Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal reference number of this OGC® document: YY-nnnrx

Category: OGC® Discussion Paper

Editor: Sam Meek

Conceptual Modeling Discussion Paper

Document type: OGC® Discussion Paper

Document subtype:

Document stage: Draft

Document language: English

License Agreement

Table of Contents

i. Abstract

Historically, conceptual modeling was utilized sporadically within the OGC. Models were used in OGC standards both informatively and normatively to describe the structure of the information within a standard for a particular domain. As independent standards-development organizations, OGC and alliance partners such as ISO TC211 did not always develop common models. There are several examples of conceptual models in OGC's Abstract Specifications, many of which have become ISO TC211 standards since their publication. Outside of Abstract Specifications there are fewer examples of conceptual models in Implementation Standards. Logical Models and Physical Models tend to be specified more in Implementation Standards.

The need for conceptual models in Implementation Standards has become apparent since the OGC is moving towards resource based architecture through the development of the OGC API suite of standards. In the previous ways of working, standards and encodings mapped 1:1, as many OGC standards were XML based and a standard described a particular set of XML documents to support a domain. The move to OGC API has led towards a separation of an information model represented in a standard from encodings, which is the way that the information models are expressed in a given technology. In other words, the move to OGC API has led to a clearer separation of the logical model from the physical model.

The utilization of conceptual modeling practices may be employed to manage, track or govern the use of concepts and terms within different standards. The OGC should adopt conceptual modeling where suitable with a new group to support the working groups with the modeling effort that may otherwise have not been completed because a lack of expertise or value recognition. Taking the concept one step further; Model Driven Architecture (MDA) is a transformation process to create a platform specific model, or implementation from a logical, platform independent model. This process could be implemented to enable quick production of standards into different target technologies or for the creation of new standards entirely. This paper does not suggest making MDA and associated mandatory for future standards generation.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, UML, Model Driven Architecture, Conceptual Modeling, Abstract Modeling, Platform Independent Model, Platform Specific Model

iii. Preface

This document proposes a new group within the OGC to support the existing groups with conceptual modeling and automatic model transformation technologies. If received successfully, the next step is to form a new group with a group charter.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s) Helyx SIS

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Sam Meek Helyx SIS

1. Scope

This Discussion Paper provides an insight into the on-going work within OGC regarding Conceptual Modeling and secondarily, Model Driven Architecture (MDA). The paper makes a case for a new group to be formed within OGC to assist the Architecture Domain Working Group (DWG) and all Standards Working Groups to define, record and ratify their conceptual models as compendiums to their standards.

2. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC: OGC 20-012 UML-to-GML Application Schema Pilot (UGAS 2020) Engineering Report (https://docs.ogc.org/per/20-012.html)

OGC: OGC 20-033 OGC Testbed-16 OpenAPI Engineering report (https://docs.ogc.org/per/20-033.html)

ISO: ISO 19101-1:2014(en), Geographic information — Reference model — Part 1: Fundamentals

OGC: OGC 19-014r3, OGC Abstract Specification Topic 22 - Core Tiling Conceptual and Logical Models for 2D Euclidean Space

3. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Best Practice.

For the purposes of this document, the following additional terms and definitions apply.

3.1. Conceptual Model

A description of common concepts and their relationships, particularly in order to facilitate exchange of information between parties within a specific domain. A conceptual model is explicitly chosen to be independent of design or implementation concerns. (SOURCE: OGC 19-014r3, after CEN ENV 1613:1995) NOTE: ISO 19101-1:2014 defines a conceptual model as a “model that defines concepts of a universe of discourse.” This definition is considered to be consistent with that presented by OGC 19-014r3, after CEN ENV 1613:1995.

3.2. Logical Model

An abstracted representation of domain knowledge that could feasibly be used in Model Driven Architecture to generate physical artifacts.

3.3. Model Driven Architecture (MDA)

Model-driven architecture (MDA) is a software design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models. Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems.

3.4. Metamodel

A metamodel or surrogate model is a model of a model

3.5. Physical Model

An implementable representation of domain knowledge for a specific target technology.

3.6. Platform Independent Model

A representation of domain knowledge that is not tied to a specific technology and is usually represented in a platform independent language. (see logical model).

3.7. Platform Specific Model

A representation of domain knowledge in a target technology that can be directly implemented (see Physical Model)

3.8. Universe of discourse

A view of the real or hypothetical world that includes everything of interest Source: ISO 19101-1:2014

4. Background

OGC has been creating open standards in the geospatial domain for over 20 years. These standards arise through a combination of formalizing domain knowledge, implementers reflecting practices, and decision by committee. These methods of generating standards achieve success proportional to interest and in-kind effort, i.e. OGC Standards are developed by the membership. Many of the initial set of standards are in use today, for example, the Web Map Service (WMS) Standard is still widely used across the web and within organizations to distribute mapping. WMS is part of the OGC Web Service (OWS) suite of standards. A new generation of Web API standards is emerging within the OGC. This new generation of standards, which makes use of the OpenAPI specification, is called the OGC API suite of standards. OGC API standards will become an alternative to the well-established OWS suite of standards.

The creation of new standards is largely organic and takes place within a Standards Working Group (SWG). The initiation for a new standard or a new version lies with the OGC Technical Committee membership. Standards creation is often an isolated affair, although there is informal cross communication between groups. The coordination of standards development relies on the activity of the group members. An overall abstract specification provides some grounding and guidelines for standards development, but does not act as a governance structure for any standard being developed, with the group members and the wider technical committee members responsible for assurance.

Often, standards are designed to reflect the requirements of a domain, at least in theory. In practice, standards development is led by implementers, probably because it is the implementers that have a route to produce the standard. Standards have traditionally been focused on a particular encoding, we have the OpenAPI suite or the GetCapabilities suite and the appropriate use of a chosen technology makes up a large part of standards definition work. DWGs and SWGs consult with industry, governmental and academic partners on development of standards, however, the domain knowledge tends to arise from within the membership that contains members form these different groups. Regardless of the intent of developers to reflect domain knowledge, it is questionable how frequently design decisions are revisited. A typical example of this is OGC API – Processes, which almost maps 1:1 what was in the Web Processing Service (WPS) version 2. This is not necessarily a negative quality, but it is unclear what work was done to understand if it still reflected domain requirements. WPS 2.0 did not receive the uptake that many of the other OGC standards have, it is unclear why.

Whereas OWS offer a GetCapabilities operation for accessing the description of capabilities offered by a web service, OGC APIs offer an OpenAPI definition document for a similar purpose. It should be recognized that in some respects, many elements of new standards are not being created, but instead new encodings are being adopted. OpenAPI is a specification that utilizes JSON or YAML encodings to create an interface definition. However, technologies evolve, therefore the dominance of OpenAPI in the API market cannot be guaranteed to persist indefinitely. This has been seen previously in OGC standards with approaches such as SOAP, which were quickly replaced with JSON or were overlooked in favor of older encodings such as XML, possibly for compatibility reasons. An overall point is that the OGC is largely a technology focused organization that develops standards to fit a domain, with emphasis on the technology aspect. This is in contrast to organizations that come represent a domain, for example, meteorology, defense, maritime, health, aviation, agriculture, to name just a few. The standards seek to optimize interoperability between technologies in a particular domain.

