WG Working Group
Internet-Draft
Intended status: Informational
Expires: 14 September 2023

N. Davis
Ciena

13 March 2023

Modelling Boundaries
draft-davis-netmod-modelling-boundaries-01

Abstract

Current modelling techniques appear to have boundaries that make representation of some concepts in modern problems, such as intent and capability, challenging. The concepts all have in common the need to represent uncertainty and vagueness. The challenge results from the rigidity of boundary representation, including the absoluteness of instance value and the process of classification itself, provided by current techniques.

When describing solutions, a softer approach seems necessary where the emphasis is on the focus on a particular thing from a particular perspective. Intelligent control (use of AI/ML etc.) could take advantage of partial compatibilities etc. if a softer representation was achieved.

The solution representation appears to require

* Expression of range, preference and focus as a fundamental part of the metamodel

* Recursive tightening of constraints as a native approach of the modeling technique

This lead to a need to enhance the metamodel of languages that define properties. It appears that the enhancement could be within, as extensions to, and compatible with current definitions.
YANG is a language used to define properties and it appears that YANG is appropriately formed to accommodate such extensions.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://example.com/LATEST. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-davis-netmod-modelling-boundaries/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:WG@example.com), which is archived at https://example.com/WG.

Source for this draft and an issue tracker can be found at https://github.com/USER/REPO.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 September 2023.

Copyright Notice

Table of Contents

1. Introduction

The essential challenge being considered in this draft is that of statement of partially constrained definition of a thing (property, collection of properties, entity, collection of entities etc.) and of progression through stages of refinement of constraints leading eventually potentially to precise single value forms of refinements of definition of that thing.

The constrained definition of a thing requires the expression of a boundary that surrounds all allowed/possible values for characteristics of that thing.

The draft will introduce the term “Occurrence” and explain that through the progression, a single Occurrence gives rise to multiple Occurrences at the next stage of refinement and that this expansion repeats from one stage to the next.

It appears that many aspects of the industries' problems/solutions require such a progression of gradual refinement and hence statement of partial constraint and of Occurrence. However, it seems that current languages do not readily accommodate the necessary structures. It is possible to use existing languages, but the realization seems clumsy and "bolted on" as opposed to inherent to the language.

Considering the apparent prevalence of need for expression of progressive refinement of ranges and uncertainty, it seems strange that there should be no readily available language, so part of the purpose of this draft is to stimulate discussion to help find an appropriate existing language. If, as appears to be the case, there is no well-suited language, then the next obvious step is to consider extension of YANG so that it can accommodate the need.

This draft works through an analysis expressed via threads of observation and exploration that are then woven together to form the fabric of the problem and solution.

In an appendix, a sketch of a YANG form of solution is set out to assist in the understanding of the problem. It is anticipated that the YANG form sketch will seed advancements in the YANG language in this area.

1.1. Observation: Terminology

We all recognize the challenge with terminology. Terms are for communication convenience (they are not fundamental). Unfortunately, each term comes with baggage and each of us has a different understanding of each term. Sometimes these differences are subtle, but sometimes our collective usage of a term spreads across a very wide space where each of us only aligns with a subspace of that usage.

Each key term used in this document has specific local meaning which the authors attempt to clarify. However, it is probable that the definitions here are too vague to ensure full shared understanding. Ongoing work will be required.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Analysis

The following section introduces concepts and associated terminology and gradually assembles a picture of the needs.

3.1. Specific challenge areas - some terminology

Each of the following require, for any property, expression of range, preference, uncertainty, interdependency etc. The specific challenges will be discussed in following sections.

The solution to the problem appears to need a language that supports expression narrowing of definition such that that language can be applied recursively to form a progression of increasingly narrowed definitions, from the very broad to the very specific.

3.1.1. Specification of "Expectation" and "Intention"

Specification of Expectation/Intention is a statement of desired outcome/experience in terms of constraints which includes statements of preference and of acceptable value ranges etc.

The specification has resulted from some negotiation (or imposition) where, in a simple case, a client and provider have formed an agreement and where the client has an Expectation that the agreement will be fulfilled, and the provider has an Intention to fulfil the agreement.

The expression of outcome/experience is as viewed from the outside of the provider solution, i.e., what is exposed by the provider and hence may be experienced (observed etc.) by the client.

Intention is a sub-case of "Specification of 'Capability'" and Expectation is a sub-case of "Specification of 'Need'" (note that 'Need' is not covered in detail in this document).

Related terms

* intent

* service

3.1.2. Specification of Capability

This is a statement of opportunity for behavior to be exhibited which includes statement of possible ranges and interdependencies etc.

The expression of capability of a provider system, presented as that of an opaque component, results from consideration, by the provider, of various potential assemblies of their internal capabilities (e.g., can be considered as a recursion of systems of components), governed by their business purpose etc.

It is becoming increasingly apparent that there is a need for a machine interpretable expression of capability, where the expression covers all detailed aspects of the capability, and hence a language for such expression.

Most recently, specific cases of need have been identified as automation solutions in the following areas mature:

* Advertising of solutions (both at a commercial boundary and for components inside a system where the advertisement is of abstracted capability)

* Negotiation towards contract (where capability requirement and offer are refined)

* Planning infrastructure buildout (where capability of solutions and of components is required)

* Intent solution formation (intent is the result of negotiation, expressing boundaries on the required solution capability)

* Any situation where specialized components need to be assembled (where minimally the capability for assembly will be expressed)

3.1.3. Expression of Partial Visibility

Any real environment will suffer imprecision, loss and disruption. Any statement made about properties (behavior, characteristics, etc.) of things in that environment may not be fully available (temporarily due to some impairment or permanently due to cost/complexity (the measure is an approximation as it is not practical to measure precisely)).

This leads to the need, for expression of any property, to include the opportunity to state absence, probability, uncertainty and vagueness (varying over the lifecycle etc. as will be discussed later).

3.2. Observation: Progressive Narrowing of definition

Traditional modeling tends to lead to a class model (potentially using inheritance), which provides precise definitions of properties etc. (current YANG models are essentially of this form), where each class is realized as instances in the solution and where each instance provides a precise value for each property defined as mandatory in the class etc.

However, what actually appears to happen in many areas of the solution is a process of gradual narrowing of definition where that narrowing takes place in a progression of discrete steps.

Consider the following rough example progression (stages may be omitted/repeated):

* A version of a standard may provide a definition of technology capability.

* A vendor solution may have a narrower capability than the standard, perhaps due to a target price point etc. A vendor may have several solutions, each with a different narrowing

* An application of the vendor solution may have a further narrowing of capability, perhaps due to some combinatorial effect in the deployment

* The use of a vendor capability at a particular point in a structure of a solution may have a further narrowing of capability as a result of need or policy at that point

* At a particular point in a structure of a solution under particular circumstances there may be an even narrower allowed capability, for example under certain environmental conditions the thermal considerations may require the solution to run at low power etc.

* Proof of concept (PoC) of the solution may have an even narrower allowed capability

* An example diagram related to a single use case for a PoC may require an extremely narrow definition

* Where there is delegated control, even the fully refined "instance", perhaps in the single use case for the PoC mentioned above, may be specified in terms of constrained definition as opposed to absolute value.

* Etc.

Traditional Classification and statement of instance specification do not deal naturally with the above. Some constraint mechanisms deal with the above in part, but these are often an afterthought, are clumsy and have significant shortfalls as will be illustrated.

3.3. Observation: Definition expansion

In addition to the recursive reduction discussed above, at each level definition may be introduced that did not appear in the previous stage as a result of capability from the intersection with a separate progression of narrowing. For example, the vendor solution may extend with proprietary features not defined in the standard, but instead derived from some related process.

Further, there will be evolution and growth so the next development of the standard etc. may extend/adjust the statements.

Of course, any definition introduced at any point in the progression can be narrowed at the next stage of the progression.

The ultimate progression is an intertwining of expansion and reduction stages.

3.4. Observation: Expression of capabilities

For any control solution component at any stage of the above intertwined progression, it is necessary to express the capability of the component and indeed some of the progression. The depth of expression of capability will depend upon the application, however, it is expected that it will be necessary to have a comprehensive statement expressing restrictions and constraints covering combinatorial effects within the solution.

The client is required to interpret capabilities expressed. To best facilitate this, the capabilities should be expressed in normalized uniform machine interpretable form. The capability statements should be available to the client at any stage of its operation. The capability statements could be published by the provider directly or indirectly via some other publication service.

As the capabilities of each instance of a function may differ and vary over time in a complex way, expressing restrictions/difference using traditional classification technique appears to be inadequate; traditional classification is too blunt a tool for this purpose as will be illustrated.

3.5. Observation: Application of expression of capability

In a control system context, capability expression applies to:

* the controlled system with respect to the exposed model and allowed activities

* the control system and its exposed functions (of both the control provider and control client)

Each of the above:

* is always an abstraction/view of the underling real system

* needs to cover any interaction at any boundary

The above may have further depth such, for example:

* the controlled system exposure can be controlled and adjusted

* the control system exposure can be controlled and adjusted

* etc.

