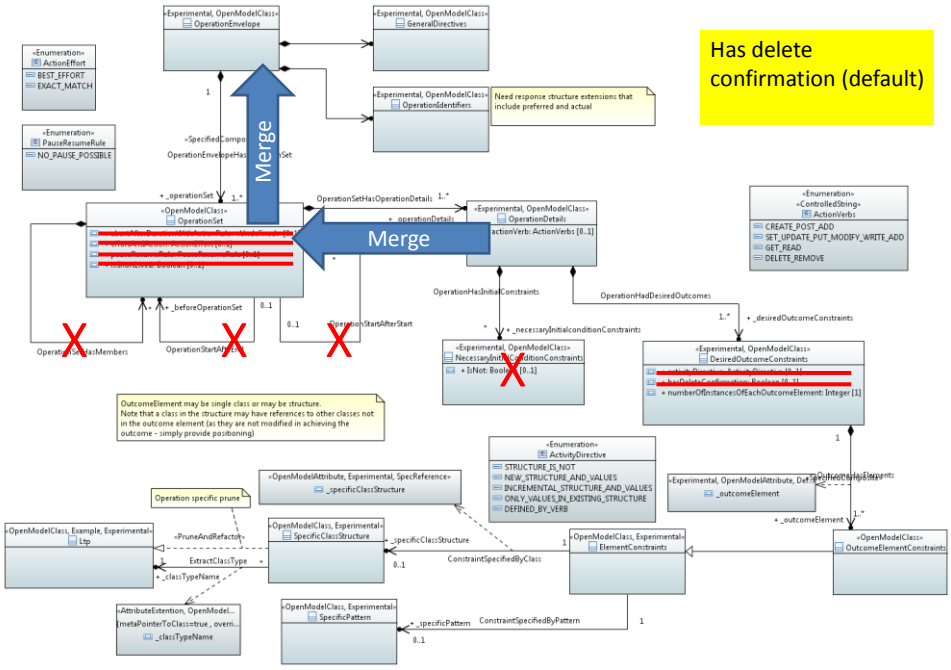


Figure 3-10 Requires delete confirmation

© 2018 Open Networking Foundation

3.4.10 Has one outcome instance

As one outcome is the default the "numberOfInstancesOfEachOutcomeElement" does not need to be stated.

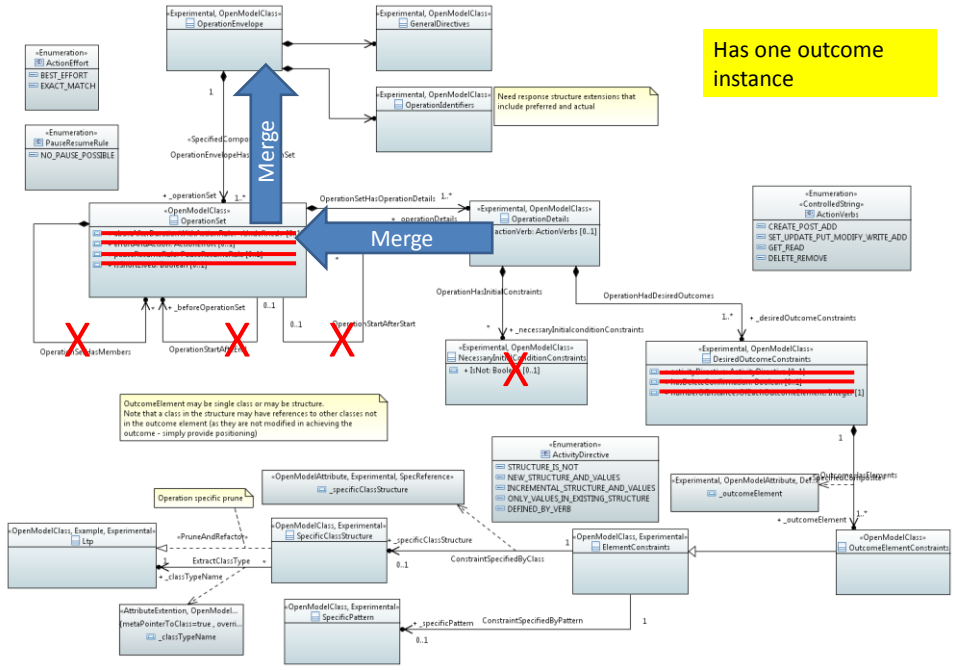


Figure 3-11 Has one outcome instance

3.4.11 Outcome is an LTP

Considering the example of "creation" of an LTP, as the outcome is an LTP the specific class structure will be an LTP. An actual usage will include an LTP decorated by one or more specifications to provide a precise model of the specific functionality. This decorated LTP will be pruned and refactored as necessary to produce the relevant structure for the interface (e.g. read only attributes will be removed). The pruning and refactoring action will also need to set the relevant ranges etc for each attribute for the specific instance of operation so as to precisely define the desired outcome. The results of this process become part of the structure of the operation.

