Semantic web platform and interfaces

Final version 1.0 | 25.02.2014

Deliverable Report: – Proficient D6.2 – Semantic web platform and interfaces - [Public]
Semantic web platform and interfaces
Publishable executive summary

CMO stands for Concept Modelling Ontology and is the main open standard used by the e-Marketplace to enable the parametric online configurators. The Semantic Web Platform is built completely on top of CMO. The interfaces are directly inherited from any existing Semantic Web (OWL2). As both IFC-to-CMO and CMO-to-IFC conversions are as alpha development available also any existing IFC interface is relevant.

In the research so far we focused on the requirements coming from other WP’s and the partners within PROFICIENT. This resulted in requirements for what needs to be modelled, but also to be able to apply the technology in a very early stage of the overall CSO processes. Next to this for the e-Marketplace itself not only configuration of knowledge is important but also the coordination of knowledge. Knowledge and experience from other projects but also the techniques and state-of-the-art solutions to collaborate and communicate within a CSO between housing owners themselves as well as potential suppliers of knowledge and services.

CMO itself is based on Semantic Web and acts as a small layer with extra limitations for ontologies on top of the existing Semantic Web technology OWL from W3C. Next to the standard abilities to communicate/interact with OWL content (including any CMO ontology) we also conducted conversion from/to existing and well known open ICT standards in the Building and Construction sector. An inventoriable of existing work on conversions is done, based on this inventorial most relevant work not yet available is identified and alpha software is created to cover this. This resulted in an alpha development of tools that enable converting IFC to and from CMO and the e-Marketplace. During the final development of the e-Marketplace these alpha developments will be improved to make real use of interaction with applications that import and/or export IFC as the most widely accepted and mature open standard.
List of acronyms and abbreviations

- **CMO**: Concept Modelling Ontology  
  http://www.modelservers.org/public/ontologies/cmo.ttl
- **RDF**: Resource Description Framework  
  http://www.w3.org/TR/rdf-schema/
- **RDFS**: RDF Schema  
  http://www.w3.org/TR/rdf-schema/
- **OWL**: Web Ontology Language  
  http://www.w3.org/TR/owl2-overview/ (OWL2)
- **IFC**: Industry Foundation Classes
- **CIS/2**: CIMSteel Integration Standards
- **CWA/OWA**: Closed World Assumption / Open World Assumption
- **SKOS**: Simple Knowledge Organization System  
  http://www.w3.org/2004/02/skos/
- **SPARQL**: A Query Language for RDF  
  http://www.w3.org/TR/rdf-sparql-query/
- **W3C**: The World Wide Web Consortium  
  http://www.w3.org/
## Contents

PUBLISHABLE EXECUTIVE SUMMARY 3

List of acronyms and abbreviations 4

1. DEFINITIONS AND GLOSSARY 8

1.1 Definition Section 8

1.1.1 e-Marketplace 8

1.1.2 CSO Housing 9

1.1.3 BIM 9

1.1.4 Semantic Web 9

1.1.5 OWA versus CWA 9

1.1.6 Layered Modelling 10

1.1.7 Ontology 10

1.2 Changes 11

1.3 Target Audience 11

2. SEMANTIC WEB 12

2.1 RDF / RDFS / OWL 14

2.1.1 RDF (Resource Description Framework) 14

2.1.2 RDF Schema (RDFS) 14

2.1.3 OWL (Web Ontology Language) 15

2.2 Reasoning 16

2.3 OWL2 16

3. CONCEPT MODELLING ONTOLOGY 17

3.1 CMO Definition 17

3.1.1 Archetypes 17

3.1.2 Typical Properties 18

3.1.3 Decomposition 18

3.1.4 Units 18

3.1.5 Topological Relationships 18

3.1.6 Requirements 18

3.2 CMO with Extensions 19

3.3 Expression Ontology 19

3.3.1 NullaryExpression 21

3.3.2 UnaryExpression 21

3.3.3 BinaryExpression 21
8.2.8 Line3D
8.2.9 Polygon2D
8.2.10 Polygon3D
8.2.11 PolyLine3D
8.2.12 Dim_2
8.2.13 Face2D
8.2.14 Dim_3
8.2.15 Blend
8.2.16 BoundaryRepresentation
8.2.17 Box
8.2.18 Cube
8.2.19 Clipping
8.2.20 Cone
8.2.21 Cylinder
8.2.22 ExtrudedPolygon
8.2.23 ExtrudedAreaSolid(Set)
8.2.24 Prism
8.2.25 Pyramid
8.2.26 Sphere
8.2.27 SweptAreaSolid(Set)
8.2.28 Dummy
8.2.29 Plane
8.2.30 Repetition
8.2.31 Transformation

8.3 Matrix
8.3.1 InverseMatrix
8.3.2 MatrixMultiplication

9. APPENDIX II - HARDWARE AND SOFTWARE
9.1 Hardware
9.2 Software
9.2.1 IFC Engine DLL
9.2.2 CSG Engine
9.2.3 doNetRDF library
1. DEFINITIONS AND GLOSSARY

The main objective of this chapter is to familiarize the reader with the specific terminology used in the context of the PROFICIENT project. In addition to new expressions and terms, this paper also includes some more ordinary terminology, in order to explain the scope and relevance of these in the context of this project.

1.1 Definition Section

1.1.1 e-Marketplace

The e-Marketplace offers tooling that supports the design process and brings demand and supply together. Both demand and supply are able to influence the design. The supply side is able to further refine possible solutions (configurations); based on the knowledge and access in the e-Marketplace this could be simple selections in existing templates or newly configured (sub) components selectable for the demand side. The demand side is able to make basic configurations with more or less freedom based on the professional level, knowledge and input from the configuration architect. The configuration architect defines all freedom professional and non-professional users of both demand and supply side have. The architect is able to merge models and knowledge from existing applications like CAD, libraries etc. towards the e-Marketplace based on widely adopted open BIM standards like IFC. The e-Marketplace will not only be able to visualize this information (in both 3D and properties), but the content can interact with other libraries, added knowledge/regulations on top and each individual configuration can be exported to the same CAD applications as well as towards calculation applications for heat/loss, performance, quantity take-off and many other via the same open standard.
1.1.2 CSO Housing

Collective Self Organized (CSO) housing refers to a group of individuals that acts in association to organise and commission the processes of formation, requirement definition, planning, design, implementation and / or maintaining their own housing project. A CSO housing project is typically characterized by a mutual dependency between the individuals participating. Participants have the right to step into contractual agreement (both on the individual and collective level).