The capability descriptions need to detail all deterministic per case variations (not just a broad-brush statement on the model versions supported).

3.6. Observation: Compatibility

The following applies to any interacting entities with respect to any aspect of interaction (e.g., a control system component interacting with another control system component about things that are controlled).

Note that here a component is a conceptual functional construct that has ports through which it can communicate with other components and that encapsulates some functional capability. Generalized component is described in [ONF TR-512.A.2]

Two components are suitably compatible and can interact with respect to a particular application so long as their exposed capabilities have an appropriate/sufficient intersection.

Interaction between Semi-Compatible Entities is possible where:

* Semantic intersection enables a subset of capabilities (resulting in a meaningful capability set).

* Partially mappable expression provides sufficient meaning (some mappings may be approximate and partially ambiguous, but only in areas where the this is not overly relevant)

The result of the intersection is usually a narrower statement of capability than the statement for the two components (at most it can be the full statement). In some cases, the intersection may be the empty set and hence there can be no interaction opportunity.

* Where a feature is preferred but not mandatory, the empty set intersection is acceptable

* Very few properties are fundamentally mandatory, importance is dependent upon specific application and specific interaction within that application.

For any interaction between two components A and B compatibility is determined by the intersection of:

* application interaction semantic

* interface transfer capability

* component A capabilities/needs

* component B capabilities/needs

If any of the mandatory application interaction semantic elements are eliminated by the intersection process, then the interaction cannot be supported. If preferred or optional semantic elements are eliminated, then the interaction can be supported at some degree of degraded capability.

Note that this discussion fragment focusses on the direct interaction and not on the implications for other interactions etc.

Compatibility must be considered over the intertwined lifecycles of the interacting components as each independently evolves in terms of both functional capability and interface expression. This also includes migration of boundaries.

3.7. Observation: Defining the boundary

The general problem identified is the representation of a semantic space by defining its boundary. Clearly, the boundary is itself defined in terms of ranges, however, the boundary is not necessarily defined with absolute range values, and it is not necessarily fixed in time.

The boundary may, for example,:

* change, in position and in precision, through the lifecycle of the thing (as it matures... where it tends to become tighter)

* be interdependent with other boundaries

* have uncertainty in position and/or limited interest in positioning (don't know, somewhere round about here, don't care, that's precise enough)

* have specification (and measurement) of acceptable, degraded and unacceptable positioning (there is also a need to indicate other aspects, e.g., for how long a particular degradation is acceptable or what the degradation costs etc.)

* have associated probability (likelihood of a particular position) and preference (for a particular position)

The above considerations apply similarly to intent specification, capability specification and partial visibility.

The same challenges also appear in planning and in negotiation where there is a need to state vaguely understood and interdependent properties etc.

Considering lifecycle stages, a property defining a boundary of a thing, e.g., the property acceptable temperature range, may have one defined range at one part of the lifecycle and another defined range at another part of the lifecycle where this variation is known at the time of specification such that the specification needs to include this lifecycle aspect.

The variation may be temporal, as note above, or may be dependent upon some other variable property.

3.8. Exploration: The nature of the solution

Considering the observations and other considerations above, the solution requires support for a property to:

* be stated in terms of ranges with focusses and complex (potentially fuzzy) boundaries where that statement defines a semantic volume and where the boundary may not be sharp

* have statements that interrelate it with another property (or properties)

* have multiple boundary preference levels and/or probability levels where the preference/importance level is per interaction and not an aspect of fundamental property definition

* be defined in terms of a narrowing of a previously expressed volume (i.e., a further narrowing) where a single point value is a very narrow range (many single values are actually abstractions of complex ranges, e.g. 2Mbit/s is +/-15ppm)

* be defined in such a way that simple property definitions are not burdened by the structures that enable sophisticated definition (i.e., the expression should be such that the complexity of expression "folds away" for simple statements)

* use a representation where there is no distinction in expression opportunity between a statement of capability definition, intent definition, actual value etc. such that all expressions are of the same essential form

This is a fundamental change in the nature of the solution... a change in paradigm and metamodel where the properties are defined by complex/fuzzy bounded/focused spaces related to other complex/fuzzy bounded/focused spaces with preferred/probable positions etc.

3.9. Clarification: Complex blurry fuzzy boundaries

To illustrate the complex/fuzzy boundaries, partial compatibility and several other aspects, an example of color usage is helpful. Whilst color may not be the most important aspect of the solution in key industries related to this work, it is an easy example to understand. It is suggested here that the same challenge applies to ALL properties at least to some relevant degree.

Consider a request for a physical item that has a color where the requestor, whilst interested in the color is not overly concerned.

Their request for the item may include an expectation on color where that expectation is that the choice of color is not a showstopper but there is a preference for red and if not red then green. The request does not need to include the color and if not included a choice will be made by the provider on some basis outside the interaction.

The provider may not know what red is but may know green and have the item in green which will be appreciated by the client.

Even if the provider does not have green any color will do. In fact, the provider, in its response, need not even say what the color actually is!

However, if the client indicates that the item provided must not be pink, then unless the provider knows what pink is, it cannot satisfy the request and the boundary now has a hard edge, so an aspect of the request is now mandatory.

In this case, assuming the provider does know what pink is, the provider could respond with "the color is not pink" but provide no more details.

In the example above the definition of color was complex/fuzzy to some degree, the providers understanding of pink may not match the client's exactly and the definition if the boundary between pink and red may be unclear and vary from occasion to occasion.

The color was specified by the client using colors by name as enumerated literals. Now consider that the provider understand color in RGB. It is possible that the color Red is not just 255,0,0, but is actually a range of colors in a volume bounded on the red scale by 255 at one extreme and 240 at the other. Further, red is a complex volume including on its boundary 255,10,10 and 250,8,8 etc.

Now when the client asks for red, the provider can select color in the range defined.

But it may also be the case that the color is not perfect so that there is always a little green and blue somewhere between 2 and 3 on both scales and the red coloration process is inaccurate so that the color produced is somewhere between 253 and 250.

Further the color fades a little over time and in some lights looks a little more bluish. These factors may need to be taken into account (as interactions between properties) if the request is for red and the duration of use is to be 10 years where the usage will be in various different lights.

So, the request for red must be qualified by the above. In a negotiation the requestor may even have broadened their view of red to include some maroon shades in their preference for Red, so that may now be a list of similar colors etc.

The example above illustrates the need for the opportunity to specify range and interrelationship as a fundamental aspect of specification of color. The color attribute needs the opportunity to deal with the above within its scope, not as a pile of arbitrary other properties.

On the measurement side, it may not be possible to distinguish between anything within a range of 255 and 252 red etc. and further if the light level is low the color measurement may dither etc.

In general, when a specific value is specified, e.g., "A" must equal 5, this equates to a fuzzy setting that has hard boundaries.

It is argued here that the above consideration applies to all properties.

3.10. Observation: Artificial Intelligence and uncertainty

As the spread of system automation progresses, the problem becomes increasingly complex. This leads to the necessary expansion of use of AI/ML techniques in the solution.

These techniques deal well with uncertainty, approximation and fuzziness unlike traditional systems that tend to only work as precise coded solutions and absolute values with the occasional specific hack to deal with ranges and approximation.

AI/ML solutions would benefit from the opportunity to express range, uncertainty etc. in any/all values/structures and to see clarity of uncertainty in all inputs. Considering that the problem space is one of range, preference, approximation as discussed, it seems fundamentally necessary to expand the opportunity for expression as discussed in this document.

3.11. Observation: Optimization under uncertainty

As networks advance in scale and complexity, interest in the optimization of their design and deployment naturally follows. While "classic" forms of optimization work with point values for the parameters that define the optimization objectives and constraints, there continues to be important parallel development of methods that work with quantifiable forms of uncertainty in the problem parameters. The approach of robust optimization for example typically seeks the best solution that is still feasible even when problem parameters achieve their worst-case values within a given range of uncertainty. A probabilistic relative of these techniques (in some contexts referred to as stochastic programming), can be applied when the problem parameters follow a given probability distribution over their range.

An important example parameter type that plays a central role in telecom network optimization is the matrix of source-destination traffic demands (i.e., "the traffic matrix").

In real-world settings, each value in this matrix carries with it both measurement uncertainty as well as its intrinsic statistical variance. On intra-day time scales modal variations (e.g. day-night peaks and troughs) can be observed overlaid with (typically short)

episodes of "anomalous" traffic. In certain longer time scale contexts, the intraday variations are less of a concern while longer term growth trends (and uncertainty around how to extrapolate from current trends) are of primary concern.

The above situation is a strong example of the value of working with ranged values, in this case representing the associated intrinsic variance and/or estimation errors of the traffic matrix parameters. Doing so allows sophisticated forms of robust and stochastic optimization to be applied that would not be possible if only point average estimates were retained and available for optimization and other purposes.