There is a balance to be struck between; creating conceptual artifacts that best reflect the requirements of the domain and creating physical standards that can be implemented as interfaces. However, physical standards can only be successfully implemented if the conceptual, domain knowledge has been initially captured - otherwise the standard is not necessarily a reflection of domain requirements and therefore unlikely to be adopted widely by the community.

4.1. Model Abstraction

There are essentially three levels or types of models that can be employed, these are the following:

- Conceptual - a description of common concepts and their relationships, particularly in order to facilitate exchange of information between parties within a specific domain. A conceptual model is explicitly chosen to be independent of design or implementation concerns (OGC 19-014r3, after CEN ENV 1613:1995).
- Logical - an abstract representation of an interface or data model that can be *executed* to produce physical artifacts. Another oft-cited aspect of this is that logical models be *machine readable*, although this is not a considered mandatory in this paper. Logical models, although abstracted from physical models usually have some overarching approach, for example, relational databases.
- Physical - the implementable artifacts, these may be produced from the logical model or produced independently.

The OGC utilizes all three models in the general sense, however, only the physical model is mandated and is usually expressed in the form of an XML schema, a JSON schema or an interface definition. In the majority of cases, the physical models are generated at the physical level, either manually or by crafting the definition using an editor to do so. There are methodologies (discussed later) that allow automatic movement from the logical to the physical level of modeling provided the artifacts are crafted according to a set of rules that fit the transformation process. Conceptual models are slightly different in this hierarchy, as there isn't a defined transformation process.

The OGC has upon occasion conflated or interchanged the terms *conceptual* and *logical* to describe many of the models. It should be recognized that these are different levels of modeling for different purposes. Confusingly, an alternative model ordering that simplifies the definition of model types is defined in Model Driven Architecture (MDA) (https://www.omg.org/mda/specs.htm) , these are:

- Platform Independent Model (PIM).
- Platform Specific Model (PSM).

The two models are utilized in MDA to describe an independent model to create multiple specific models, that is, the relationship between PIM and PSM *should* be one to many. Whether the definition of a PIM requires it to be transformed into a *PSM* is a matter for discussion. An additional consideration is what a PIM and PSM looks like in terms of technology. PIMs by definition need to abstract from any particular technology in order to generate specific representations ready for implementation. In many cases, the PIM is a UML representation (but not always). PSMs maybe a specialized version of the PIM in UML, but they are more often representations in a target technology (for example, an XML schema).

PIMs in practice are not necessarily completely independent of a target technology and maybe abstracted to a technology approach. An example of this is relational databases modeled using UML class diagrams. The concepts within relation databases are well accepted - 1st, 2nd and 3rd normal form, the use of primary and foreign keys, code lists and enumerations are all well accepted within the relational database modeling community, which in turn allows for technology specific artifacts (Postgres DDL, Oracle, SQL Server etc) to be derived from the same PIM. At this point, it is a case of ordering the model detail into the correct representation for the target technology. However, generating database artifacts (PSM) with a different approach such as NoSQL or an Elastic instance is likely to be more challenging.

NOTE

The use of the term conceptual model in this paper is not specific to the conceptual, logical, physical hierarchy and is used in the context of capturing domain knowledge in an abstract, but formalized way.

4.2. OGC Discussions on Conceptual Modeling

This section contains a summary of recent previous discussions within the OGC regarding conceptual modeling and its applicability to the OGC as a whole. The Boulder 2015 TC (https://docs.ogc.org/as/04-084r4/04-084r4.html#_conceptual_modeling) captured the following benefits regarding conceptual modeling:

- Promotes consensus on the concepts, independent of implementation-specific standards;
- Communicates the concepts and their intended meanings – easier than reading an encoding; and
- Helps ensure compatibility between multiple implementations.

The last point in that list is key to the push for conceptual modeling and an associated group to understand and govern the creation of the models. OWS Common, and more recently, OGC API - Common are essentially specifications that all of the other standards in each suite utilize. Compatibility between implementations of individual standards should be granted by underlying concepts such as *Common* and having a set of tools and agreed upon modeling language to help that cause is key to success. Additionally, implementation compatibility can be facilitated if processes such as (MDA) can be employed appropriately.

A presentation at the aforementioned Boulder meeting proposed that *all* OGC Standards *shall* have a conceptual model and cites similar benefits with the addition of conceptual models being a *precursor to the semantic web* that has an explicit goal to make data on the web *machine readable*. In order to make that happen, there has to be agreement on the meaning of terms and abstracted concepts. Tools like conceptual modeling can help with this as standards can inherit concepts already defined. A typical example is the concept of a *feature*; a feature has certain properties including a *geometry*, which is also a recognized concept and every standard that utilizes a *feature* has the same concept, but not necessarily the same encoding (GML Vs GeoJSON, for example).

Conceptual modeling has been discussed in the OGC Architecture Board (OAB) several times and has shown that it has been considered by several different groups. For example, issue 1438 (https://portal.ogc.org/?m=projects&a=view&project_id=228&tab=5&act=details&issue_id=1438) suggests that CityGML 3.0 group has sought to create a conceptual model for their standard that can be used as a template for all other standards within the OGC. Whether this lofty ambition comes to fruition is currently unknown and the group have created and refined a complex and sophisticated set of conceptual models. However, whether they can be considered *conceptual* or *logical* is debatable, as they utilize many ISO 19000 standards, which are inherently XML based.

A known issue with conceptual models is their accessibility within the standards in which they do appear. Another OAB issue (https://portal.ogc.org/?m=projects&a=view&project_id=228&tab=5&act=details&issue_id=1485) highlighted the lack of accessibility of conceptual models when they are developed. An example given is again CityGML 3.0, where although the UML is present in the documentation, however, accessibility of the model is limited to the generated images. A solution to this is to make the full model available either in its raw form (an Enterprise Architect -EA file or an XML Metadata Interchange - XMI file), or via a generated set of HTML pages available in EA to enable implementers to explore the model attributes.

Another noted issue (https://portal.ogc.org/?m=projects&a=view&project_id=228&tab=5&act=details&issue_id=1047) is that some standards that contain UML models are incorrect with many errors.An additional consideration is how much UML models should be harmonized across the OGC and with models that are relied upon in ISO TC211, which many of the older standards relied upon.

Observations from this section are as follows:

- There is a recognized need for conceptual modeling within the OGC, although the scope and requirement is unclear.
- The use of the terms *conceptual* and *logical* to describe models are often conflated or used interchangeably.
- Some standards do have normative UML models, notably CityGML 3.0
- The models that are available are not necessarily accessible, and therefore not as useful as what they otherwise might be.
- Worryingly, some of the UML models that are available are incorrect in terms of their construction and do not necessarily harmonize with each other or supporting models from ISO TC211.
- The *value* added by conceptual modeling differs between efforts.