1.1.3 BIM

BIM stands for Building Information Model(ling). The BIM becomes shared knowledge resources to support decision-making about the building from earliest conceptual stages, through design and construction, through its operational life and eventual demolition. PROFICIENT especially focuses on open standards available in BIM; IFC is one of the most, if not the most widely adapted open standard in BIM at the moment. Many of the drawbacks and limitations in current BIM standards can be solved via the use of Semantic Web technology; this is what CMO is trying to do.

1.1.4 Semantic Web

The Semantic Web is an initiative from W3C to extend the classic ‘web of documents’ into a ‘web of data’ to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. Instead of only storing and interlinking data also the meaning of the data can be described and interlinked. Based on triple storage and mathematical defined relations a meaningful ‘database’ of decentralised data can be created. The ontology Concept Modelling Ontology (CMO) is created and used to make Semantic Web useful for the e-Marketplace within PROFICIENT.

1.1.5 OWA versus CWA

CWA stands for Closed World Assumption: It is common to make information structure for a specific limited world, say a database for a company. Everything not in such database is considered “false”. If one is not in the personnel database of TNO one is considered not to be an employee at TNO.

OWA stands for Open World Assumption: Lately we see a transformation from closed systems to web-based system where information is distributed and never complete another interpretation is more useful: if info is not available it is treated as “unknown” instead of false: we don’t know yet. Seemingly subtle this has serious implications on the information modelling and processing.

Most current BIM standards are standards with a closed world assumption. This is true for standards in the area of product and process modelling, but even holds for standards in the area of data dictionaries. Semantic Web standards are typically based on the open world assumption. Within PROFICIENT we will try to bring them together enabling non-professionals, professionals from different disciplines and especially CSO’s (Collective Self Organized housing owners) to work together.
1.1.6 Layered Modelling

In semantic web we distinguish three abstraction levels:

- The language used
- The ontology described with the language, and
- The instances, here called individuals that comply to the ontology (the individuals are members of the classes defined in the ontologies).

These layers and example content is shown below.

1.1.7 Ontology

In computer science and information science, an ontology formally represents knowledge as a set of concepts within a domain, using a shared vocabulary to denote the types, properties and interrelationships of those concepts.

Ontologies are the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework.
1.2 Changes
If any of these definitions is considered to be incorrect or with an incorrect focus which has implications for any of the Deliverables in PROFICIENT, any person involved in the PROFICIENT project is entitled to send a request to change/alter, including the improved definition or text component, to the main authors of this deliverable (please find details in colophon).

1.3 Target Audience
This deliverable is meant for anybody that is interested in the Semantic Web Platform used within PROFICIENT, or in the interfaces this Semantic Web Platform delivers. The Semantic Web Platform is based on the open standard CMO and CMO with Extensions. This deliverable expects a technical background from the reader and will go into the technical details, issues and possibilities from the platform.
2. Semantic Web

In the world full of data we need a common approach to access and structure the information situated in different sources. We need not only to access it but to relate and organize it. The Semantic Web provides a common framework which allows data to be shared and reused between applications, enterprises and community boundaries. The Semantic Web is about two things:

- Formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents.
- It is also about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing.

The Semantic Web involves publishing in languages specifically designed for data: Resource Description Framework (RDF), Web Ontology Language (OWL), and Extensible Markup Language (XML). RDF, OWL and XML, can describe arbitrary things and objects such as people, meetings, or car parts. The HTML, by contrast, describes documents and the links between them. Semantic Web technologies are combined in order to provide descriptions that supplement or replace the content of Web documents. Thus, content may manifest itself as descriptive data stored in Web-accessible databases, or as Markup within documents (particularly, in Extensible HTML (XHTML) interspersed with XML, or, more often, purely in XML, with layout or rendering cues stored separately). The structure makes it easy to create an algorithm for parsers. Once the descriptions are machine-readable, that enables content managers to add meaning to the content, i.e., to describe the structure of the knowledge we have about that content. An application can process knowledge itself, instead of text, using processes similar to human deductive reasoning and inference, thereby obtaining more meaningful results and helping computers to perform automated information gathering and research.

The advantages of Semantic Web are found in practice by:

- Linking data - The Semantic Web is a web of data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. RDF provides the foundation for publishing and linking data. Various technologies allow to embed data in documents (RDFa, GRDDL) or expose the data stored in SQL databases, or make it available as RDF files.
- Vertical Applications - Semantic Web technologies can bridge many forms of information with different nature (medical, biological) across institutions.
- Inference - Near the top of the Semantic Web stack one finds inference — reasoning over data through rules.
Vocabularies - Using OWL (to build vocabularies, or “ontologies”) and SKOS (for designing knowledge organization systems) it is possible to enrich data with additional meaning, which allows more people (and more machines) to do more with the data.

Query - If the Semantic Web is viewed as a global database, then it is easy to understand why one would need a query language for that data. SPARQL is the query language for the Semantic Web.

The term “Semantic Web” refers to W3C’s vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS.

Above figure of the Semantic Web Stack shows the architecture of the Semantic Web. The Semantic Web Stack is an illustration of the hierarchy of languages, where each layer exploits and uses capabilities of the layers below. It shows how technologies that are standardized for Semantic Web are organized to make the Semantic Web possible. It also shows how Semantic Web is an extension (not replacement) of classical hypertext web. The bottom layers are currently well known and standardized. The topic of the current research is how to implement the top layers to achieve the full picture of the Semantic Web.
2.1 RDF / RDFS / OWL

2.1.1 RDF (Resource Description Framework)

RDF (Resource Description Framework) supports creating and processing meta data by defining a default structure. This structure can be used for any data, independent of their character. Thus, the application areas of RDF are numerous, e.g., web-based services, peer-to-peer networks, and semantic caching models, they all have in common that huge amounts of data have to be processed when querying RDF data. RDF data can be represented using XML, a triple structure or a graph. Only the graph representation enables the semantic interpretation of the RDF schema.

All of the elements of the triple are resources with the exception of the last element, object, that can be also a literal. Literal in the RDF sense is a constant string value such as string or number. Literals can be either plain literals (without type) or typed literals typed using XML Datatypes. These triples together form RDF graph. A normative syntax for serializing RDF is RDF/XML.

RDF reification allows to disassemble a statement (triple) to its parts and to use the whole statement or parts of the statement as a part of other triples. The whole triple can then be treated as a resource which allows to make assertions about the statement. For example, for the statement

:john :has :cat

the RDF reification is as follows (note that the result is resource that can for example participate as a subject in another triple):

[ a rdf:Statement;
 rdf:subject :john;
 rdf:predicate :has;
 rdf:object :cat ].

To summarize, RDF triple is a triple <subject, predicate, object> where subject can be URI or b-node, predicate can be URI, and object can be URI, b-node, or literal. RDF graph is a set of RDF triples.