Apart from traffic matrix, models of endogenous risk (e.g. equipment defect) or exogenous risk (e.g. regional flood/fire/power outage) if quantified statistically can factor together into optimizations in a way that allows management of worst-case scenarios while pursuing primary optimization objectives. This ties in as well with observation that advanced AI/ML applied to suitably dense datasets provides probability distributions of the problem parameters, expediting application of robust or stochastic form of optimization as outlined above.

Summarizing, from the above there appears fairly clear value from an optimization perspective of network models that facilitate and invite the expression of ranged parameters.

3.12. Observation: Everything is expectation-intention including instance config

The idea of Expectation/Intention as discussed earlier can be further developed. Consider an example where a client has an expectation that the integer property "A" (perhaps a temperature of an "oven" containing a component that needs to operate within a particular range) is to take a value between 5 and 10 (with some unit, e.g., Celsius. It is OK to leave the units open when there are no specific values, but once values are being expressed, the units MUST be provided). The provider could have the intention to make A take the value between 5 and 10 as requested, but to achieve this a complex process needs to be performed so it will take time to achieve a value in the range and there is some progression that needs to be reported.

Eventually the provider achieves a value in the target range, but it is unable to state the value precisely as there is high-rate jitter and hence it can report that the value is between 6 and 9.

The above example reflects the need to be able to state and report ranges for a property.

Now consider the case where the system is more precise. The client requires and has the expectation for A to take the value 5 (which is simply a very narrow range from 5 to 5) and the provider has the intention to achieve this, but again this will take time. The provider reports progress towards 5 and eventually reports that 5 has been achieved.

The above example reflects the need to be able report convergence on a property value even where the value is simple. In general, the client may want to state a maximum time allowed to achieve any specific outcome and/or the provider may want to state a predicted time for any specific interaction.

Finally consider a case where the system has greater performance. The client requires and has the expectation for A to take the value 5 and the provider the intention to achieve this. The value can be achieved immediately, so the provider can simply report this back directly. In this case the provider would predict an effective delay of 0 seconds (which can be implied as the value is returned immediately).

The final case could be viewed as the operation SET of the property of an instance of a class and hence special, but it is no less of an intent-expectation case than any of the others. Indeed, it is possible that for a particular specific intent-expectation, on some occasions the achievement is immediate and on others it takes a while and for some parts of the range of possible settings the value is precise, but for other parts it is a range (perhaps at the extreme ends of operation).

Clearly, it is highly beneficial (and even arguably, necessary) to have one uniform representation that caters for all cases. Ideally, this method would appear as light as a SET where the value is precise and the achievement is immediate but would deal with the sophistication required where the value is a range, the result is a sub-range, and it takes time to achieve the result.

Assuming that such a representation is achieved, then a traditional "instance specification" is nothing more than a sub-case of intention/achievement (or "intent" as defined by rough common usage) and hence not something distinct. Indeed, the notion is that an instance is simply an Occurrence (see later for further explanation of Occurrence) at the most extreme narrowing, the lowest and most detailed available view, of definition and as noted above, this lowest available (visible) view of a realization may not be precise. There are always many details "below" this "lowest available visible view" that are not exposed (as they are not of interest and are essentially noise).

Any expectation/intention statement expression may have a mix of degrees of tightness of statement from vague to single value (and hence suitable to use for all cases of "instance specification") and allow representation of a mix of ranges and of single values.

3.13. Observation: Intention-Expectation interaction

Clearly, a solution does not operate on a single requirement in isolation, there may be multiple agreements and hence multiple Intention-Expectations competing for the solution resources. Within the expression of each Intention-Expectation there is a need to state importance and this will interact with preemption policy.

3.14. Observation: Intention-Expectation layering

As a result of a negotiation between two interacting systems, the provider commits to intend to supply some capability to the client (which the client expects). As noted earlier, negotiation may degenerate into imposition. To supply this capability, the provider needs to use the services of underlying systems and hence it is the client in interactions with these other systems (client and provider are roles with respect to a specific interaction, provider is not an all-encompassing role of a system etc.). Where a provider is adding value, it will combine capabilities of its providers in such a way as to provide to its clients the necessary capability. This combination is complex and outside the scope of this document. At the lower parts of this hierarchy will be real physical providers (at which point the intention-expectation relationships degenerate into the relationships of physics).

Note that here a physical thing is something that can be "measured with a ruler".

3.15. Observation: Instance state

As discussed above, an instance is an Occurrence at the lowest and most detailed available view at the extreme end of the narrowing. In addition to state related to progression of achievement of expectation/intention (traditional "config"), there is also state related to monitored/measured properties of the solution (not directly related to config).

These properties are derived from monitoring devices that perform some processing of events within the solution. The events are detected by a detector. Very few of all of the possible event sources are monitored by detectors.

All detectors are ultimately imprecise and may fail to operate. The information from a detector may be temporarily unavailable, delayed, degraded etc.

The representation of the simple detected value should include qualifications related to its quality etc.

A machine interpretable specification of capability for the property should provide details of its derivation from other less abstracted properties. For example, there may be a property that is detected where the detections are counted over some period and compared with a threshold where the crossing of that threshold is reflected by another property that is itself counted and compared with another threshold that if crossed changes the state of the property of concern. An example of a property resulting from this pattern is the Severely Error Second alert. In this example, errors are detected and counter over a second. If there are more than a particular number of errors in a second, the second is declared as severely errored. The number of severely errored seconds are counted over a period (perhaps 15 minutes) and if more than a particular number of severely errored seconds are counted in that 15 minute period, the severely errored second alert is raised.

Understanding this pattern and other related patterns would enable a control solution to interpret the relationship between the various properties (currently, at best, solutions are explicitly coded to deal with properties with human oriented similarities).

3.16. Observation: Foldaway complexity

It was noted in an earlier section that "Ideally, this method would appear as light as a SET where the value is precise and the achievement is immediate but would deal with the sophistication required where the value is a range, the result is a sub-range, and it takes time to achieve the result.".

In general, it is highly desirable for the representation of common and simple cases to look/be simple and not be burdened by the more sophisticated structures that allow for more complex cases. Ideally the representation has "foldaway complexity".

An analogy can be drawn with human language grammar where the structure that allows sophisticated speech is not visible in simple speech.

Several sketches (in rough JSON) of a configuration statement for a property "temperature" follow.

```
   Basic:

     "temperature":"5",

   More versatile:

     "temperature":{
           "acceptable-range":"5-12",
           "preferred-range":"7-9"
     }

   More sophisticated:

     "temperature":{
           "acceptable-range":"5-12",
           "preferred-range":"7-9",
           "upper-warn-threshold":"11",
           "lower-warn-threshold":"6",
           "Fail-alarm"{
                 "less-than":"5",
                 "greater-than":"12"
           }
     }

   In this example the schema for:

     "temperature":{
           "acceptable-range":"5-12",
           "preferred-range":"7-9"
     }
```

would identify preferred-range as optional, would identify "acceptable-range" as mandatory and the primary property and would identify the foldaway nature if only one value is provided in the range:

```
  "temperature":"6"
```

is conformant with the schema.

In addition, in a simple case a subset schema could be designed that was compatible with the main schema but that only allows the single value temperature.

Ideally, considering the common requirements across all properties, the terms used in the schema nested within the property name (where an example of a term is "acceptable-range") would be standardized etc.

3.17. Exploration: Focusses boundaries and partial descriptions

Considering the progressive narrowing of boundaries, it is possible to consider anything as represented by a fully capable generalized thing with constraints (everything is a focused thing). It is also possible to consider a subset of things where the focus is thing with ports. Not all things have ports. A thing with one or more ports can be called a component. The component is a thing which has ports. Anything that has ports is a component. The boundary around this focus is the presence of ports.

A physical thing is a thing that can be measured with a ruler. Some, but not all, things that are physical that can be measured with a ruler have ports.

A functional thing is a thing that exposes behavior. A functional thing is not physical, although it must be realized eventually by physical things and hence physical things have associated functional things. Functional things have ports, as this is where the behavior is exposed (although the ports need not be represented under some circumstances).

So, there is a set of physical things disjoint from a set of functional things and a set of components that has an intersection with both the physical thing set and the functional thing set where the functional thing set is a subset of the component set.

Anything that has ports is a component (as per the component-system patter described in [ONF TR-512.A.2]). Many components will not yet be known etc., but the semantic of "component" remains unchanged when

they are discovered/exposed. As not all components are known, not all properties that can apply to components are known. Adding properties does not change the semantics of "component", but it does improve the clarity.

The progression above is essentially, specialization by narrowing of focus. The broadest "Thing" has all possible characteristics and capabilities (including many unknown characteristics and capabilities). Specific semantics relate to the model of a specific thing where that is derived from the narrowing of the broadest thing. The definitions of things do NOT need to be orthogonal/disjoint.

Consider the thing "Termination". The Termination:

* Covers all aspects of "carrier" signal processing

* includes recursive definition of encapsulated forwarding

* includes all possible properties of termination for any signal (including those not yet defined)

* includes capabilities to extract signal content and further adapt to another signal

* etc.