4.3. OGC conceptual modeling initiatives

There have been several formal attempts to utilize conceptual modeling within the OGC. Two recent documents include the following:

- UGAS Pilot (2020) (https://docs.ogc.org/per/20-012.html)
- Testbed-16 OpenAPI (https://docs.ogc.org/per/20-033.html)

The UGAS Pilot (2020) was had many objectives, the following are relevant to this discussion:

- Refine UML to JSON rules for the NSG wide UML schema.
- A preliminary investigation into MDA for OpenAPI interfaces (work eventually completed in Testbed-16)

The UML to JSON schema aspect took a direct approach when converting from UML to JSON, that is, they did not enforce a metamodel/model pattern and the approach in the same manner as Testbed-16, where the model classes are essentially specializations of the metamodel. The UGAS Pilot, as suggested by the acronym, utilizes the *Application Schema* model from ISO 19109 and depends on the requirements of ISO 19103 Conceptual Schema Language. Although there is no direct JSON alternative to Application Schema, the approach taken in the Pilot seeks to fulfill the requirements of Applications Schema using a first principles to derive translations based upon common JSON patterns of use, GeoJSON design and other pragmatic considerations.

Testbed-16 OpenAPI thread enabled the creation of an MDA pipeline for OpenAPI interfaces with an initial demonstration using OGC API - Features Part - 1:Core as well as some other simplistic, non-OGC conformant OpenAPI interfaces. The work involved creating a metamodel for the OpenAPI specification that could be specialized into implementations that were then run through the ShapeChange (https://shapechange.net/) transformation software. A plugin for ShapeChange was created to do the transformation from UML to JSON, this was made simple by using the metamodel construct to apply rules at the abstract level. The exercise resulted in a compliant OGC API - Features - Part 1:Core output, however the test case demonstrated was simplistic and relied on references to defined components, schemas and parameters.

Although the OpenAPI tests completed in Testbed-16 were a success, there were several recommendations that were made regarding the MDA process, the salient ones are thus:

- Complex modeling process should be simplified through packaging.
- The process of creating an interface should be reversible.
- The relationship between a standard and encoding should be recognized.

The final point provides the strongest argument for conceptual modeling as an end rather than just a means (such as in MDA). Domain knowledge changes infrequently while technology changes frequently. Therefore, many version changes in standards simply reflect the old domain knowledge in a new encoding, potentially with refinements and extensions. Conceptual modeling allows the domain knowledge to be held in a single place that can be extended and amended at the most abstract level. If conceptual modeling is extended to be used in MDA, then any changes reflected in the domain knowledge can be automatically added to abstract standards and interface implementations.

These projects and subsequent discussions have resulted in the same question being asked time and time again, to paraphrase; *what is the purpose/place/utility of conceptual modeling and transformation technologies and how should they be used within OGC if at all?*

5. Conceptual Modeling

As mentioned previously, up until recently, the only encoding that implementers in the OGC world had to concern themselves with was XML as the web services and data were presented in XML based profiles. However, this is no longer the case with datasets and standards required in a variety of encodings including:

- JSON
- YAML
- XML
- JSON-LD
- Google Protocol Buffers
- GeoPackage and other databases

As the encoding chosen no longer maps 1:1 with the standard, overarching concepts are required to abstract away from the encodings to a unified, standardized model.

A conceptual model should hold the domain knowledge that is expressed in the standard via some encoding. As mentioned previously, when all encodings for standards were XML based the standard and encoding held all of the domain knowledge and expressed it at the same time. However, in the world of mixed architectures, standards and encodings, the expression of the domain knowledge can be done in a variety of encodings.

Creating conceptual models to hold domain knowledge at the highest level allows formalization of concepts via relationships and eventually, semantic agreement. This means that when changes are made to the overarching concepts, its semantic relationship with other concepts can also be formalized. Additionally, conceptual modeling allows semantic harmonization across standards. Eventually, these concepts can be extracted from their individual concept level standards and brought together to harmonize the concepts. The advantage to this is that the addition of new domains and therefore new standards to the OGC is simplified through the formalized process of capturing domain knowledge and the semantic concepts already defined.

5.1. Model Abstraction

How far to abstract a model from an implementation to its core concepts is a balancing act between maintaining usability to complete independence from any particular implementation. From an OGC perspective, conceptual models are utilized in many of the standards both established and emerging but not all. The Sensor Observation Service (SOS) used full conceptual, logical and physical modeling to produce the standard. SOS is XML based, therefore many of the reported benefits of this approach such as producing multiple targets from one set of models are likely to be lost. Some standards have had success with full transformation into multiple target technologies; OWS Context has a model that can be transformed directly into XML or JSON, likewise, O&M (https://www.ogc.org/standards/om) and CDB (https://www.ogc.org/standards/cdb) also have similar approaches.

An open question within the OGC is to understand whether a conceptual model should be mandatory for a standard and if it is, what form should it take? The lowest level of burden requiring in a model at all would be to produce a purely conceptual model that reflects the organization of the domain with little regard to any target implementation. Indeed, these models have been specified in a number of standards, however, the utility of producing one for anything more than documentation purposes is questionable.

An alternative approach to the conceptual, logical, physical differentiation is to simplify and use the terms *platform independent model* (PIM) and *platform specific model* (PSM). This sidesteps the conflation of *logical* and *conceptual* modeling to just two sets of models;

1. abstract representations
2. implementable schemas

Testbed-16 and the forthcoming Testbed-17 seek to utilize models for directly generating API definitions. By the nature of the requirement, the models were created at the logical level with enough information to perform a transformation to a physical model. If conceptual models are to be recommended or required for OGC Standards in the normative sense, then their utility will also have to be defined, i.e. can they be normative and are they suitable for MDA?

5.1.1. Diagramming Convention

The OMG Standard *Semantics of Business Vocabulary and Business Rules* (SBVR) is a method of relating models types and levels to each other via the use of UML and semantic relationships. The diagram below describes what looks like a hybrid between the concept model (note that this is different from the *conceptual model* and described in a later section) and the data format (sometimes called *encoding* in this paper) through the PIM and the PSM. In essence, it is a way of getting to the data format from the concept model through some intermediate steps. The semantic descriptions of the relationships differ from standard UML in that they are providing information that is not easily described using standard UML. This type of model structure may eventually provide the OGC with a method of separating high level concepts and encodings or data formats in a structured manner.