The RDF data model does not make any assumptions about the application area in which the data is used. There are no reserved terms to model the data. Additionally, the RDF data model has no mechanism to define names for properties or resources. For that purpose, the RDF schema is needed to define resource types and property names. Different RDF schemas can be defined and used for different application areas.

2.1.2 RDF Schema (RDFS)

RDFS is extending RDF vocabulary to allow describing taxonomies of classes and properties. It also extends definitions for some of the elements of RDF, for example it sets the domain and range of properties and relates the RDF classes and properties into taxonomies using the RDFS vocabulary.

An illustration of the use of RDFS vocabulary is an example showing taxonomy of classes and properties and usage of range and domain of properties:

@prefix : <http://www.example.org/sample.rdfs#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

:Dog      rdfs:subClassOf :Animal.
:Person   rdfs:subClassOf :Animal.

:hasChild rdfs:range :Animal;
     rdfs:domain :Animal.
:hasSon   rdfs:subPropertyOf :hasChild.

:Max      a :Dog.
:Abel     a :Person.
:Adam     a :Person;
     :hasSon  :Abel.

All resources can be divided into groups called classes. Classes are also resources, so they are identified by URIs and can be described using properties. The members of a class are instances of classes, which is stated using the rdf:type property. Note that class and a set of instances does not have to be the same. The set of instances is the extension of the class, and two different classes may contain the same set of instances.

Properties in RDFS are relations between subjects and objects in RDF triples, i.e., predicates. All properties may have defined domain and range. Domain of a property states that any resource that has given property is an instance of the class. Range of a property states that the values of a property are instances of the class. If multiple classes are defined as the domain and range then the intersection of these classes is used.

In conclusion the RDF schema statements are valid RDF statements because their structure follows the structure of the RDF data model. The only difference to a pure "resource - property - value"- triple is, that an agreement about the specific meaning for reserved terms and statements has been made. Next to that, the RDF schema provides a vocabulary for defining the semantics of RDF statements.

2.1.3 OWL (Web Ontology Language)

OWL is a W3C standard. The abbreviation stands for Web Ontology Language and is a language for processing information on the web. It is built on top of RDF and RDFS. OWL was designed to be interpreted by computers and parsed by applications. It is not meant for being read by people. OWL is written in XML and has three sublanguages - OWL Lite, OWL DL (includes OWL Lite) and OWL Full (includes OWL DL).

The Ontology is about the exact description of things and their relationships. For the web, ontology is about the exact description of web information and relationships between web information. The standard OWL is a part of the “Semantic Web Vision” - a future where:

- Web information has exact meaning
- Web information can be processed by computers
- Computers can integrate information from the web

OWL and RDF are much of the same thing, but OWL is a stronger language with greater machine
interpretability than RDF. OWL comes with a larger vocabulary and stronger syntax than RDF. OWL extends RDF and RDFS. Its primary aim is to bring the expressive and reasoning power of description logic to the semantic web. Unfortunately, not everything from RDF can be expressed in DL. For example, the classes of classes are not permitted in the (chosen) DL, and some of the triple expressions would have no sense in DL. That is why OWL can be only syntactic extension of RDF/RDFS (note that RDFS is both syntactic and semantic extension of RDF). To partially overcome this problem, and also to allow layering within OWL, three species of OWL are defined.

By using XML, OWL information can easily be exchanged between different types of computers using different types of operating system and application languages. Note that there are several serializations for OWL, within PROFICIENT we promote using Turtle (.ttl), this serialization is not XML formed, but if needed it can be serialized in XML format at any time.

OWL became a W3C (World Wide Web Consortium) Recommendation in February 2004. A W3C Recommendation is understood by the industry and the web community as a web standard. A W3C Recommendation is a stable specification developed by a W3C Working Group and reviewed by the W3C Membership.

2.2 Reasoning

OWL enables “reasoning”, as mentioned above. That means it gives the possibility to check the logical correctness of statements and add statements that are implied by other statements.

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms to work with. The inference rules are commonly specified by means of an ontology language, and often a description language. Many reasoners use first-order predicate logic to perform reasoning; inference commonly proceeds by forward chaining and backward chaining. There are also examples of probabilistic reasoners, including Pei Wang's non-axiomatic reasoning system, and Novamente's probabilistic logic network.

2.3 OWL2

OWL 2 adds new functionality with respect to OWL 1. Some of the new features are syntactic sugar (e.g., disjoint union of classes) while others offer new expressivity, including keys, property chains, richer data types, data ranges, qualified cardinality restrictions, asymmetric, reflexive, and disjoint properties, and enhanced annotation capabilities.

OWL 2 also defines three new profiles [OWL 2 Profiles] and a new syntax [OWL 2 Manchester Syntax]. In addition, some of the restrictions applicable to OWL DL have been relaxed; as a result, the set of RDF Graphs that can be handled by Description Logics reasoners is slightly larger in OWL 2. Although slightly larger, some specific newly supported RDF Graphs are very relevant for CMO with Extensions.
3. Concept Modelling Ontology

Semantic Web technology developed by the World Wide Web Consortium (W3C) is the perfect medium for developing large-scale, web-based, multi-stakeholder information structures or ontologies that can be used for knowledge management and/or data exchange. Some modelling capabilities are however missing that we do not want to reinvent in every modelling case. These essential ingredients are modelled one time in CMO and then reused every time we need them.

3.1 CMO Definition

Concept Modelling Ontology (CMO) is a reusable, generic ontology that enables full-power, pure semantic, parametric modelling adding extra semantic modelling capabilities cleanly on top of the existing W3C OWL language like:

- Decomposition (via Dublin Core’s dcterms:hasPart),
- Quantities (via the NASA “QUDT” unit model),
- Typical properties and typical values,
- Static values,
- Links to documents & drawings (via “Dublin Core”).

This CMO is imported in all CMO/OWL2-compliant end-user ontologies. CMO is a fully open standard that can be freely reused without any constraints. This CMO is co-developed/used in several European projects (Odysseus, PROFICIENT, V-Con, EcoDistrict) by TNO (NL), CSTB (FR) & RDF (BG)) but also in the Dutch BIR CB-NL initiative developing an ontology (concept library) for the Dutch construction sector. In other words, a first version of this CMO has been developed using experience from several EC RTD projects and national developments in the Netherlands.