If a traditional create is assumed then each of the writeable attributes remaining in the pruned structure will have a value range with only one value allowed.

© 2018 Open Networking Foundation

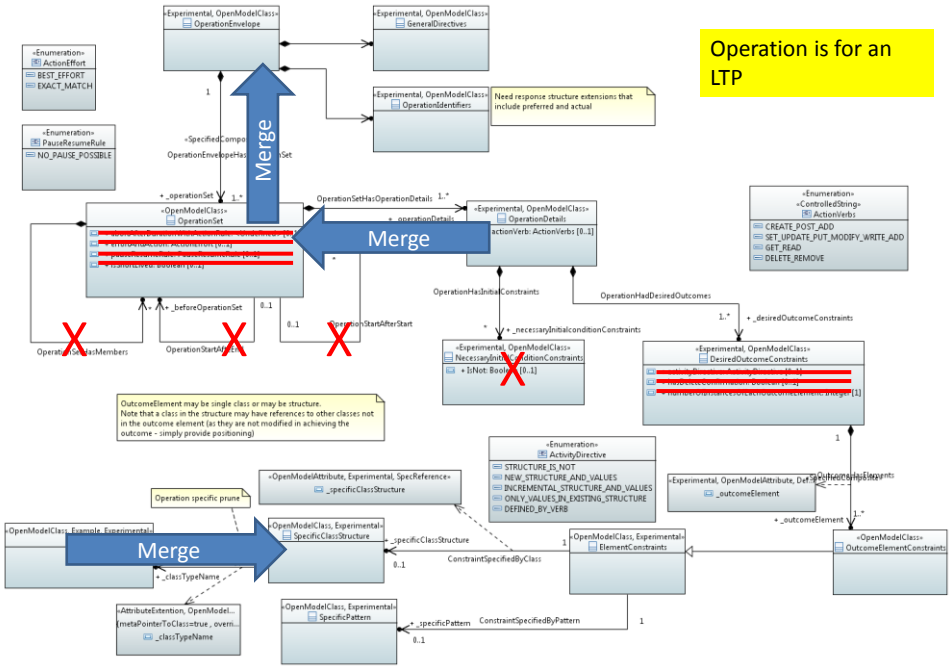


Figure 3-12 Outcome is an LTP

3.4.12 Has one specific class outcome constraint

In this case only one instance of one class (LTP) is ever going to be operated on at a time so the multiplicities become always 1 of 1 and hence the specific outcome LTP statement can be merged into the OperationEnvelope

© 2018 Open Networking Foundation

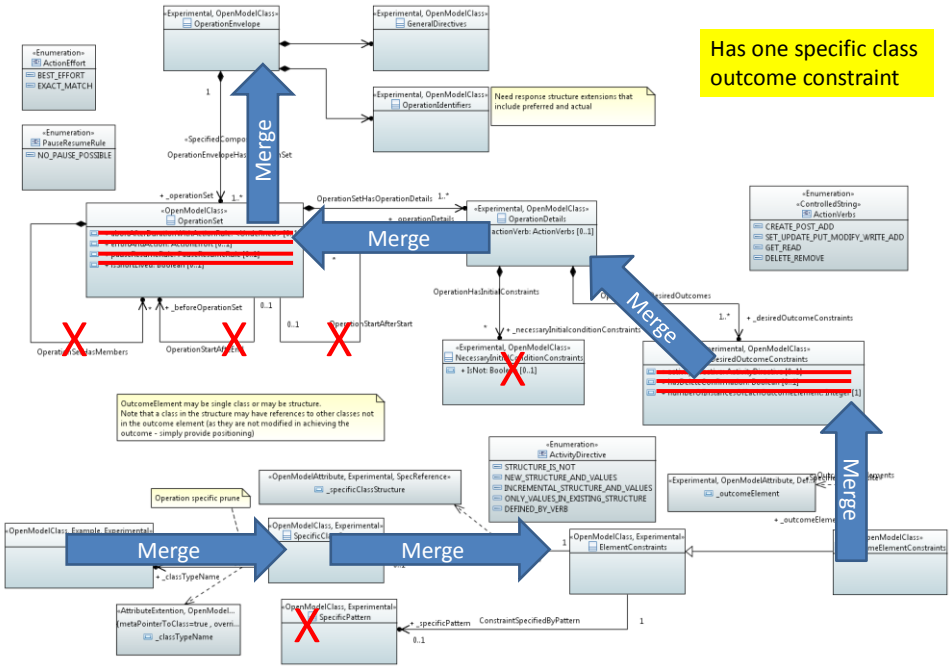


Figure 3-13 The outcome is a single LTP

3.4.13 Has no special directives

The operation has no general directives.