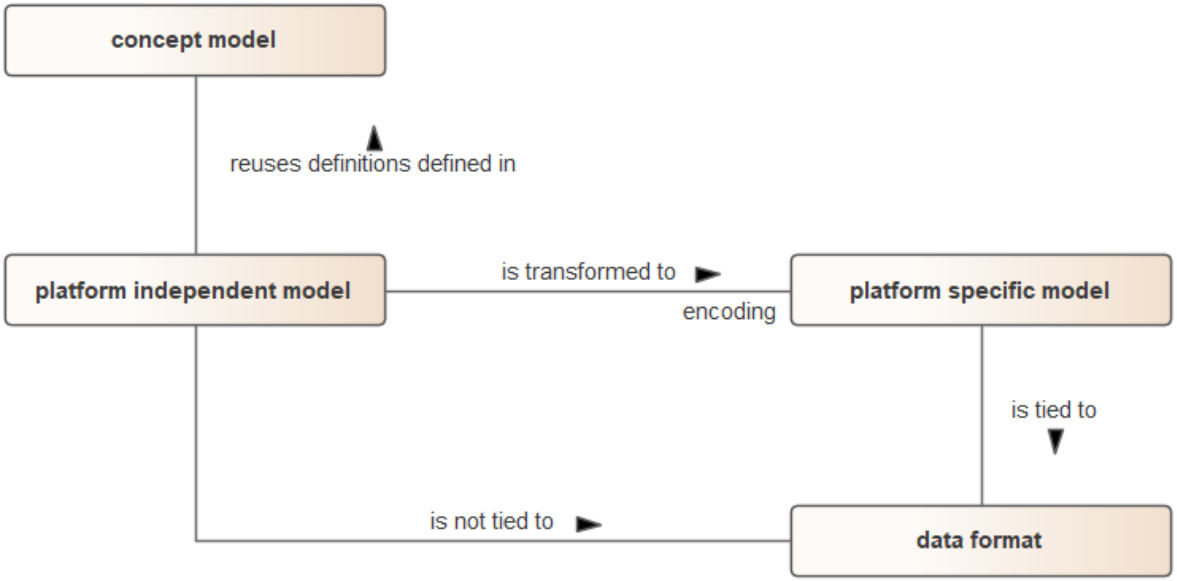

Figure 1. Semantics of Business Vocabulary and Business Rules model relationships

5.2. UML

Unified Modeling Language (UML) is often used as the standardized notation for modeling computational artifacts. It is not the only language used for this purpose, Archimate is quickly becoming the notation for architectures and Business Process Modeling Notation (BPMN) is used for modeling business processes.

UML class diagrams are often the notation of choice for describing objects and their relationships. Originally, class diagrams were designed for describing programs in object oriented (OO) languages. It was envisaged that variables and method stubs could be generated from class diagrams automatically. The obvious floor with this approach is that not all languages are OO, and the transformation to each language requires a new development delta.

A similar issue occurs today when UML Class Diagrams are used to describe data structures and interfaces as these are also assumed to be OO. A salient example of this is with OpenAPI, which has a specification aimed at JSON and YAML targets, which are not inherently OO, therefore compromises have to be made in the modeling to ensure readability, usability and conformity with the specification and UML.

From a modeling perspective, UML class diagrams are usually used to create logical models, as they can contain all of the information to generate an interface. However, this begs the question as to what should be used for a conceptual model. If conceptual modeling is done using a UML Class diagram and logical modeling is done using the same notation, then is a conceptual model a logical model that omits detail? Is a conceptual model organized differently to a logical model? The example SOS model does not differentiate between a conceptual and a logical model, likely because the only encoding it supported was XML, therefore the conceptual model in this case is a blend of the concepts and the logical to create the standard. A defining decision will be whether a model is organized according to the domain it is modeling or according the encoding or output format it is trying to represent.

5.3. OGC Abstract Specification

The OGC Abstract specification aims to create and document a conceptual model with sufficient detail to to aid the creation of implementation standards. Up until this point, this paper has looked at conceptual modeling within the OGC in terms of the membership comments and recent initiatives. However, it should be recognized that OGC has already considered the topic of conceptual modeling formally as part of the OGC Abstract specification. This set of documentation is lengthy, but obviously provides some insight into the vision of conceptual modeling within the OGC by providing an overview of OGC implementation standards in the form of a conceptual model.

The concept of conceptual modeling for standards is noted within the *Abstract specification* and makes the following points:

- Conceptual modeling addresses aspects of the standards definition, notably these include:
  - Identification of the concepts of interest.
  - Identification of the relationships between concepts.
  - The significance of a concept, its properties and attributes.
  - Identification of concepts from outside the standard scope.

The section also espouses the benefits of doing conceptual modeling within the OGC, however, it crucially mentions that the inclusion of a conceptual model within a standard is up to the *submission team or SWG*, and it is therefore not mandated. UML is also mentioned as a language that, *has been used successfully by many standards developers for conceptual modeling* although again, it is not mandated.

5.3.1. Use of modeling *within* the OGC Abstract Specifications

The Abstract specification makes use of UML as a modeling language within it to describe objects and relationships within class diagrams. It should be noted that the Abstract Specification contains its own diagrams, but much of the information on the Abstract Specification is held in other documents, notably from ISO TC 211 and other OGC documents and groups such as Simple Features SWG. An example of a model within the Abstract Specification is shown below:

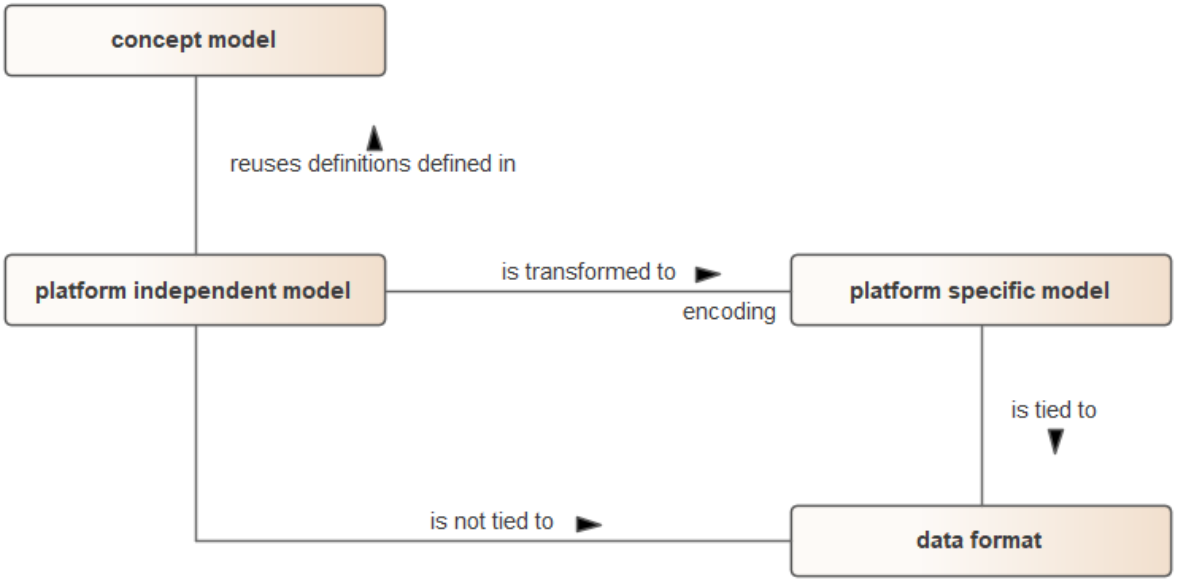

Figure 2. A Taxonomy-based Instantiation in Classical Object Oriented Programming Languages

As an example, the diagram uses UML to provide the structure and other relational aspects such as *qualifiers* to provide context between objects.