3.1.1 Archetypes

Within CMO the following distinct ‘archetype’ classes are defined:

- cmo:Concept
- cmo:Energy (with cmo.Concept as direct parent)
- cmo:Object (“anything in space”) == “Nouns” == “State” (with cmo.Energy as direct parent)
- cmo:Activity (“anything in time” == “Verbs” == “Behaviour” (with cmo.Energy as direct parent)
- cmo:SpaceTime (with cmo.Concept as direct parent)
- cmo:Space (like a 3D space) (with cmo.SpaceTime as direct parent)
- cmo:Time (start, end, duration, …) (with cmo.SpaceTime as direct parent)

Within CMO any user-defined class is a (indirect) subclass of either cmo:Object, cmo:Activity, cmo:Space or cmo:Time. At moment of writing this document the first priorities are subclasses of cmo:Object and cmo:Space.
3.1.2 Typical Properties

Following the idea behind Semantic Web CMO ontologies are advised not to use rdfs:domain. This means that when a property is relevant for a class but has no cardinality restrictions it is not directly possible to distinguish it from any other property that has no cardinality restrictions. A property can be defined a typical property in the context of a class such cases, it really is meat info over the property, i.e. that it is a relevant property for that class. Any property (both data type and object type) that has cardinality restrictions towards a class is a typical property of that class, even if maxCard = 0 is defined what means it is not allowed to exist in that class. CMO defined object properties hasPart, hasPartDirectly, isPartOf and isPartOfDirectly are already implicitly typical properties, no need to make explicit as typical properties. The same is true for meta-properties like typicalHasPartDirectly and of course typicalProperty itself (including all their inverses).

3.1.3 Decomposition

One main difference between Semantic Web and existing product modelling standards is that decomposition is missing in Semantic Web. CMO added decomposition as a relation (object type property). Decomposition can be defined on instances as well as on classes. In both cases there is a distinction between direct decomposition and potentially nested decomposition. Note that CMO does not prevent you from making loops in nested decomposition structures, this is however conceptually not allowed.

Where possible decomposition constraints are modelled by Qualified Cardinality Constraints (QCR’s) to keep the ontology as modest as possible,

3.1.4 Units

Units are a direct reuse of the ‘NASA model’: QUDT (http://qudt.org). Units are modelled for the quantitative data type properties on class level. It can be seen as replacing the definition of an integer or float by a unit definition.

3.1.5 Topological Relationships

This part of CMO is not yet perfectly defined at the moment of writing this document. We make a distinction between 0D, 1D, 2D and 3D, similar to the GeometricItem subclasses (see also Appendix I). Also this part of CMO will add topological relation types like A is boundedBy B, or A is in B. As this comes very close to the original Set Theory it can be part of CMO itself.

3.1.6 Requirements

Next to Topological Relationships this is also a new part of CMO. Where Concept in CMO stands for product as well as for process there is a different distinction to make. Is a concept a requirement or is a concept a designed or as-is definition. This meta info about the concept is important in many practical situations and will be part of CMO, the exact definition is not crystallized yet however.
3.2 CMO with Extensions

As CMO acts like a very thin layer on top of OWL it is necessary to have larger extensions to OWL to enable support for parametrics and parametric geometry. Similar to CMO, CMO with Extensions acts like an extra set of restrictions on top of OWL2, where the restrictions are written as ontologies. This makes CMO with Extensions fully compatible with CMO and therefore also with OWL2.

CMO with Extensions adds three extra ontologies:
- Expression Ontology,
- Relation Ontology,
- Geometry ontology.

Together they enable creation of parametric geometry. In the next chapter the link towards other open standards will be discussed. We will see that especially on the geometry part conversion from IFC towards CMO with Extensions and back is possible. The current prototypes created as part of PROFICIENT are already able to have scalable and nearly 100% working conversions for the geometry part of IFC and CMO.

3.3 Expression Ontology

The Expression Ontology can be seen as the ontology to store formulas. The formulas can be stored as a network. Each formula can be represented by a tree where a branch can be split into 2 new branches, one new branch or ending as a leaf. Different to most tree structures it is possible that parts of a single tree are repeated in the tree structure or be dependent. Also the three does not need to be completely defined.

Technically an addition of (datatype) properties ‘a’ and ‘b’ is a subClass of Addition where the firstTerm is restricted to ‘a’ and the secondTerm is restricted to ‘b’. If a subClass of Multiplication is created that has its firstTerm restricted to ‘c’ and the secondTerm restricted to the subClass f Addition we have the expression:

\[ c \times (a + b) \]

Note that any subproperty of resp. ‘c’, ‘a’ and ‘b’ can be filled in. Note also that a subClass of a BinaryExpression does not need to restrict both terms as well as a restriction does not need to be limited to one other expression or property.
Important in the definition of operations is that the inverse solution is known and defined, as well as the order of importance. This enables applications to rewrite formulas when used in the context of equations.

For example when the following equation is given:

\[ a = d \times (b + c) \]

If \( b \) has to be calculated the application needs to be able to rewrite it into:

\[ b = \frac{a}{d} - c \]

It is clear this is impossible without knowing the inverse relation of multiplication. In the sense of putting as much as possible knowledge in the data adding inverse relations is enabling applications that need to rewrite equations (and more generic relations) to be more abstract.

An example of importance of order of operations:

\[ a + b \times c \] equals \( a + (b \times c) \) and NOT equals \( (a + b) \times c \)
The fact that within the standard it is not possible to enable more than binary operations does not mean this has to be also the interface towards the user. For example:

\[
\text{Sum}(a, b, c, d, e) = a + b + c + d + e = (((a + b) + c) + d) + e
\]

All three representations result in the same result, however only the last one is possible to store semantically in CMO with Extensions. The application using CMO with Extensions is however free how to represent data given or how to convert incoming data.

### 3.3.1 NullaryExpression

A Nullary Expression defines either a fixed value or a (datatype) property. Nullary Expressions are exclusively the leaves of a formula tree.

### 3.3.2 UnaryExpression

An Unary Expression defines the expressions based on a single other expression or (datatype) property. Typical examples of an unary expression are Sin, Cos, but also non invertible expressions like Abs and Not. Although the data type of Not can be different from its used expression (i.e. it returns a Boolean (true or false) and can work on integer, float, string etc.) CMO with Extension does not store such limitations in the ontology.

### 3.3.3 BinaryExpression

A Binary Expression defines the expression on top of two other expressions. A typical example is addition. Not needed but potentially new binary expressions are binary operators as AND, OR, XOR etc.
3.4 Relation Ontology

Being the smallest ontology of this extension this ontology focuses on connecting parametric knowledge towards the classes. Where the Expression ontology is

![Diagram of owl:Thing, owl:Nothing, Relation, Assignment, Equation]

3.4.1 Equation

An equation exist of two expressions that are defined to be equal. A typical example could be:

\[ 2 + 3 \times a = b + c / 2 \]

The following equations say exactly the same:

\[ a = (b + c / 2 - 2) / 3 \]
\[ b = 2 + 3 \times a - c / 2 \]
\[ c = 4 + 6 \times a - 2 \times b \]

As part of PROFICIENT an alpha version of a library is created that enables rewriting of these equations depending on the need and depending on other equations (if available).