3.18. Observation: Two distinct perspectives and viewpoints

Considering a system taking a provider role, there are two distinct perspectives

The external perspective (the effect) - "exposed"

* Capability (advertised to enable negotiation and selection)

* Intention (the agreement resulting from the selection at the end of negotiation)

* Achievement of intention (causing the client to experience the desired outcomes)

The internal perspective (the realization)

* Realizations (alternative system design approaches to achieve exposed capabilities)

* Specific chosen realization (the system to be deployed)

* Actual realization achievement

Both perspectives are expressed using the same model pattern and model elements, i.e., the component-system pattern, where a component is described in terms of a system of components and components are specialized as appropriate (as discussed earlier).

There are two viewpoints:

* that of the client where only the external perspective is available

* that of the provider where both the internal (which is private) and external perspective are available

The provider also has control of the mapping between internal and external perspective.

Note that:

* the provider system may have internal components with distinct roles where each has access to a subset of the provider viewpoint.

* the perspectives and viewpoints apply to both the capabilities being controlled by the system and the capabilities supporting the control system (which are controlled by another system).

* the external perspective relates terms defined by TM Forum to "Customer Facing Service" (which is essentially what is exposed to the customer) and the internal perspective to "Resource Facing Service" (which is essentially how it is realized (roughly))

* At any arbitrary demarcation, the same approach may be applied from the broadest inter-business demarcations right down to demarcations deep within a device

* The actual chosen demarcation may shift as the solution is developed and as it evolves

* This can be stated as how it looks from the outside (as an opaque box - not "black box" in the strictest of senses as internal behavior can be sensed from the outside) and how it looks on the inside

This is discussed further in [ONF TR-512.8] where it is noted that a client talks through a provider port to the provider control functions about the controlled system where that controlled system is presented as a projection that has been mapped to an appropriate abstraction of the underlying detail.

3.19. Observation: Capability in more detail

A Capability statement is the statement of visible effect of a thing and is not a statement of the specific realization of that thing. The visible effect may be complex. The thing may have many ports and activity at one port may affect activity at another. Capability statements will include performance and cost (environmental footprint etc.).

It is important to recognized that the statement of capability is NOT exposing intellectual property related to how the capability is achieved.

Expression of externally visible capability and expression of realization of that capability can both use a model of the same types of things, but the specific arrangement will often be very different as the externally visible capability is a severe abstraction of the internal realization (as the externally visible capability is emergent from the intricate assembly of the capabilities of the realization) where a subset of the capabilities of the underlying system are offered potentially in different terminology and in a different name space.

The approach of expression of capability can be applied recursively at all levels of control abstraction from deep within the device to the most abstract business level.

3.20. Observation: Occurrence

A system is constructed from components. Often a system will make repeated use of the same type of component where each usage will be distinct from each other. Whilst in a realization of that system the components are considered as instances, in the design, they are clearly not instances. But there are many. The term "Occurrence" has been used in ONF work (see [ONF TR-512]) as a term an item of the many.

In a component-system approach, an Occurrence is a single use of a particular component type in a system design structure. There may be many uses of that type in that system design structure, and hence many Occurrences, where each use of that type may have subtly different narrowing of capabilities to each other use and certainly different interconnectivity.

Capability, intent and realization are all specified in terms of system structures and hence all require the use of Occurrence.

Considering the "progressive narrowing" observation earlier in this document, what is a singular thing at one level of narrowing may result in several separate things at the next level, each of which is a slightly different narrowing of the definition of the original singular thing. These things are Occurrences.

Hence, the progressive narrowing starts with a single Occurrence of thing at the first level and splits this into multiple Occurrence at the next and then each may split at the next etc.

Note that:

* the pictures of devices in a network structure example diagram are essentially Occurrences.

* the presentation of an instance in a management view can be argued to also be an Occurrence.

* that through each progressive narrowing of definition, what was a single "type" at one level of narrowing may cause many "occurrences" at the next level.

* a type is simply an Occurrence with a particular definition where that Occurrence is used in the next level of definition as a type.

3.21. Observation: One model

From a model perspective there appears to be no relevant distinction between expression of what can be requested and expression of how that can be achieved. A single model representation of the things and their effect, based upon the recursive/fractal component-system pattern appears appropriate.

The intention/expectation is expressed in terms of a structure of occurrence and how that is realized is in terms of a similar structure of occurrence (where at the extreme the structures are exactly the same as the exposed capability was expressed at the finest possible granularity exposing everything).

There are however several stages and consequent perspectives:

* the original request (the initial expectation retained by the client and progressed through negotiation to a final agreement)

* the accepted request (the intention, retained by the provider, after agreement (normally the same as the expectation) retained by the client after the agreement)

* the achievable outcome (expressed by the provider, normally the same as the intention)

* the current realization as exposed to the client (more precise than the intention as the constraints of the intention are not absolute single values and hence the current settings/states are usually more precise)

* the actual realization as understood by the provider and not exposed to the client (more detailed and significantly different in structure to that exposed to the client and sufficient to fully describe the solution that realizes the agreed outcomes)

* the effects of the realization in the perspective of the original request (achievement)

* the difference between the client expectation and the achievement (discrepancy)

For example:

* the client may wish to request an E-Tree with particular characteristics including endpoints, bandwidth, temporal characteristics etc.

* the provider may accept this minus one endpoint.

* the provider may not be able to achieve the accepted request initially as some hardware is not yet in place

* the realization will provide subordinate details.

* the effect of the realization can be abstracted back, freed from some irrelevant detail, to form the achievement (reflecting the details of the original E-Tree request)

* the provider can represent the differences between the original request and the achievement

For all of the above, the key model elements are a multi-pointed connection and a set of terminations. The detail of the realization is supported by a recursion of multi-pointed connections. There is no reason for different representational forms at the different stages of development.

3.22. Observation: Partially satisfied request

The original request from the client may not be satisfiable

* during the progression of activities formulating the solution and acting on that formulation

* initially, although it may be later

* at some intermediate stage in the lifecycle, although it was initially and will be again shortly

* ever

Where there is knowledge of a temporary shortfall, expression of what is achievable as a lifecycle of related statements appears necessary and beneficial. Parts of this lifecycle appear to be definable within individual properties using the mechanism in the metamodel suggested by the various observation here.

3.23. Observation: Other solution elements that benefit

The progressive narrowing approach that yields levels of Occurrences where those Occurrences are defined in terms of semi-constrained properties (as discussed in this document) appears beneficial in the construction of:

-Policy (as per general definition): The condition statement could benefit from a generalized metamodel approach to range etc.

* Profile/Template (for all the various interpretations of these terms): Various methods that support the specification of constraints in a single statement to be applied multiple instances simultaneously.

The constraint statement would benefit modeling in general. For example, in UML, OCL is an add-on that tends to be "beyond" the normal model. An advancement to the essential metamodel would inherently include interaction constraints etc.

3.24. Observation: Outcome and Experience

The term Outcome is being used here to relate to the apparent/ emergent structure/capability made available by the provider to the client and the ongoing behavior of that structure/capability.

The term Experience is being used here to relate to the client perception of the effect that the provider Outcome has (i.e., the client perception of the Outcome). It can also be argued that the client Experience is the client's Outcome and that the provider Experience includes the feedback by the client on their overall experience of the provided capability etc.

An Outcome/Experience may be a:

* Momentary event or set of events (and the client's perception of the effect of the event(s))

* Constant state or set of states over time (first order... and the client's perception of the effect of the ongoing constant state(s))

* Constant change of state or set of state changes over time (second order... and the client's perception of the constantly changing state(s))

* ... (nth order)

* Change of state defined some algorithm or set of changes of state defined by a set of algorithms (complex landscape... and the client's perception of traversing this landscape_

* Etc.

Both outcome and experience can be expressed in using the same approach discussed in this document (although the model of experience will be a significant abstraction, ultimately emotional).

In the connectivity example discussed earlier, considering the client:

* the expectation for the Outcome is an E-Tree (a resource!)

* the Experience is effect of the E-Tree which is complex apparent adjacency (the true "service", a change of apparent proximity and the feeling of closeness and immediacy).

3.25. Observation: Metamodel v Model

The concept of metamodel involves a model that constrains one or more other model(s). The term metamodel is essentially about the role of a model in the relationship to other models. The model with the metamodel role when related to another model influences that other model and is specifically designed to do so.

A model taking a metamodel role may express how another related model may express properties to be represented.

Clearly a model that takes a metamodel role is just a model and hence may have a related model that takes a metamodel role for it etc.

Note that meta means "providing information about members of its own category" (Merriam Webster) where, in this case, the consideration is the category of models.

4. Solution: Formulation

The following subsections consider the observations from the earlier sections and point to aspects of a potential solution.

The first few subsections consider the solution in general independent of specific realization. The final subsection considers the applicability of YANG. Some considerations in the realization independent sections are already supported by YANG, others are not.