The document Core Tiling Conceptual model (https://docs.ogc.org/as/19-014r3/19-014r3.html) contains a discussion on conceptual modeling and logical modeling that is relevant to this discussion. The paper makes the case for characteristics of conceptual models and *logical* models; it describes the purpose of a conceptual model to *support the expression of natural language statements* and therefore *concept centric*, however, a logical model is *thing-entity-or-class-centric*. This description is in contrast to the *Concept Model*, which is described in a future section, which is unlikely to use UML to create a suitable model, as UML does not have the semantic capability to describe the relationships between concepts.

Overall, this aspect of the Abstract model is in contrast to some of the other thinking around the OGC. In some circles, conceptual models are seen as the *top level* of a conceptual, logical, physical hierarchy where each level abstracts away from implementation. The PIM and PSM hierarchy doesn't consider the *conceptual aspects* at all and is more concerned with implementation than describing concepts. There is also a *concept model* that seems to perform the same function as the *conceptual model* within the OGC Abstract specification.

5.4. CityGML 3.0 & other standards with conceptual models

This section is not intended to be an exhaustive list of standards with conceptual models, instead it is presents examples of OGC using conceptual models. A recent example is CityGML 3.0, which is an example of a mature practice of the OGC using conceptual models to define, express or generate standards. CityGML 3.0 case of conceptual modeling used to translate into multiple formats (https://github.com/opengeospatial/CityGML-3.0Encodings). A key aspect of the standard is that the model is in fact, normative - where there is a conflict between the documentation and the model, the model takes precedent.

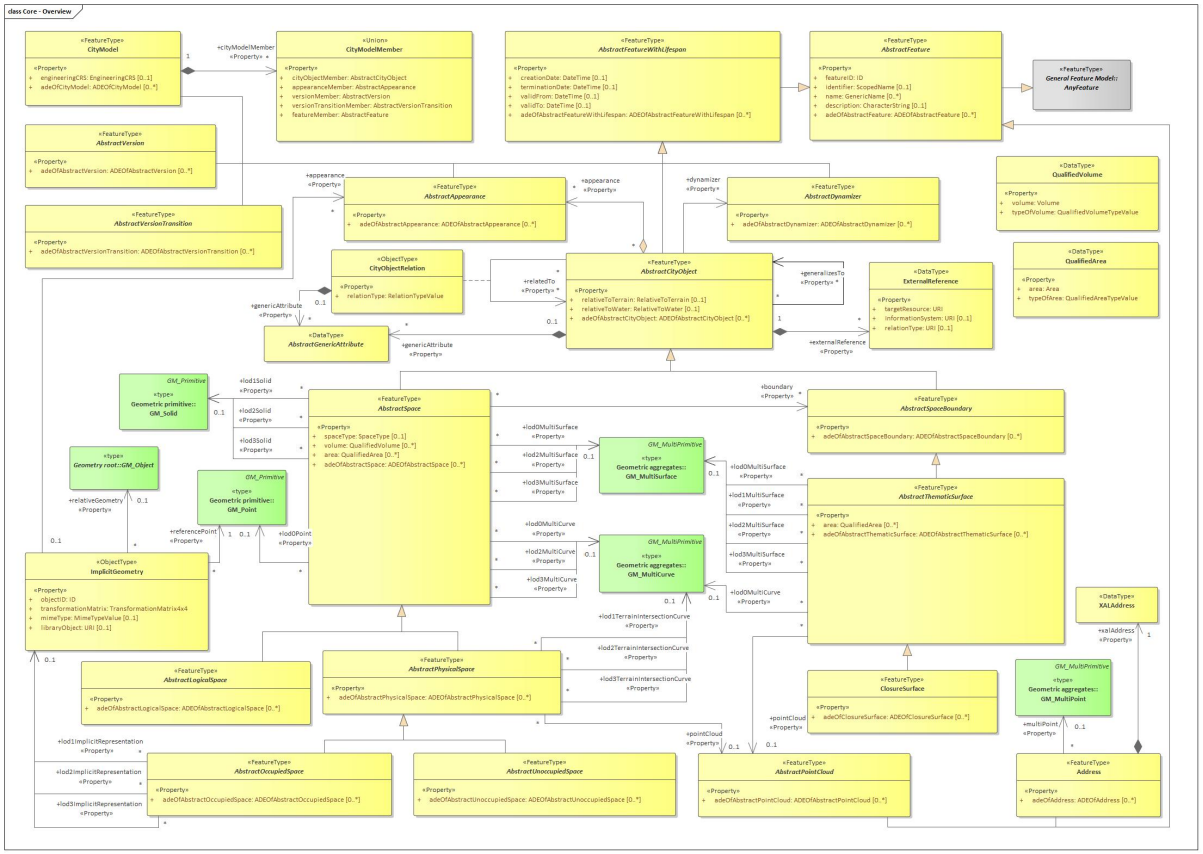

Figure 3. CityGML 3.0 Core conceptual model

In addition to the conceptual model being the normative part of the standard, a transformation process is in place to generate the standard artifacts such as the application schemas directly from the model with no manual intervention ideally required. This is an example of producing automatically generated artifacts from a conceptual model because, as mentioned, the model can be used to create other encodings beyond XML (from the GML). This calls back to the initial discussion earlier in the document regarding PIMs and PSMs, in MDA a PIM is not necessarily *independent* of all standards and encodings, it has to be based upon some rules (the example given was *relational databases*).