3.4.2 Relation

A relation is a generalization of an Equation where we discriminate between the following types of relations:

- `<` Smaller Then
- `<=` Smaller Then or Equal
- `=` Equals (Equation)
- `!=` Not Equals
- `>` Larger Then
- `>=` Larger Then or Equal

3.4.3 Assignment

The assignment is a one directional equation. One expression is assigned to a (datatype) property and the property is also really given a value by this expression. This means that the property will not drive the knowledge in the expression. For example:

\[ a := b + 3 \]

If somehow the value a has already a value 5, in an equation this would imply that b gets the value 2. In case of an assignment an ambiguity warning should be fired in case b is not exactly given value 2 somewhere else in the calculations.
3.5 Geometry Ontology

The Geometry Ontology contains the basic geometry blocks available in CMO. Based on this basic geometry concepts new concepts can be created. The application that visualizes the geometry only needs to know and being able to represent the basic Geometry Ontology. All user created geometry concepts based on this ontology can be visualized by the same application by applying the rules as defined in the data. Different from most standards complicated parametric definitions can be defined in data by using typical Semantic Web structures.

For example different profiles can be defined by drawing the exact geometry of a beam or by defining parametric values from a profile type. Well known standards like IFC and CIS/2 have the ability to support such parametric definitions, for example the T-Profile in IFC 2x3 (similar in IFC4):

![T-Profile Diagram]

Especially such parametric Profile descriptions are used frequently as a lot of non-geometric knowledge (for example for strength calculations) depends on the parametric definition.
Within CMO with extensions we do not have a T-profile defined. However it is possible to define it completely in a new ontology based on the given geometric ontology above. The exact same parametric definition can be created by modelling its dependencies parametrically:

Using this user defined ontology it is now possible to apply it to a beam filling in the dimensions parametrically (Flange Width, Fillet Radius etc). The viewer component can then visualize the beam based on the parametric values; these parametric values also can be used for non-geometric use like strength calculations for example. Below beam based on this profile and a railway track using this beam.
Similar to the CIS/2 and IFC standards we are able to define parametric profiles and visualize them. The important difference is however that in CIS/2 and IFC every application that wants to visualize the beams using these profiles has to implement and understand the parametric definition correctly. This is error prone, time/resource expensive and reducing flexibility. Within CMO with Extensions this knowledge is present in the data and the viewer component therefore has to understand only a fraction of the geometric concepts available in most open standards. Not only this, also creating new parametric definitions does not require software updates. This is not only beneficial for the amount of work done on application side, also:

- it improves the potential quality of standard certification (often the limited geometry quality of certified application for open standards is due to the large set of potential combinations and complexity in checking this in an automated way),
- it enables alternative parametric definitions and flexible sub standards that are geometrically 100% understood as long as they use CMO with Extensions.

The tree structure as visualized next to here is described in more detail in Appendix I.
3.6 Example

To give a small impression of the modelling capabilities using CMO with Extensions we created a small example. Within the Modelling Guidelines a glimpse of some more detailed examples is given.

Above a Perforated Metal Sheet based on the following parameters

- depth
- diameter
- height (not in picture, but represents the thickness of the metal plate)
- length
- distance
- xOffset (minimal value)
- yOffset (minimal value)

The following existing concepts are used:

- BooleanOperation (using difference)
- Box
- Collection
- Transformation
- Repetition
- Cylinder
- Matrix
Above the used properties within building this concept are shown, as well as the rules that apply to this concept. Some of the used properties have to be assigned a value for the system to calculate an instance of this concept.
3.7 Future Extensions

Based on review of latest version of open standards and testing content we foresee two important parts that are missing in the current definition:

- Splines in general for surfaces,
- NURBS (Non-uniform rational B-spline).

They are not yet defined as the use in Building & Construction industry is limited, apart from some early test files no content for open standards is available yet. The typical use and required parameters are therefore not clear yet. The expectation is however that this will change in the coming years, as soon as the use is wider adopted and typical use is better known this is planned to be an update of the Geometry ontology.
4. Interfaces

The technical interfaces are clear as CMO is based on Semantic Web and Within the Building and Construction the choice for open standard IFC is logical. Both technical interfaces are discussed in 4.1 and 4.2. The user interface will be explained in more detail in 4.3.

4.1 Interfaces via underlying open standards

As CMO and also CMO with Extension are both fully OWL2 compliant any OWL2 supporting tool can read, write and edit the content.

Generic software applications are available, both open source and closed source products:

- TopBraid Composer (commercial closed source application)
- Protégé (open source development)
- Reasonors, like FacT++, Pellet, HermiT, TrowL and CEDAR

Several API’s are available:

- OWL API
- HP JENA
- dotNetRDF

Both for tools and API’s named above the majority of the software is developed under JAVA.

On a higher level typical Semantic Web technology is available to retrieve subsets of the data via query languages that work on several ways of storing the ontologies. SPARQL is the most used Query language.

4.2 Interfaces via external open standards

To be able to access and interact with existing content and software it is important to connect in both directions to existing software solutions that create, edit and use typical content relevant for CMO.

As we want to support a wide range of products a wide range of disciplines within the sector and prevent ourselves from building an interface or conversion tool for every application we want potentially to connect to it is important to connect to well known, used and mature (open) standards. A good choice within the Building and Construction sector for standard to support is IFC from BuildingSmart.

The roots of IFC can be found in 1985, the early days of STEP/EXPRESS modelling and also the roots from famous standards like AP214 used often in automotive and aerospace. In 1994 Autodesk started a fork called IFC and pushed it into a open standard controlled by BuildingSmart since 1995. It took at least another 10 years before the standard was really used within commercial projects, the real use started...
with version IFC 2x3 TC1 what is still the most widely supported version in early 2014, even though a new version IFC4 is available.

While adaptation in the market took over 20 years from the initial start, the integration in the market of software products and the availability of tools supporting implementation and use is now growing quickly. Many companies are using IFC now professionally and this group of companies is growing rapidly. This not in the least due to efforts from BuildingSmart in professionalizing the standard, moving it into an ISO standard, improving and automating certification processes. A new version IFC4 is released in 2013, it is expected to be supported by the main software vendors end of 2014 or early 2015.

Conversion between IFC and Semantic Web is a subject of interest for many years. In research for existing conversions and companies working on conversions we found 10 companies busy with these developments and 8 real conversions.

All conversions were from IFC towards Semantic Web and schema independent. This is important due to its benefits and drawbacks. Also it is important to note that conversions in the other direction could not be found.