4.1. Solution: Methodology

Each type/structure/property is specified in terms of constraints narrowing prior definition/specification. Note that this is a common approach, but the recursive nature is not well represented, and each level of the recursion tends to seem special (whereas here each level is considered just another level in the progression).

Examples are as discussed earlier:

* A standard may narrow an integer range

* A usage may narrow the standard integer range

* Etc.

The prior definitions must be explicitly (efficiently) referenced. Note that this is a common approach but the specific usage here is distinct from normal usage. Examples relate to earlier discussion:

* A prior definition may be used for several Occurrences in the same specification

* A definition syntax may be transformed, but the semantics cannot be changed (shifted etc.), they can only be narrowed

* A property may have preferred values etc.

* Etc.

4.2. Solution: Considering the property

Extending the approach could lead to a more uniform specification of properties in a control system context.

Any property, e.g., temperature, may have combinations of the following features:

* A detector: Allowing opportunity for approximate value, unknown value, value range etc. and allowing notification of change with definable approach to hysteresis etc.

* An associated control: Which has intent, achievement etc. and, especially where it takes time to take the control action, may have some progress on the action etc.

* Threshold based alert: Which has intent (as above) and which has an associated state (allowing opportunity for approximate etc.), notification etc.

* Have inter-property interrelationships for any of the above

* Have Units for any of the above

* etc.

4.3. Solution: Occurrence Specification formation

Any property and its range of opportunities is stated in a specification. Any invariant values in the specification need not be reported in the state of the "instance" (unless the instance is no longer behaving as defined in its specification).

Ideally the metamodel should be such that, when a model designer chooses to define a property, they pick which of the features, sketched in the previous section, are relevant and need not specify each separately. This would lead to automatic name generation etc. where the name structure can be predefined.

A uniform way of expressing each of the above features could be developed as could tooling to generate the representation (e.g., YANG) for each feature given a property name. The application of each feature to a property is essentially the occurrence of the feature.

4.4. Observation: Uniformity of expression

A uniform expression for Occurrence could be applied at all levels of the progression of narrowing/refinement from the statements of most general capable to the statement of the most resolved form (the traditional instance). This expression form must allow for a mix of constraints and absolute values. Ideally, each statement of absolute value will be as compact as it is in a traditional instance language and each statement of constraint will be as clear and concise as possible.

A uniform expression form at all levels will maximize tooling usefulness and minimize the need to learn varieties of structure patterns etc.

4.5. Solution: Tooling support

Clearly, tooling support will be vital for any initiatives in this area to be successful. The following areas would benefit from tooling:

* model pattern construction
* occurrence formation supporting derivation from less refined occurrences, formation of rules etc.
* property expansion and formation of rules for expanded properties
* validation of occurrence conformance
* occurrence visualization
* generation of code supporting occurrence
* etc.

5. Target and next steps

There does not seem to be readily available terminology to label/ define the concepts in the problem space

* Hence it has been difficult to discuss what properties the language needs to possess.

* Action: Improve terminology definitions

It appears that there is not a good language suited to solve this problem fully.

* This may only appear to be the case, i.e., there may be a language out there (as it has proved very difficult to describe the problem)

* Action: Continue to explore and refine

It is possible that YANG could evolve to be more suitable

* YANG does not have all the necessary structures or recursion

* Progress sketch for a JSON form of YANG as an illustration of the unification of class and instance statement representation. Work the proposal to suitable maturity (requirements first)

6. Conclusion

Tackling the challenge of modelling of boundaries leads to a more complete method of specification of gradual refinement of definition and of statement of "occurrences" including classes, instances and various forms between.

This method promotes:

* Expression of range, preference and focus as a fundamental part of the metamodel

* Gradual refinement and recursive tightening of constraints as a native approach of the modeling technique

Gradual refinement is required in many areas of the problem/solution space and this more complete method naturally allows for the necessary representation of uncertainty and vagueness. The rigid boundary representation of the current approaches (e.g., class, instance) is accommodated within the method as a narrow case of application of the method.

This softer and more continuous approach to specification refinement with the opportunity for uncertainty, ranges and biases better describes any real-world situation and hence appears more appropriate for an intelligent control solution (using AI/ML etc.) where that solution could take advantage of partial compatibilities etc.

It appears that the enhancements to the language metamodel could be within, as extensions to, and compatible with current definitions of YANG as it appears that YANG is appropriately formed to accommodate such extensions.

Note that the problem appears in expression:

* Intent
* Capability
* Partial Visibility
* Planning
* Negotiation
* Policy
* Profile/Template
* Occurrence
* Etc.

7. Security Considerations

None

8. IANA Considerations

This document has no IANA actions.

9. Informative References

[ONF TR-512] TR-512 Core Information Model (CoreModel) v1.5 at https://opennetworking.org/wp-content/uploads/2021/11/TR-512_v1.5_OnfCoreIm-info.zip (also published by ITU-T as G.7711 at https://www.itu.int/rec/T-REC-G.7711/recommendation.asp?lang=en&parent=T-REC-G.7711-202202-I)

[ONF TR-512.A.2] TR-512.A.2 Appendix: Model Structure, Patterns and Architecture (in Model Description Document within [ONF TR-512])

[ONF TR-512.8] TR-512.8 Control (in Model Description Document within [ONF TR-512])

[ONF TR-512.7] TR-512.7 Specification (in Model Description Document within [ONF TR-512])

[ONF TAPI] ONF Transport API SDK 2.4.0 at https://github.com/OpenNetworkingFoundation/TAPI/releases/tag/v2.4.0

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Appendix A. Appendix A - Problem/Solution Examples

This appendix sets out some examples to illustrate usage of the capabilities set out in the main body of this document.

A.1. Temperature sensor

Temperature is a general property of a physical environment where there are several different scales and temperature, like all properties, may be influenced and observed. The property temperature can be applied to both measurement and control, when considered for measurement, the property is read only. This is the first of a recursion of narrowings.

An ideal temperature sensor would operate over the entire range of all possible temperatures but would be read only. Any particular type of real temperature sensor operates within a sub-range of all possible temperatures and with particular resolution, precision and accuracy (that might vary with temperature). The property temperature, when used as part of the specification of a real type of temperature senor will need to be bounded and constrained so as to define these ranges. Perhaps the range is from -50.535 C to +100.0012 C (note the chosen units will also need to be identified) with a resolution of .001 C for the lower temperatures worsening to a resolution of .003 C with following a specific equation etc. As the

specification applies to a a set of real manufactured things, and recognizing that the manufacturing process is imprecise, the values will also have to be toleranced. So, the upper temperature may be +100.0012 C +/- 0.001 C. This is further narrowing.

In a particular usage of the temperature sensor, the valid range may be further narrowed so whilst the temperature sensor may have a significant negative range, the usage may not take advantage of this and it may be constrained to express temperatures from +10 C to +50 C, and the precision further constrained so that it only exposes a single decimal place. A further narrowing.

This is a specific case of the general case of specification of a detailed capability.

A.2. Temperature sensor features

It is likely that in some designs for use of the specific temperature sensor additional features will be provided such as threshold crossing alerts and upper/lower watermark etc. In this case ideally, when building the specifications to describe the temperature sensor and its associated features, the property temperature would have been expanded using the "Occurrence Specification formation" method described earlier where the designer would "pick which of the feature patterns are relevant". The expectation is that the designer would simply reference each pattern in the specification and "need not specify each separately". The various feature patterns selected during the specification design would be represented in the expression of the Occurrence in an abstract form and expanded at run time.

The patterns would lead to Occurrences with explicit threshold properties, but these would be of a normalize form from case to case. The same threshold pattern could be used for a temperature measure and a power measure where the pattern yields property names. For example, "temperature-high-threshold" and "power-high-threshold".

This is a specific case of the method of definition using patterns as opposed to explicit expansions.

A.3. Temperature sensor application

Considering the temperature sensor discussed above, it is possible that in a design several sensors of the same type may be used. These occurrences may all have the same roles in the design, or they may have very distinct roles. For example, there may be four sensors of the same type on a specific type of circuit pack. In the design, one of the sensors may be placed at the bottom and one at the top of the

circuit pack, as the circuit pack is to be used in a force air cooled rack. The bottom sensor is measuring in-flow temperature and the top one out-flow. These two sensors may operate over the same range etc. and both reference the same constraint specification, although they are clearly distinct occurrences and have different position-role names.

Each of the other two sensor occurrences may be used in another role to measure the specific temperature of a particular sub-unit in the circuit pack, e.g., a CPU, where in this case there are two CPU occurrences. The application of the senor to a CPU brings a different set of constraints. There may also be a need for an alert when a specific temperature has been exceeded etc. These two sensor occurrences have the same spec as each other but not as the first pair discussed above. In this case ideally, the property temperature would have been defined using the "Occurrence Specification formation" method.

It is possible that not all of the features specified for the sensor are used in the application or perhaps some features are used in a constrained form. In this case, a further narrowing is performed, and new, more refined occurrence specifications are constructed. When a client interrogates the system for its capabilities, the various occurrence specifications that provide details of the features will be reported.