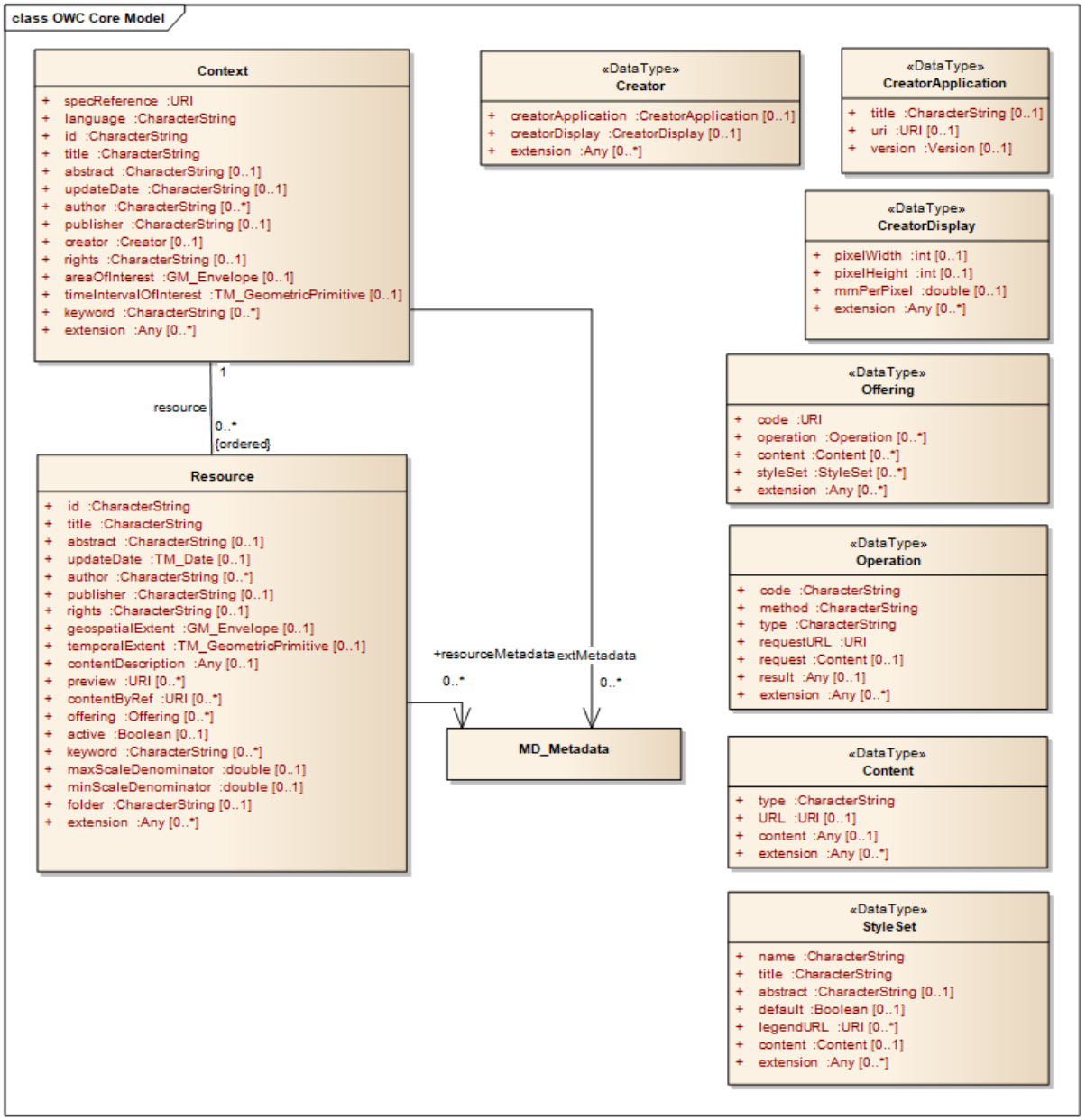

Figure 4. OWS Context Core Model

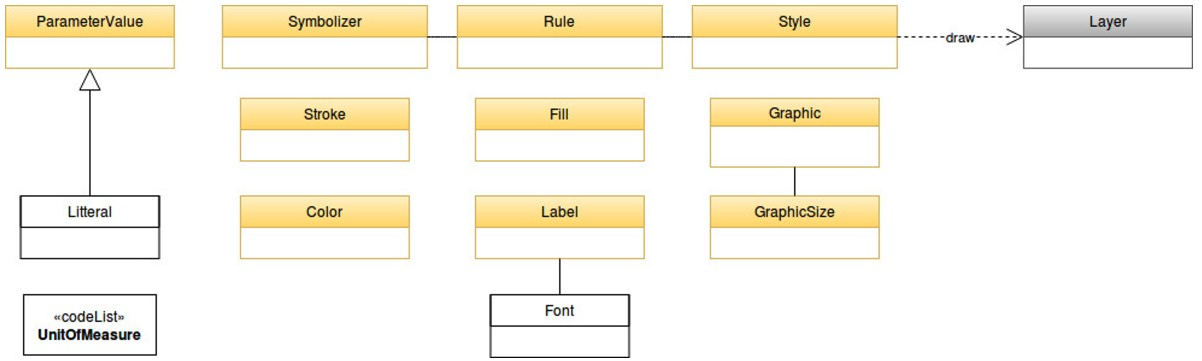

Figure 5. Symbology Core conceptual model

6. Model Driven Architecture (MDA)

MDA has been around in one form another for over 20 years. The principle behind the approach is that low level artifacts such as code, scripts and interface definitions should be derivable from higher level representations and models. The relationship between the high level representation and the low level artifacts should be one to many. The rationale behind this approach is that domain knowledge can be kept at the high level and implementations derived via transformations. In theory, this should enable simple, rapid and governed approach to deriving implementations or standard definitions from higher level models. An addition to generation of physical models from logical models is that the process should be reversible, in which case, all standards with physical models (schemas, for example), should be transformable into physical models. In terms of model levels, MDA utilizes PIMs and PSMs, rather than truly, abstracted *conceptual* modeling - again, this implies that that the PIMs conform to some rules.

Testbed-16 saw the utilization of MDA in the OpenAPI Thread. The objective of the thread was to produce OpenAPI compliant definitions from a UML model. This was achieved by creating the OpenAPI specification metamodel and then specializing the classes of the metamodel to create a model of the implementation. It should be noted that this is a logical model of a standard specifically targeted at OpenAPI, therefore to make it truly interoperable, an abstraction of the OpenAPI model would have to be created to enable target generation for standards other than those that are OpenAPI based. This will be a challenge as all information that is required for any target will have to be represented in the model somewhere.

Following on from the modeling discussion, the MDA process requires a logical model and a transformation process in order to generate a physical model (the artifacts required to create an instantiation). This begs the question of what the appropriate use if for MDA (if any)?

6.1. Appropriate use of MDA

There are two places in the standards definition process that MDA maybe utilized:

1. Standards definition - using existing models to create a new standard with extensions made according to the requirements of the domain.
2. Implementation generation - using an existing metamodel of a standard to produce an implementation by specializing the abstract model or specification.

Creating a standard using the MDA process maybe a good use of conceptual models, but it is reliant on the conceptual modeling artifacts having already been defined. However, it would encourage an in built governance process by requiring the use of already ratified concepts. Contrarily, MDA may not necessarily have a significant role in the standards definition process as the overhead and prior work requirement maybe deemed too high, a jump directly to implementation may suit the SWGs.

MDA should really be utilized for creation of *implementations* of the standard to fulfill a particular use case. In Testbed-16 OpenAPI, the OGC API - Features Part 1:Core metamodel was specialized to generate the interface definition in JSON, another transformation algorithm could be easily constructed to produce the same representation in YAML, alternatively, there is no reason why an XML representation of the interface implementation could not be constructed using the same process. This is the tried and tested method of using MDA as it has been proven recently in Testbed-16 and UGAS(2020) Pilot as well as being investigated in many prior Testbeds. At least if a PIM was defined for each standard, it would provide implementers the *option* of using MDA, even if it is not a requirement.

6.2. MDA Criticism

As MDA has been around for decades as a concept, a criticism of the approach is that is has not been widely adopted in that time. This is for a variety of reasons;

- Expertise in modeling is required as well as developers representing an overhead for business and a direct cost for governmental organizations.
- Models cannot be truly independent, they are based on a formalized language, usually UML (this does not imply that they are not useful). Additionally, logical models still have to have rules enforced.
- Models do not guarantee that all information required is captured formalized way due to the constraints of the modeling language and the trade-off between abstraction and pragmatism for MDA.
- The process can be more complex than simply creating an artifact from direct implementation.