An important benefit of a schema independent conversion is that the converted data can still be called IFC data, with this it benefits from the status of IFC and the years of research behind. Another important benefit is that all data can be converted to semantic web, even if the schema slightly changes what happened in version TC1 (Technical Corrigendum) of IFC2x3 and now again with the new version IFC4.

The idea is that once being in the Semantic Web domain conversions can be done with existing tools available within the Semantic Web context.

In the context of PROFICIENT we decided to disagree with two (implicit) assumptions:

- conversion from Semantic Web to IFC is not relevant
- conversion to real Semantic content within Semantic Web context is possible with available tools today

The first assumption is from organizational viewpoint easy, as no existing conversions can be found and from the relevant parties nobody is known to be busy within this area we created our own alpha development for conversion from Semantic Web towards IFC. As we especially focusing on CMO with extensions our content is limited to Semantic Web content also in line with CMO with extensions. The alpha development including source code can be found at the PROFICIENT internal website.

The steps to take based on the second assumption are less clear. As there is so much work done already in this area it is a pity and even impossible within the available time to ignore all existing developments and redo everything. The assumption is mainly expected to be true for the geometry part
of the IFC schema. As there is a conversion available from IFC geometry towards a more generic geometry representation within a development that applies to the ‘rules of commercial software’ (see Appendix II, embedded software), it is possible to use this as a base for the conversion.

What is done is create the CMO with Extensions for geometry ontology and apply this structure to the internal structure of the IFC Engine DLL, also adding an API call enabling exporting CMO content created a first example for conversion of IFC content into Semantic geometry content.

Important assumptions:
- knowledge is kept as intelligent as possible after the conversion
- conversion is complete and works for both IFC2x3 TC1 and IFC4

In the current version we still lack these assumptions partly, for example extruded polygons are faceted into a polygon existing of line parts (even though the original IFC file could have defined circle parts) and then applying the extrusion. Important knowledge like Boolean Operations and Clipping are however kept in place to be able to be reused in Semantic Web context (i.e. Clipping planes or Boolean Operations could be driven parametrically giving the need to store the original structure).

Both applications are alpha developments but working and scalable to real life model sizes.

As the converted content in CMO applies to the CMO with Extensions definition it can be visualized as such with the Geometry Kernel component.

4.3 User Interfaces

The user interface is depending on the requirements from other WP’s within PROFICIENT. From the Semantic Web point of view this can mainly be divided a part about the type of content and the way of bringing this to the user.

The type of content is slowly getting clearer; the idea behind CMO (and Semantic Web) is that it is so flexible that whatever is needed is able to be stored in CMO. In practice this will not be the case, however at the moment it is not expected that not being able to store certain knowledge within the Semantic Web Platform will be an issue.

The way of bringing this content to the user is a bigger question. At the moment the most clear requirement is that the user want a clear, clean and easy interface. As the user will often access the platform from a private location without control over the system it seems logical to look at Web Based interfaces. For non geometrical data several standard solutions are available like WebProtege, however the usefulness is unclear at the moment as these developments are not ready yet and seem to be limited in flexibility. For the geometry some first client server implementations are developed to enable the parametric 3D on a client via a web interface.
4.4 IT Architecture and Infrastructure

The IT architecture of the Semantic Web Platform is shown in the figure below. The CMO Content can be created by either directly interfacing Semantic Web by standard tools described in 4.1 or by existing CAD applications and libraries via the IFC-to-CMO conversion tool.

In the other direction again the same technical interfaces are present, both directly via Semantic Web or indirectly via the CMO-to-IFC converter technical connections can be created.

The user interface handled by the configurators will be a web based interface as described in 4.3.
5. Modelling Guidelines

This chapter is meant for everybody that wants to benefit from our lessons learned in the past year modelling CMO. Next to that this chapter also gives some more detailed examples of the power and capabilities of CMO. All example ontologies and tools can be found on the internal PROFICIENT website.

CMO with Extensions, developed in the scope of PROFICIENT defines a new conceptual way of modelling. That means not only a different way of accessing the content as described in the chapter about interfaces, but it also means that new ways of modelling knowledge is needed. To be able to model (complicated) knowledge it is not only needed to be able to store the knowledge but it is also needed to know how to model it correctly.

During the development of the standard CMO with Extensions modelling examples were created simultaneously to find potential issues in the standard but also to be able to create guidelines how to create content.

5.1 Generic Findings

One of the typical benefits of CMO is the potential for fragmentation of knowledge and reuse of these knowledge parts. The following findings are especially important as a consequence of this fragmentation:

- introduce a naming convention in the ontologies created,
- when building a library of knowledge/components/ontologies be aware that individual projects have a strong push to make small modifications to this library, this quickly results in out of sinc data. The best solution seems to, in the mindset of Semantic Web, really enable projects to access distributed libraries and arrange editable rights correctly,
- when creating libraries don’t mix different types and different abstraction levels of data, instead it is better to increase the number of ontologies,
- create (and review from time-to-time) a clear and detailed definition of available types of knowledge (ontologies) and the structure used to separate and integrate them.

One of the specific features form OWL and therefore CMO is that properties (including relations as object type properties) are defined independent from classes. For people that are used to original closed world modelling it is recommend to keep the number of classes per file limited
5.2 Simple Roof Example

The real interesting cases start when it is not directly clear how to solve modelling. Although the following picture looks simple the concepts behind can be already challenging. Actually we will only look at the roof.

To be able to model the following house (and for this case only the roof) we need to understand what are the driving properties behind the geometry. Let’s say the roof is defined by the following properties:

- length
- width
- height
- thickness

Given these basic understandable values the question is how the roof should be drawn. Whatever concept we will use to create the drawing we will need to fix the following abstract challenge:

The basic idea behind the challenge is that w, h and t are given, but to be able to make the real drawing we need the value for x. We will use this first example as a base for the other examples.

The first ontology to create is Parameters, here the relevant parameters are defined. The second ontology created is the Equations ontology, here all the knowledge we have about the issue needs to be put into formulas, in this case we typically inherit the knowledge from a library containing generic version of a QuadraticEquation. The third (not counting used libraries) ontology defines the knowledge we know. In case of our roof we could know for example a value for w (half the roof width) and h (roof height).
Based on this we can create two possible options:

- \( t \) is given (roof thickness) and \( x \) has to be calculated
- \( x \) is given and \( t \) (roof thickness) has to be calculated

Note: the problem becomes more interesting (but not even so much more complex) when also the roof angle could be a potential input value.

In the picture above also the representation of the issue is given, this can help defining the problem as well as the potential solutions/options. Although both ontologies Representation and Options are not relevant for the eventual roof ontology they are handy to create for understanding the problem, especially when problems are getting more complicated.
Above you find a picture of the Representation ontology. The same picture is used to define both options.