Thus is a specific case of the general case of narrowing of capability.

A.4. A circuit pack

Clearly, it is vital to not just know the components and their features, but also the arrangement of those components. There may be several dimensions to the arrangements including physical positioning and functional positioning where the latter would need to describe interconnectivity between occurrences and flow of signals etc. between the occurrences.

In the case of the temperature sensors discussed above these were considered positioned on a circuit pack and related to units of functionality on the circuit pack. The sensors of air flow temperature may simply require physical positioning data in the occurrence specification. The CPU sensors will require a stronger relationship to the specific CPU. Neither require any functional modelling.

Consider a circuit pack with traffic functional capability. This circuit pack will have occurrences of interface, termination-point, potentially vrf etc. these will all support some subset of the corresponding standards, may support some proprietary capabilities and will be arrangeable in various configurations depending upon the specific capabilities and restrictions of the circuit pack.

There may be several distinct occurrences of a specific type of termination point and several occurrences of another specific type. Each of the types can be defined as an occurrence and then each of the specific occurrences of termination-point can reference one or other of the type occurrences.

There may be two specific occurrences of termination-point that have a fixed relationship by design. This can be set out as part of the circuit pack type occurrence spec using a simple relationship rule occurrence (e.g., a "must connect" rule). There may be semi-flexible relationships between a collection of occurrences, and these can be set out in the circuit pack type spec using relationship rule occurrences. For example,

* any interface of occurrence type "trib" may connect to any interfaces of occurrent type "line"

* any interface of occurrence type "line" may connect to any interface of occurrence type "line"

But occurrence type "trib" may not connect to occurrence type "trib" so that rule is not stated.

The approach described for representation of connectivity restrictions is partially supported by [ONF TAPI] and is described in [ONF TR-512.7] in more detail.

There are many more examples such as a circuit pack with a fixed mapping that supports a narrow subset of the capability defined in the relevant standards.

This is a specific case of the general case of specification of a capability.

A.5. A slot in a shelf

For device with a backplane (an equipment) that has slots that support field replaceable circuit packs (equipments) and for a circuit pack that have slots that support SFPs (equipments) it is usually the case that several different types of equipment can be plugged in any particular slot. To cater for this, the equipment, e.g., the backplane, specification would identify slot occurrences and for each would reference the specification for each compatible equipment. In some cases when a particular equipment type is inserted in a particular slot its capabilities are reduce, perhaps due to thermal considerations or access to the backpane bus etc. In these cases, an additional stage of narrowing would be applied to create a new equipment specification for the particular slot application.

It is often true that some functionality emerges from specific combination of circuit pack. Here a combinatorial specification would be created to explain the emergent behavior from the combination.

This is a specific case of the general case of specification of compatibility.

A.6. Temporary loss of visibility

In some applications the reporting of a particular detector may be mandatory. It is possible that under some circumstances the detector value cannot be read (e.g., due to a communication issue inside the NE, because the unit that hosts the measurement is restarting etc.). Under these circumstances it is necessary to report that the detector value is not available. This should be within the normal property definition and ideally the YANG metamodel would allow a string statement "value not available" or similar in a field that is defined as an Integer or as another more complex structure.

This is a specific case of the general case of loss of visibility.

A.7. Degraded detection

Considering the detector further, it is possible that some environmental factor temporarily or permanently degrades the quality of a detector output. It is important that this degradation is reportable as an aspect of the detector output. Ideally this degradation would be conveyed as part of the detector value and that the method for conveying information on the degradation would be part of the metamodel for the language (as discussed in elsewhere in this document).

In some cases, the value may be used in combination with other values to make a single abstract statement, that are available. It is possible

This is a specific case of the general case of loss of visibility.

A.8. Use in a configuration

Consider an application of some resources. It is generally the case that the same type of resource will be used in various places in the application. Consider use of an ethernet termination in a particular configuration of functionality such as an G.8032 ring where the ethernet termination (interface) deals specifically with the termination of signaling. In this application the termination has a specific subrange of acceptable vids and has a particular relationship to other terminations in the solution. These restrictions need to be conveyed in the model of the application and ideally. Ideally thw ability to convey these restrictions would be an inherent aspect of the metamodel for the language such that the use of a termination, i.e., the occurrence could be declared with its specific subrange of allowable values and its specific restricted associations to other termination points.

This is an example of a point on a progressive narrowing of definition and an example of Occurrence.

A.9. Bandwidth time variation

There could be a request for an e-line where the bandwidth requirement varies over the day and where that variation is dependent upon the setting of other properties. Specification here would benefit from an inherent capability in the metamodel of the language of expression of the property definition to cater for time variation and interrelationship such that any temporal properties and interproperty interrelationships can be specified as a natural part of the language.

Taking this further, assume that for this example: - the bandwidth is specified by an integer in bits/second and that this is sufficient to deal with the full range of variability (clearly this is a simplification for the example). - the temporal characteristic is specified by a normalized temporal expression that allows for a variety of repeat and calendar-based strategies, but for this example is simply 1000000 bits/second for even clock hours and 0 for odd clock hours. - the bit rate depends upon the value of another property, "load-factor" which is an integer with a range of 1-10, by some defined algorithm, here simply the product of bandwidth and load-factor

Ideally the specification of the bandwidth within the definition "integer" would provide an opportunity to state (or reference) the temporal characteristic and the relationship to load-factor such that the statement of Integer could include complex detail as opposed to just one value when complexity is required, but would fold-away to a simple form when a simple single value is to be stated.

This is an example of complex fuzzy boundaries.

A.10. Intent

A request for an e-line where there is an acceptable range of bandwidths and/or a cost profile for bandwidth or some cost/bandwidth combination and where this bandwidth is perceived by the client.

A.11. Recursive narrowing/refinement during negotiation

An initial position for planning some infrastructure where the capacity of termination is known and the building is known, but not the specific device. Through negotiation the details are refine to the point where sufficient is stated for the client. This may still not be a fully refined termination point as the cabling to the chosen device has not yet been resolved. The choice of termination can be made by the provider as a result of the activity to satisfy the intent.

A.12. Capability slice

Where the request is for some range of capability and the solution is an approximation to that capability where the approximation is stated as a bounded space.

Appendix B. Appendix B - Sketch of an enhanced YANG form

In this appendix a sketch of a language, that is a development of YANG, that resolves some of the issues is set out. The language is not completely formed, and it is not the intention that this necessarily be the eventual expression, this is simply used for illustrative purposes.

B.1. Progression

Using this language, a progression of increasingly specific models can be set out (as set out in the main body of this document). The refinements at each stage would be in terms of reduction of semantic scope compared to the previous stage. At an early stage of refinement, the component-system pattern emerges.

The language provides definitions for terminology (in a name space) where each term is defined by the specific narrowing (note that the terms are simply a human convenience).

The progression is not a partition. It does not divide the space up into a taxonomy. The progression does lead to sets of capability where those sets may intersect etc.

A traditional model is replaced by what is emergent at a stage in the recursion. A label (in a label space) chosen for a semantic volume should not be reused for anything else. Once defined the label should never be deleted, although it can be obsoleted (i.e., not expected to be used).

If the label is encountered, its definition should be available such that it can be fully interpreted. The label may apply in a narrow application and hence the narrowing definition should be available.

B.2. Language

As noted, it appears that there is not a good language suited to solve this problem fully. This may only appear to be the case, i.e., there may be a language out there, as it has proved very difficult to describe the problem (there does not seem to be readily available terminology for the concepts in the problem space) and hence it has been difficult to discuss what properties the language needs to possess.

To cut through this, the best approach appeared to be to sketch a language and show how it could be applied to hopefully tease out an existing language that could then solve the problem (or so it can be a basis of a new language).

As noted earlier, considering the general and apparently broad extent of the problem, it seems strange that there is not an appropriate machine interpretable language of expression available. It is possible that existing languages can be used to deal with the problem in a somewhat cumbersome way and that this has not yet been observed. Cumbersomeness can be refined out over time. Preferably there will not need to be Yet Another Language!

B.3. Key concepts

Recursive narrowing appears to give rise to "occurrences", where each of a set of occurrences is in part the same as and in part different from each other. Curiously:

* there also appears to be a parallel with the specific approach to specialization taken in ONF (and in YANG models using augment) where a "class" is actually a narrowed "occurrence" of a more general metaclass (see earlier discussion). The distinction in the general thinking is that there are no specific meta-levels. Narrowing can take place through a recursive continuum until all properties have been constrained to absolute precision (which is actually never possible as there is always some uncertainty, rounding etc.).

* When all properties have been constrained the result is the statement of a unique instance at a moment in time (where each occurrence has a lifecycle which intertwines with the lifecycles of other occurrences). But this instance is itself an opaque representation of the effect of underlying detail.

* this approach seems to point to some usages of the term "instance" being flawed and that actually, the supposed instance is an "occurrence".