Although this criticism is valid, MDA could potentially be employed in OGC in specific ways with a standardized set of tools. This approach does not require MDA to work in all cases for all models into all encodings, just the subset of each required by the OGC. The criticism leveled at MDA is valid, especially if an idealistic approach is adopted where models created are done so with the explicit function of being used in the MDA process. This is unlikely to be a requirement for a conceptual model, however, standards groups should be aware of the process and ascertain its utility. As mentioned in the previous section, providing a *normative* conceptual model along with the standard would at least provide implementers the *option* of MDA.

6.3. Management of concepts

An aspect that is missing from UML modeling, MDA and tooling in general seems to be an overall agreement on concepts. The current UML models are missing peripheral materials that are probably best held in a different format, for example, a data dictionary or business glossary for terms building on the existing OGC Glossary of Terms (http://www.opengis.net/def/glossary). If such a repository existed, it would assist in the semantic harmonization of concepts across the OGC - it would be like an OWS/OGC API Common but for terms that are agreed upon. This type of model would provide UML models with semantic meaning, rather than just describing the relationships in terms of its multiplicity, for example.

Some of the confusion regarding semantic descriptions of *concepts* and the way that the term *conceptual model* is being used in this paper. Concept models are concerned with semantics and capture of *meaning*, where as, concept models are concerned with engineering and relationships.

Characteristic	**Conceptual Model**	**Concept Model**
What is the core purpose of the concept model?	To initiate and/or orient the engineering of something	To create an inventory of defined words to make statements and to disambiguate those statements

Characteristic	Conceptual Model	Concept Model
What is the target of the model?	Some capability, which can pertain to inventories, processes, locations, roles, timing, or goals, or some combination thereof	Always concepts and only concepts, which can be about anything
Where are the things referenced in the model?	Often in the real world	Always in people's heads source 'https://rubygems.org'
What form does the model take?	Usually graphical, sometimes with definitions	Always definitions and structural statements, often illustrated for convenience by a graphical diagram
What kind of connections does the model feature?	Structural, sequential, spatial, collaborative, cyclic, or motivational -or some combination thereof- depending on the target of the model	Always semantic - either verbal concepts that can be expressed by verbs or verb phrases) or logical
What kind of engineering does the model address?	Any kind	Only knowledge, at least directly
How deep does the model go?	Usually the first in an ordered trio of models - conceptual, logical, physical - based on level of detail, consideration to design, and conformity to platform conventions	As deep as needed in expressing business knowledge; no counterpart kinds of model for higher-level or lower-level views
How is the model represented?	Usually graphically, sometimes with definitions	Always by vocabulary and definitional criteria, usually illustrated by a diagram

Although concept modeling is outside of the scope of this paper, which is focused on data and interface modeling, it is an important topic that should be explored further as part of the Architecture DWG or other relevant group. Perhaps this type of *concept* modeling could be used as a companion to the *conceptual model*, or in some cases, it maybe sensible to replace the conceptual model with a concept model depending on the value added.

7. Moving towards a unified approach to domain knowledge capture

So far, this paper has made the following assertions:

- The OGC has considered conceptual models and some standards have UML, conceptual models as supporting documentation, however:
  - The models are not in themselves standardized or harmonized with supporting models.
  - Some models are incorrect in terms of their use of UML and their representation of a particular standard.
  - Due to the time that some standards were released, their conceptual models were built to support XML only as an encoding.
  - The models are not necessarily accessible in a useful form, they are often represented as images.
- Standards generally express domain requirements in a standardized way. These requirements maybe for a particular application such as data distribution or processing or a domain such as EDM/LEAPS, Aviation and others.
- OGC standards have changed from being XML based by default to the option of having to support multiple encodings.
- Changes in domain knowledge are evolutionary (unless the domain is entirely new to the OGC), where as changes in technologies and specifications are revolutionary. This has resulted in translations having to be made from one technology to another, often without changing the domain knowledge. For example, OGC API – Processes is a copy of WPS 2.0, OGC API – Features Part 1:Core looks a lot like WFS 2.0.
- It is an open question regarding the amount of engagement that the OGC has outside of the technical committee on capturing domain knowledge. Given these points, it is potentially prudent to capture domain knowledge across the standards in a unified manner for the following reasons:
  - Capturing domain knowledge in a unified manner abstracts from any one specification or encoding. Therefore when there is a requirement for a new specification, it starts from the requirements of the domain rather than another implementation.
  - Cross-working group communication and knowledge store can be done in a standardized way. I.e. domain knowledge can be communicated without a requirement for a particular encoding or standard.
  - Generation of new encodings from high level models may be achievable through MDA therefore enabling a domain led approach rather than the current technology led approach.

Conversations within the TC membership and particularly the OAB over the past five years have yet to yield a consensus on the place, if any, of *conceptual modeling*, *UML* and/or *MDA* within the OGC and its development of standards. The big change recently is the shift to REST based, OpenAPI architectures and the need to also support XML based legacy architectures. This goes further than interfaces and also has an impact on data models and encodings, for example CityGML and CityJSON.

8. Recommendations

This discussion paper has sought to cause members to consider the use of conceptual models in the standards definition process. The following are a set of recommendations derived from the history of conceptual modeling considerations within the OGC, the recent work completed in Testbeds and derivations from peripheral projects and knowledge.

8.1. The terms *conceptual model* and *logical model* should be well-defined and used in an appropriate way

OGC has previously conflated the terms *logical* and *conceptual* when describing models. If these terms are disentangled and well defined then the appropriate model type can be used in the standards definition process. This will in turn help users to decide whether the modeling process would add value to the resultant standard. Coupled with this distinction, Platform Independent Models (PIMs) and Platform Specific Models (PSMs) should be used specifically in reference to the MDA process. Although PIM loosely translates to *logical model* and PSM loosely translates to *physical model*, the terms should not be used interchangeably.

8.2. Conceptual models should be recommended standards where they add value

Recommending the creation of conceptual models for new standards going forwards would start to enable the harmonization of models and standards across the OGC. Conceptual models should not be mandatory as OGC standards contain rules for implementation, and a model is not required for this to be a success. Models, however, do add value in certain circumstances. For example, if the model is entwined with the MDA process then it can be specialized to create implementations or used as a baseline to extend specifications for different domain requirements.

8.3. Conceptual models should be UML class diagrams when describing objects, relationships and structure

Although an older standard, UML is still the baseline for data models, it is also coupled tightly with the MDA process which should at least be an aspiration for standards developers in OGC. Although UML is OO, it has been shown through Testbed initiatives that it can be used to represent models that are not OO provided all of the information for a standard or implementation can be captured. In addition to providing modeling artifacts that are machine readable, UML also provides a standardized graphical interface that is well understood.

Concepts are difficult for UML models to describe as they lack the semantic depth of some other modeling languages. The OGC Abstract Specification contains (specifically OGC Core Tiling Concept) describes the Conceptual Model (opposed to a logical or physical model) as being the place where concepts and semantics are captured before being specialized into logical models for transformation into physical model (automated or manually). In contrast to this the use of *concept model* as being the place where concepts including their semantics are captured. This will need to be resolved, possibly by introducing another modeling paradigm to manage concepts and semantics.