And finally the ontology used to create the visualization as shown in the beginning. Note that this is ontology is used purely for the roof and clipping of the house body.
5.3 Complex Examples

Although the previous example already can look complicated for a relatively simple situation it is expected that increasing scale and complexity will be possible without increasing the complexity in the solutions too much.

As we were in the start of the PROFICIENT project when selecting good test data and the actual detailed use cases still had to be cleared with project partners it was needed to create an artificial example containing complexity and being large enough to test scalability as well. After an extensive search in the area of PROFICIENT and with project partners we ended up with a completely different example, namely the LEGO 8448 model.

Even though this is not a typical PROFICIENT example, it covers exactly what we want to test within the Semantic Web platform:

- scalability in Semantic Web representation as well as visualization
- the modelling of complex parametric relations

The resulting model is a combination of 268 distributable interlinked CMO files and the amount of vertices and indices is similar to the amounts we find in larger CAD models that are exchanged with the open standard IFC.
5.3.1 Engine Cylinder Example

The first example as part of the LEGO model is about the Cylinders, the abstract issues is about the way every Cylinder is moving based on the rotation of the axis and the angle the cylinder is placed in.

The cylinders above are parts of a larger 8 cylinder engine block. The question is how to model the axis rotation and movement of the individual cylinders.

In the first picture on the left we find the abstract version of this problem where angleI is the rotation of the axis and where e and f are given lengths of the abstract construction. In the right picture a more advanced version where (angleI – angleIII) is the rotation of the axis, however what is making it extra complicated is that the direction of the cylinder is defined under angleIII. Note, that in the situation above we have cylinders with angleIII = 45 degrees and cylinders with angleIII = 135 degrees.
Looking at the knowledge we have, we can define the following rules:

\[ a = \sin(\text{angle}I) \cdot e \]
\[ b = \cos(\text{angle}I) \cdot e \]
\[ f^2 = c^2 + a^2 \]
\[ d = b + c \]
\[ \text{angle}II = \arcsin \left( \frac{a}{f} \right) \]
\[ g = \sin(\text{angle}I - \text{angle}III) \cdot e \]
\[ h = \sin(\text{angle}I - \text{angle}III) \cdot e \]
\[ l = h + i \]
\[ j = g + k \]
\[ i = \sin(\text{angle}II + \text{angle}III) \cdot f \]
\[ j = \cos(\text{angle}II + \text{angle}III) \cdot f \]
\[ k = j - g \]

Also in our specific example we know the following information:

\[ e = 0.4 \]
\[ f = 1.785 \]

Given all knowledge above the system should be able to deduct all values when \text{angle}I and \text{angle}III are given as input values. The following image shows this works (\text{angle}I set to 60 degrees and \text{angle}III set to 45 degrees):
The following ontology is created based on this. Using this knowledge the cylinders can be drawn and the complete engine can be created. In this picture the engine with cylinders parametrically working embedded from below:
5.3.2 Suspension Example

This example is about a suspension construction. The following picture shows the actual situation:

Based on this we can already write down the following equations:

\[
\begin{align*}
    a &= \sin(\text{angleII}) \cdot h \\
    b &= e + l - a \\
    c &= g + i + j \\
    f &= d + i \\
    e &= \sin(\text{angleI}) \cdot k \\
    g &= \cos(\text{angleII}) \cdot h \\
    j &= \cos(\text{angleII}) \cdot k \\
    n^2 &= b^2 + c^2 \\
    a + b &= e + l \\
    k^2 &= j^2 + e^2 \\
    \text{angleIII} &= \arccos\left(\frac{c}{n}\right)
\end{align*}
\]

In our example we have the following fixed values:

\[
\begin{align*}
    d &= 1.6 & h &= 2.4 \\
    i &= 2.4 & k &= 3.2 \\
    l &= 0.8 & m &= 1.6 \\
    \text{angleII} &= 180 - 127
\end{align*}
\]
This results in 3 typical unknown values, angleI, angleII and n. We have 2 different options:

- When angleI is given angleII and n can be calculated
- When angleII and n are given angleI can be calculated

In our specific example we are mainly interested in the first option, so given an angleI the rest should be calculated. The following ontology is created based on the abstraction of the issue.

In practice it means we are able to apply a suspension into the model:
5.3.3 Crank Shaft Example

This last example is a bit different from other examples as it contains reusing several instances of another issue. The main real and abstract issue to solve:

It actually represents a complex combination of the following example:
The small Crank Shaft Example results in:

Applying these rules inside the bigger picture for the complex Crank Shaft example:
5.4 Future Examples

One thing found during developing all kind of constructions and learning how to solve complex structures in a generic way is that what we are missing in the current solutions are the concept of Ports.

Even though putting extensive effort in finding a generic solution for this it is not yet solved. It seems possible to handle ports with only applying guidelines and examples where this is done can be found on the PROFICIENT website, but the issue was to complex to create clear guidelines for this until now.

Also the real ‘proof is in the pudding’, as PROFICIENT prolongs the requirements for modelling power will become more and more clear. The current modelling examples are taking based on discussions with project partners and generic known modelling issues in the area of parametric modelling.

5.5 Disclaimer

Although the modelling guidelines are made with the biggest care and by creating a lot of different content it has to be taking into consideration that the guidelines are based on experience during development of CMO.

When new experiences will be influencing or extending the guidelines, the updated guidelines will be added to the e-Marketplace deliverable.

As CMO is partly based on earlier EU projects like SWOP and ManuBuild modelling knowledge is based on more then only the first 18 months of PROFICIENT.
6. Conclusions

This chapter is meant for explaining how this semantic platform will be used to connect user’s/supplier’s software and web-applications. To give a clear image of how this platform will be used, first some well known state-of-the-art examples of competitors.

Afterwards we will go into the PROFICIENT architecture, see also the picture used before in the definition of the e-Marketplace.

For both supply and demand side we are making distinctions between professional users that are able to change structures (in technical terms create and edit classes) and non-professional users that only make configuration selections (in technical terms create and edit instances).

6.1 Current State-of-the-Art Platforms

Platforms, configurators and e-Marketplaces normally lead to dedicated software that needs to be reconfigured for each situation. Typical well known state-of-the-art examples are:

- eBay.com/Marktplaats.nl, a perfect example of an e-Marketplace where both supply and demand site can define instances according to predefined structures. Three important things are missing for a CSO Housing platform however:
  - Structures are defined by eBay itself and cannot be applied by an external architect,
  - Only non-professional users can apply items, it is not allowed to extend structures or create/edit classes as professional users would like to,
  - Limited data structures, no support for multiple inheritance (only grouping mechanisms) and no support for different connected aggregation levels, i.e. I want a set of doors in the context of my house. This also results in no support for real collaboration between suppliers of demand site partners.
• IKEA configurators
  o The supply side is professional but only internal. Nevertheless also this platform is a solution with only internal access to the professional structure,
  o Given the limited amount of types (only kitchen and office configurator) it is expected that there is code dependency for each individual configurator.