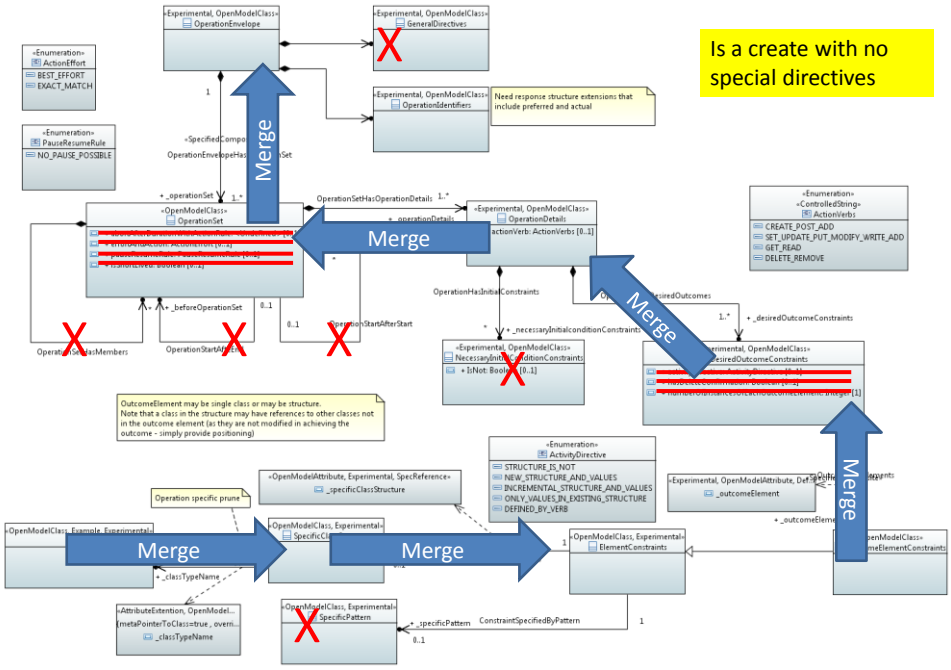


Figure 3-14 Basic spec pattern with rule sketch

© 2018 Open Networking Foundation

3.4.14 Resulting structure

The above merging results in a single level operation structure with CREATE_POST_ADD as the operation verb and with contents of an LTP structure with absolute values for each relevant writeable attribute. This is essentially the same structure complexity as a Create in a CRUD system, but it is fully conformant with the grammar of the far more sophisticated Outcome-Oriented form. The sophisticated structure is unnecessary for the basic CRUD style operation. The structure would be viewed by a CRUD designer as massive unnecessary complexity if imposed on the simple cases. The mechanism set out by example above "folds away" the complexity.

A basic application could simply operate with the CRUD form whilst interworking with a more sophisticated application that downgrades its operation based upon the structures it receives (and the specification of capability presented by the basic application).

If the basic application is upgraded to a more sophisticated form there is no need to recompile the interface as the more sophisticated form will still be a simple folded sub-set of the full interface representation.

3.4.15 Further examples

More specific examples will be provided in future released of this model.

3.5 Discussion

3.5.1 Alternatives explored

A stereotype approach was considered. It was concluded that the stereotype might not be helpful and would not be sufficient unless as complex as the structure discussed in this document. An existing stereotype scheme was discussed and it was noted that it did not provide the depth required.

3.5.2 Details of the structure

PauseResumeRule: Is not Boolean as more values are expected to be defined.

abortAfterDurationWithActionRule: Is used to provide action on timeout of operation where the action is an abort. A policy approach may be more appropriate in general here.

OperationDetails: ActionVerbs [0..1]: The ActionVerb is not necessary as the InitialCondition and DesiredOutCome provide all necessary information to the subordinate system to allow it to do its job. The ActionVerb has been included in recognition of the existing familiar verbal style. It is clear, however, that the verb is often unnecessary even in existing interfaces especially when complex verbs such as "createAndActivate" are used in conjunction with an idempotent operation approach.

The ActivityDirective needs to be developed further. The current directives are relatively blunt and have the effect of the ActionVerb. For example STRUCTURE_IS_NOT could be read as essentially indicating a delete. But it does not assume any specific action in the underlying system and can be used to indicate a complex partial delete with clean-up of properties of instances of related things as the structure can include only aspects of the overall.

 © 2018 Open Networking Foundation

4 Future work (see also TR-512.FE)

Future work will cover:

- Notification approach (including Alarm notification)
- Operation interaction modelling (sequence diagrams etc) including pause/resume and progress indications/queries
    - Note that the operation is performed by a controller. The client talks to the controller about the relevant classes representing the solution resources/services
- Temporal modelling

End of document

© 2018 Open Networking Foundation