8.4. MDA should be used where appropriate, or at least be an option for implementers

MDA is useful, especially when a base model can be used to derive many encodings. It also allows standards to broaden their encoding base through implementation of new transformations. A harmonized, model driven approach should remove some of the complexity from the process that was discussed in the criticisms of the process. Even if MDA is not recommended by the OGC in the formal sense, having a set of standardized, domain led conceptual models provides standards implementers with MDA as an *option*.

8.5. MDA is useful for new standards as well as new implementations

Taking a model driven approach to creating new standards may be helped with the use of modeling OGC API - Common components if suitable. This enables designers and implementers to achieve conformance with the common requirements easily as they could exist as predefined artifacts. It is recognized, however, that new artifacts would have to modeled to create new standards.

Perhaps the real power of MDA is best exploited when creating new implementations of standards fit for a particular domain. The modeling artifacts for a standard should already exist as part of the standards definition process (although perhaps in the abstract), therefore, these artifacts could be specialized to generate the model for an implementation of the standard to fit a particular domain.

8.6. A new Conceptual modeling group should be setup to support the Standard and Domain Working Groups

Conceptual modeling with or without the use of MDA is a value adding exercise to any standards definition process and should be encouraged throughout the OGC in groups with existing standards and those yet to be ratified. It may also be advantageous to utilize MDA to produce implementations and even new standards, this, however requires work on OGC API - Common and agreement on a methodology to enable this. Although the modeling is a useful exercise, it requires people with expertise in UML modeling and potentially MDA, these expertise likely exist already within the OGC, but there is no place for them to *live* apart from disparately in the existing groups. A new group should be setup to house these expertise and provide support to the DWGs and SWGs. This group would also have a birds-eye view of all of the modeling work across the OGC and could help rationalize and standardize relevant concepts, models and processes.

The OGC Abstract Specification should be the first point of call for developing new standards, however, it currently says very little about the use of conceptual modeling, and how this differs from *concept modeling* for example. The OGC Abstract Specification should be updated to provide more comprehensive guidance on the use of conceptual modeling within the standards defintion and implementation processes.

9. Conclusion

This OGC Discussion Paper has sought to answer the question regarding the purpose and place of conceptual modeling and associated technologies such as MDA. Conceptual modeling has been utilized within the OGC for sometime to different degrees. A few standards such as SOS used conceptually modeling to create their standard and drive the entire process, where as some of the recent OGC API Standards have not used a conceptual model at all. Conceptual modeling is a way of capturing domain knowledge that is independent of any specification or encoding, this is useful because implementation specifications and encodings change more frequently than core domain knowledge. Technologies such as MDA can assist in the automatic generation of implementations models that can provide substantive record of an implementation, ensure conformance and allow simple extension and regeneration of an interface. MDA could also be used to create new standards by ensuring conformance through OGC API - Common and provide a route to generating the standards in new encodings without having to start from scratch each time. OGC should not mandate conceptual modeling, but should support a group to assist, oversea and implement conceptual models for new and existing standards as required.

Annex A: Conceptual Modeling Group Terms of Reference

Regardless of the form that the conceptual modeling group takes, it requires a terms of reference to outline the backgound, activities and goals of the group as well as its place within the OGC. This annex provides an initial terms of reference.

A.1. Background

The OGC has been utilizing conceptual modelling within its standards definition programme for many years, almost since its inception. There have been numerous sponsored endeavours, mainly through Testbed initiatives to explore conceptual modelling and its place within the OGC. Conceptual modelling in the broad sense has been used as simply a diagramming technology, that is, as a method of communicating a standard's objects, relationships, and semantics, to utilization within Model Driven Architecture (MDA), where models are machine readable and transformed into target technologies for implementation. Conceptual modelling activities have resulted in a patchwork of models of variable intent, semantics, language, and quality. This has been brought to the attention of the OAB and members of the Architecture DWG and subsequently reported as OAB actions. It is felt that there needs to be some activity within the OGC to support DWGs and SWGs with conceptual modelling within their domains. Conceptual modelling should have the following characteristics, regardless of the domain in which they sit:

- Correctness – conceptual models should be correct. This means that they should represent the information that they are trying to show accurately, they should also follow the standard and best practice for the modelling language being employed. Although many of the conceptual models within OGC are UML, the OGC does not mandate this.
- Consistency – many standards reuse concepts from other standards; geometry representation is a prime example. There needs to be more communication across the SWGs and DWGs to ensure concepts described by modelling in one standard is semantically harmonized to all the other representations of the concept.
- FAIRness - The establishment and improvement of processes to ensure conceptual consistency occurs in a systematic and transparent is necessary. This may be described as FAIRness: Findable, Accessible, Interoperable and Reusable. The details of requirements and mechanisms needs a forum to allow for discussion and consensus to emerge, backed by tooling and infrastructure support for critical processes.
- Value – conceptual modelling should add value. Undertaking modelling is not a trivial piece of work, therefore the outputs should provide value over and above the rest of the documentation. The value added by modeling could differ depending on the standard; models maybe normative and are considered the gold standard, with the surrounding documentation supplementary information. Models can also be used simply to communicate information.

A.2. Group organisation, Activities and Goals

The Conceptual Modelling activity is being performed as a sub-group of the Architecture DWG. However, this activity may result in a DWG in the future if activity warrants it. The sub-group may call its own meetings as a special meeting of the Architecture DWG, with relevant agreement from the Architecture DWG Chair and co-chair. Alternatively, agenda items will appear on the Architecture DWG schedule as part of normal business. The proposed activities of the Conceptual Modelling Sub-Group are as follows:

1. Assist the Architecture DWG and relevant DWGs and SWGs on their modelling activities.
2. Create Best Practice for modelling within the OGC, this should follow on from the UML Best Practice document being completed as part of OGC Testbed-17.

3. Be a centre of knowledge within the OGC on conceptual modelling approaches, languages, and techniques.
4. Initiate a survey of existing and upcoming OGC standards, their models and seek to harmonize objects, classes, and relationships for common concepts.
5. Advise the Architecture DWG and other related Domain Standards on model creation and offer to be a point of review for such models.
6. Encourage the use of Conceptual Modelling within the OGC where models would add value.
7. Work with other modelling groups in relevant external organisations to ensure cross-domain and cross-organisation interoperability.

These activities lead to a series of goals for the group to work towards:

1. Improve the overall Conceptual Modelling work within the OGC.
2. Initiate a project to create and publish an overall modelling framework for the OGC using the Best Practice documents and OGC body of knowledge. This framework will likely be created iteratively with knowledge gained over time by standards creation, refinement and other initiatives such as Testbeds.
3. Provide DWGs and SWGs with suggestions or improvements to existing models held within standards to ensure correctness, consistency and value are achieved.
4. Provide the OGC with a body of knowledge on models within standards as they related to external organisations.

Last updated 2021-05-20 09:20:23 +0100