Definitions may be of some type and may cover a subrange of that type. Even in an occurrence that is traditionally called an instance, the property values may be ranges. The key difference for the properties of an instance is that they are unlikely to be further decomposed.

B.4. Observation

At all levels of the recursion there is a mix of schema definition and absolute value (instance values). So, some of the information in a spec looks like instance data and some like schema. This should not be surprising when observing through the lens of recursive narrowing and of occurrence.

Curiously a YANG model definition has instance like data and schema data in it. For example, there is instance like data at the beginning of the definition with things like module, namespace, prefix etc. YANG does not appear uniform in its representation of instance like data and schema data.

The example language developed here attempts to achieve greater uniformity.

B.5. Progressing to the language

Taking JSON as a language of expression of instances and setting out a YANG definition in a JSONized form appeared to allow a uniform blend of instance and schema.

It has been recognized that YIN is a more well-structured form of YANG that points further towards this, and a JSONized form of YIN looked like the hand crafted JSONized YANG that has been constructed here (exploring the expression of recursive narrowing).

JSON has also been simplified here in that every property value is considered as a string and hence in quotes (even numeric quantities). It is not intended that any eventual language is restricted in this way, the approach just simplifies the representation and resultant discussion.

Note that regardless of the form of language chosen, there will be a need to enhance tooling etc. It is intended that the approach should evolve in small backward compatible increments and hence it may be possible to identify value-justified increments in tooling.

B.6. JSONized YANG

The following two snippets show the instance-like header and the schema data in a JSONized form.

B.6.1. JSONised Header

The opening of a YANG module (called a header in this document) is normally of a form illustrated in the example below:

```
module equipment-spec-xyz {

  yang version "1.1";

  namespace "urn:onf:otim:yang:spec-occurrence:equp-spec-xyz{uuid}";

  prefix equip;

etc.
```

In the JSONized form all of the fields are assumed to be "instance" values (where it is assumed that a higher level of specification has specified these). The JSONized form of the example (extended with some other suggested fields) is:

```
"module" : {
```

```
  "name": "equipment-spec-xyz{uuid}"
  "yang-version" : "x.y",
  "namespace" : "urn:onf:otim:yang:spec-occurrence:equp-spec-xyz{uuid}",
  "prefix" : "equip",
  "import" : [
        {
              "name" : "module",
              "prefix" : "mod"
        }
        {
              ....

  "occurrence-encoding" : "JSON",  //Field to explain encoding
  "rule-encoding" : "OCL", //Field to explain encoding
  "utilized-schema": [ //Ref schema at next higher "occurrence" level
        {
              "namespace" : "urn:company:yang:holder-schema-xyz{uuid}",
              "prefix" : "holder"
        }
        {
              "namespace" : "urn:company:yang:tapi-spec",
              "prefix" : "tapi-spec"
        }
        {
              "namespace" : "urn:onf:otcc:yang:tapi-equipment",
```

```
            "prefix" : "tapi-equipment"
            ...
      }
      {
            "namespace" : "urn:onf:otcc:yang:tapi-occurrence",
      ...
      }
      {
            "namespace" : "urn:company:yang:equp-schema-abc{uuid}",
            "prefix" : "equipment"
      }
      ...
 "augment":  [
      {
            "path" : "..."
      }
      ...
```

B.6.2. JSONized body

The core of a YANG module (called a body in this document) is normally of a form illustrated in the example of a fragment below:

```
grouping connection {

    list connection-end-point {

        uses connection-end-point-ref;

        key 'topology-uuid node-uuid nep-uuid cep-uuid';

        config false;

        min-elements 2;

        description "none";

    }

    list lower-connection {

        uses connection-ref;

        key 'connection-uuid';

....
```

In the JSONized form all of the fields are assumed to be "instance" values (where it is assumed that a higher level of specification has specified these). The JSONized form of the example (extended with some other suggested fields) is:

```
       "grouping":{
           "name" : "connection",
           "list": {
               "name" : "connection-end-point",
               "uses" : "connection-end-point-ref",
               "key" : "topology-uuid node-uuid nep-uuid cep-uuid",
               "config" : "false",
               "min-elements" : "2",
               "description" : "none"
           },
           "list": {
               "name" : "lower-connection",
               "uses" : "connection-ref",
               "key" : "connection-uuid",
               "config" : "false",
               "description" : "none"
           },
```

Notice the uniformity/consistency between the representations of the header and the body.

B.7. Schema for the schema

With this a YANG model can define a YANG model (within reason... and probably similar to other self-defining languages - improvements could probably be made to make this more possible).

Considering an extract from tapi-equipment formulated in JSONized YANG:

```
    "grouping":{
          "name":"common-equipment-properties",
          "description":"Properties common to all aspects of equipment",
          "leaf": {
                "name" : "asset-instance-identifier",
                "type" : "string",
                "description" : "none"
          },
          "leaf": {
                "name" : "is-powered",
                "type" : "string",
                "description" : "none"
          },
                "leaf": {
                "name" : "equipment-type-name",
                "type" : "string",
                "description" : "none"
          },
          "leaf": {
                "name" : "manufacture-date",
                "type" : "string",
                "description" : "none"
          },
          "leaf": {
```

```
            "name" : "serial-number",

            "type" : "string",

            "description" : "none"

        },

        "leaf": {

            "name" : "temperature",

            "type" : "decimal64",

            "description" : "none"

        },
```

It is expected that the above model would have been derived from a broader model and that it would reference that model.

In the following development of the model a reference would be provided back to the model above.

This could be used as a general definition, then constrained for a particular application as follows:

```
 "grouping":{

       "name":"common-equipment-properties",

       "description":"Properties common to all aspects of equipment",

       "leaf": {

            "name" : "asset-instance-identifier",

            "type" : "string",

            "pattern" : "^[0-9a-zA-Z]+"$, // A narrowing constraint.

            "description" : "none"

       },

       "leaf": {

            "name" : "is-powered",
```

```
                "type" : "string",
                "supported-constraint" : "NOT_SUPPORTED",//A narrowing.
                "description" : "none"
          },
          "leaf": {
                "name" : "equipment-type-name",
                "type" : "string",
                "description" : "none"
          },
          "leaf": {
                "name" : "manufacturer-date",
                "type" : "string",
                "description" : "none"
          },
          "leaf": {
                "name" : "serial-number",
                "type" : "string",
                "description" : "none"
          },
          "leaf": {
                "name" : "temperature",
                "type" : {
                      "name":"decimal64",
                      "fraction-digits":"1", // A narrowing constraint.
```

```
                "range" : "0.0..100.0",

                "precision":"+0.2,-0.2"

            },

            "units" : "Celcius", // A narrowing constraint.

            "description" : "The temperature of the boiler."

        },
```

Which could be summarized with a reference to the earlier schema and then as follows, where the absent fields are unchanged from the earlier schema and the fields mentioned simply show the change/ addition:

```
     "grouping":{
           "name":"common-equipment-properties",
           "utilized-schema" : "tapi-equipment", //The reference
           "leaf": {
                 "name" : "asset-instance-identifier",
                 "pattern" : "[0-9a-zA-Z]"
           },
           "leaf": {
                 "name" : "is-powered",
                 "supported-constraint" : "NOT_SUPPORTED"
           },
           "leaf": {
                 "name" : "temperature",
                 "fraction-digits": "1",
                 "range" : "0.0... 100.0",
                 "units" : "Celcius"
           },
```

And eventually instance values can be mixed with schema...

```
    "grouping":{
          "name":"common-equipment-properties",
          "description":"Properties common to all aspects of equipment",
          "utilized-schema" : "tapi-equipment",
          "asset-instance-identifier" : "JohnsAsset"
          "leaf": {
```

```
            "name" : "equipment-type-name",
            "type" : "string",
            "description" : "none"
        },
        "leaf": {
            "name" : "manufacturer-date",
            "type" : "string",
            "description" : "none"
        },
        "leaf": {
            "name" : "serial-number",
            "type" : "string",
            "description" : "none"
        },
        "leaf": {
            "name" : "temperature",
            "type" : "decimal64",
            "range" : "0.0... 100.0",
            "units" : "Celcius",
            "description" : "none"
        },
```

Notice that as with normal JSON the name of the property "asset-instance-identifier" is followed by its value (or json-object). The "utilized-schema" has defined "asset-instance-identifier" and the utilization method hence allows the property to either be defined further or narrowed to a fixed value. There are clearly "key words" such as "leaf" that will have been defined in an "earlier" schema, so something like:

```
  "field": {

        "name":"leaf",

        "type": "strings"

  ...
```

And further there would need to be a definition of "name" and "type" etc. and their usage in a structure.

B.8. An example of spec occurrence and rules

This example detail will be added in a later version.

B.8.1. Rough notes

Considering an occurrence of holder in a spec for an equipment, there is a need to identify all compatible equipment types (i.e., that that holder can accommodate). Each type would be associated with a general spec of capability and that spec would relate to the specific application (in the holder) either directly (as the holder is fully capable or the spec is specific to the holder) or via some variables in the spec (that allow modulation of the spec statements).

There are also combinatorial rules. For example, a slot may not be equipped if blocked by a wide card. This can be represented by....Wide card

It is possible that there are multi-slot compatibility rules.

The functional capability may be simply the capability of the equipment type, but there is often functionality that emerges from a combination of hardware.

In this case an equipment may support some fully formed capabilities and some capability fragments that need to be brought together with other fragments to support a meaningful function.

In some cases, functionality from one piece of hardware might compete with functionality from another or functionality might be completely nullified by the presence of another specific piece of hardware.

The equipment-type-identifier is the reference to the equipment spec. This brings details of size and capability.

It is assumed that the holder specification would provide width, position etc. and that there could be a general understanding of size, or there could be a more abstract representation to enable overlap to be accounted for.

B.9. The current schema

This detail will be added in a later version of this document.

B.10. YANG tree

This detail will be added in a later version of this document.

B.11. Instance example

This example detail will be added in a later version of this document.

B.12. The extended schema

This detail will be added in a later version of this document.

B.13. Versioning considerations

This detail will be added in a later version of this document.

Appendix C. Appendix C - My ref / your ref

This appendix will cover "my ref/your ref" naming in relationship to intention/expectation. The detail will be added in a later version of this document.

Appendix D. Appendix D - Occurrence

An "occurrence" at one level of specification is a narrow ("specialized") use of an "occurrence" at the previous higher level of specification. There will be many "occurrences" at a lower level derived from an "occurrence" at a higher level. The "occurrences" at the lower level will be distinct from each other.

Considering the problem space, "Thing" has the broadest spread of semantics covering everything. Function could be considered as a narrowed thing covering only functional aspects and not physical aspects. Termination covers only those functions related to terminating a signal (and not those related to conveying it unchanged.

Considering the formulation of a traditional model, a specific object-class (e.g., Termination Point) is a specific name for an "occurrence" at one level of narrowing ("specialization"). The name object-class is a specific "name" for an "occurrence" at the previous higher level. That previous higher level is called the metamodel in this naming scheme. This is more a narrowing of how to express as opposed to what to express.

Considering the traditional model, "Thing" is effectively an object-class. Considering "Component-System", "Thing" has ports/facets, as do all of its derivatives (unless, of course, they have been narrowed away). The "Component-System" formulation is essentially a narrowing of the expression opportunity in the broadest of problem space considerations at a higher level than "Thing". It effectively provides a quantized representation of a continuous space allowing representation of a refinement of conceptual parts.

In the generalized "occurrence" approach, no specific names are given to the levels and the levels are intentionally not normalized. The approach allows any number of levels, and the number of levels in one "branch" does not need to match the number of levels in another. The approach allows for whatever degrees of gradual refinement are necessary.

So, a traditional "instance" is also an "occurrence" where it is an "occurrence" of specific object- class, i.e., of an "occurrence" at the previous higher level. The "instances" is a leaf in the metamodel structure.

In the generalized "occurrence" approach there is no specific end leaf. Even when the model level has only fully resolved specific values, it is possible to merge in a model with non-specific value "occurrences" and continue to refine.

A specific occurrence:

* Is narrowing of a previously defined occurrence (where there may be many separate distinct narrowings defined at that "level", many occurrences

* Is a mixture of absolute values and definitions

* May be a merge of narrowed previously defined occurrences (where the semantic phase is shifting)

* Is the basis for further narrowing

It also appears that an absolute value of a property at one level may be considered as an unstated approximation of a non-fully resolved property at the next level down. For example, a statement of 2 at one level, refines to 2+/- 15ppm over a short period at the next as the visibility "improves". This seems to say that all levels have a natural uncertainty that may not be known and hence need not be stated.

Appendix E. Appendix E - Narrowing, splitting and merging

E.1. Narrowing

On the most abstract level is "thing" - without any stated parameter, hence without any constraints. "thing" is anything without restriction and can take any shape etc. All possible properties are allowed without restriction (they do have to be declared, but there is no boundary as to what is allowed to be declared). The declared properties are of "thing". It is a semantic set including every possible behavior etc. and all parameters possible.

At the next level of narrowing, some behaviors and hence possible parameters are eliminated and some constrained versions of allowed parameters are exposed. At this point, a convenient name for the specific narrowing can be provided and later references. However, this can also be considered still as "thing".

In the most complete realization, the semantic boundary would be fully defined such that properties on the boundary were appropriately constrained and properties beyond the boundary eliminated. For example, a termination-point cannot have a temperature. So, an expression eliminating this possibility would be necessary and so on for all other properties that cannot be supported. In a more practical implementation, most properties that are beyond the boundary can be assumed to be known to be beyond the boundary and only those with complex constraints need to be stated.

When narrowing "thing", what it can be is reduced and hence so are the parameters that can be exposed. For example, if it is not physical, I cannot expose the parameters for weight.

As the "thing" is narrowed the properties that are allowed to be exposed reduces, but there is a tendency to have more properties exposed as more and more properties become constrained. For example,

color may be irrelevant and hence not exposed or constrained for some of the broader "occurrences", but as the narrowing process approaches the useful definitions, the color becomes constrained and useful, and hence exposed.

Through the narrowing process the set of opportunities becomes smaller and "lighter weight" (possibilities), but more is exposed (semantic mass reduces, semantic visibility increases).

Consider an equipment. It is a physical thing. It may be narrowed to the point where it is constrained such that it can only be plugged into a slot. It is still an equipment, albeit a constrained forms of the general equipment properties for an application. The equipment has a property "requires-slot" set to true. During the first formulation, color may be of no relevance and hence is not exposed. Just because it is not important does not mean that the equipment does not have a color, it means that it is not relevant. It can take any color and I don't care.

Later, the color becomes important and hence it is exposed and later still it becomes necessary to choose to constrain the allowable colors for an application. It is still an equipment, but it has a spec that constrains what is allowed. The spec is for a narrow form of equipment. It can still be considered as an equipment even though it is a narrow case. It can also be considered as a "thing" with exactly the stated property with some further directly stated constraints.

Narrowing could be considered as pruning (removal of unwanted parts). For example, take a property (leaf/structure in general) from the higher occurrence and potentially:

* Reduce its legal range (perhaps to a single value) of the type. Note that changing the type is allowed if the new type covers the same semantics as the old type. So, Integer to real seems OK and Enum to its corresponding semantic space dimensions seems OK (e.g., color Enum to RGB ranges)

* Specify the units where relevant (or change the units)

* Relate its value to other property values such that its value is constrained

* Remove the property completely

* Change its name (label) to one that represents the narrow version of the broader property, for example, component --> termination-point

This is how modelling is often carried out although it is never formally described as a method. Consider the termination point, the model is for any connection at any layer etc, and there are "profiles" of parameters for particular technologies which augment the termination point. The profiles may may add (actually expose) further parameters for the same technology or for new technologies, but termination point can never have weight as a parameter and certainly cannot be sat on!!

YANG augment is the same process in general, so YANG is positioned appropriately to be the language for this approach.

Interestingly, an AI solution may eventually prefer to use semantics closer to thing than to EthernetTerminationPoint as it can deal with the shades of specification of general things.

E.2. Splitting

Splitting semantics is relatively straight forward. Two distinct occurrences each narrowed from the same higher-level occurrence where the two new occurrences have distinct characteristics.

E.3. Merging

As everything is an "occurrence" of thing, everything has a common highest level of thing. When merging, it may be necessary to go some way back towards that common highest level.

Where the two occurrences are disjoint in distinct characteristics and identical in common characteristics, merging is simply a union. The result may adopt the label of the shared higher level or may have a new label depending upon the labeling (naming) strategy.

Where common characteristics are not identical:

* one may be a simple superset of the other in which case the superset is adopted.

* There may be contradictions in the two specifications in which case there needs to be a simple precedence, e.g., Not overrides Must,

Where merging two (or more) properties from higher models into one property

* There must be a new name for this rephasing of the semantic if neither of the source properties were derived from an origin with the same breadth of space as the new property

* One of the names can be taken if it earlier in the narrowing corresponded to the superset of the new merged result

For example, TerminationPoint narrowed to have no OAM capabilities and then OAM picked up and merged in (again) at some later stage so that this is still TerminationPoint (but it is NOT OAM).

For example, a narrow form of TerminationPoint (processes of traffic at a point) and a narrow form Connection (conveys traffic of space with no transformation) merged into a (strange) long termination that does some distributed processing of traffic. This is neither a TerminationPoint nor a Connection. It could revert to Component with full spec, or it could become a ProcessingSpan (or similar).

Appendix F. Appendix F - A traffic example

This example detail will be added in a later version of this document.

Acknowledgments

Contributors

Martin Skorupski
highstreet technologies
Email: martin.skorupski@highstreet-technologies.com

Wade Cherrington
Ciena
Email: wcherrin@ciena.com

Author's Address

Nigel Davis
Ciena
Email: ndavis@ciena.com