• House configurators (for example: http://www.lekkereigenhuis.nl/configurator.html)
  o A great looking configurator, although only the demand side is targeted,
  o The main issue with such type of configurators (also found for cars, bicycles, clothing etc.) is the code dependency in the configurator itself. The solution is simply too complex, time consuming and expensive to initialize for an individual CSO project.

We are expected not only to cover technically all above cases. We also should be able to do this in a generic way where the structures are contained in the data and not in dedicated software. This includes the expectation that geometry should be supported and this geometry has to be depending on parametric behavior.

### 6.2 Semantic Web Platforms

The potential of Semantic Web is also seen by large companies, proven by companies like HP investing heavily in applications like JENA and ORACLE supporting the underlying triple store and structures from RDF standard in its professional databases. Nevertheless it is still a solution of the research area with mainly implementations in JAVA and huge issues scaling up content towards real life models without losing significant performance. Within this deliverable of PROFICIENT we were able showing that it is possible to bring this very promising technology to the commercial C/C++/.NET world and being able to scale up towards real life models by making very limited restrictions to the use of Semantic Web. These restrictions are combined in the open standard CMO and with parametric geometry solutions enabled in CMO with Extensions.

From a user point of view the benefits are the large potential flexibility in modelling the solutions, the independency of dedicated software and reusability of data from previous projects. It is clear that creating a complete eBay system or even an IKEA-type configurator for each individual CSO project will be far too expensive and time consuming. The creation of a generic system where the knowledge is mainly kept in the data, where the data of previous projects can be reused and where input directly from the mayor CAD systems is possible enables an e-Marketplace possibility for each CSO project where this was clearly not possible before.

As the solution is very generic it can really be seen as a platform supporting different steps in the process, i.e. the Semantic Web Platform. The exact situations where to use this platform depends on the different processes executed for a CSO Housing project. After enabling this CMO technology and platform in collaboration with the other WP's within PROFICIENT the practical implementation and use of it will become clearer. Both the technical interfaces towards existing OWL/Semantic Web tooling, IFC importing and exporting applications and the user interfaces that are web based with an existing
WebProtege solution and a WebGL enabled 3D parametric web based interface will help supporting the different use cases of the e-Marketplace.

All together, the PROFICIENT Semantic Web platform is a perfect example of enabling the power of Semantic Web in a scalable, professional environment; making distributed knowledge available for end users and (sub) component suppliers, including non-professional parties.
7. References


http://www.wikipedia.org/  (last accessed 28 Jan 2014)

http://www.w3.org/  (last accessed 28 Jan 2014)
8. Appendix I - Geometry Concepts

8.1 Appearance

Appearance can be seen as an abstract class as it is just a collection of Color, ColorComponent, Material and Texture.

8.1.1 Color

An instance of this class defines a color. It contains the following four optional single relations:
- ambient (optional ColorComponent)
- diffuse (optional ColorComponent)
- specular (optional ColorComponent)
- emissive (optional ColorComponent)

Each relation is restricted to ColorComponent (or any subclass of it). Color also contains an alpha value of type double. Alpha is a double which should be between 0.0 (transparent) and 1.0 (opaque). A value of 0.5 will be half transparent, meaning that whatever is behind this object will contribute to 50% of the final color. Alpha overrides the W values of the different color components when defined.

8.1.2 ColorComponent

The ColorComponent contains 4 double values:
- R a double value which should be between 0.0 (no Red) and 1.0 (100% Red)
- G a double value which should be between 0.0 (no Green) and 1.0 (100% Green)
- B a double value which should be between 0.0 (no Blue) and 1.0 (100% Blue)
- W a double value which should be between 0.0 (no White/Transparency) and 1.0 (100% White/Transparency)

8.1.3 Material

A Material references optionally to one or two textures and one optional color.

The used Relations
- textures (optional set of 1 or 2 classes of type Texture)
- color (optional Color)

8.1.4 Texture

Each texture has the following optional double values:
- offsetX is an optional double value relative between 0 and 1
- offsetY is an optional double value relative between 0 and 1
- scalingX is an optional double value relative between 0 and 1
scalingY is an optional double value relative between 0 and 1
rotation is an optional double value relative between 0 and 2 * Pi
original optional point in 3D space, defined as set of 3 doubles

The texture coordinates are depending on the type depending on offsetX/offsetY
or origin depending on:
  type is an optional integer value defining a set of different ways how to
  map textures on different 2D faces and 3D solids.

8.2 GeometricItem

GeometricItem can be seen as an abstract class that is the (indirect) parent any class that has an
Geometric Representation. Be aware that not every GeometricItem has to have a representation in 3D,
also a representation can result in an empty visualization.
Each GeometricItem can refer (has an optional relation) too a Material.

8.2.1 BooleanOperation

A BooleanOperation class has two relations:
  firstObject (optional GeometricItem)
  secondObject (optional GeometricItem)

The result is only non trivial in case both objects
are in the same domain Dim_0, Dim_1, Dim_2 or Dim3. Be aware that an instance of BooleanOperation
becomes an instance of BooleanOperation3D when both firstObject and secondObject are of type
Dim_3. Similar BooleanOperation2D when both firstObject and secondObject are of type Dim_2, etc.

We know three different types of Boolean Operations:
  • Union
  • Difference and Inverse Difference
  • Intersection

The type of Boolean Operation is represented in the integer value type, resp, value 0, 1, 2 and 3.

8.2.2 Collection

A collection consists of an optional set of unlimited amount of other GeometryItems. This relation is
called objects. Be aware that an instance of Collection becomes an instance of Collection3D when all
instances of objects are of type Dim_3. Similar Collection2D when all instances are of type Dim_2, etc.

8.2.3 Dim_0

Dim_0 can be seen as an abstract class that is the parent of any class that describes a point. Dim_0
explicitly does not overlap with Dim_1, Dim_2 and Dim_3.

8.2.4 Point3D

An instance of Point3D describes a point in 3D space, the point is described
as a set of 3 double values. The property point that always contains exactly
3 double values describes the x, y, z coordinates.

8.2.5 Dim_1

Dim_1 can be seen as an abstract class that is the parent of any class that describes a line. Dim_1 explicitly does not overlap with Dim_0, Dim_2 and Dim_3.

8.2.6 Ellipse2D

An Ellipse is a line following an ellipse defined by two radius values.
The radius of the line transforms linear from radiusI in x-direction (rotation is 0 or Pi) towards radiusII in y-direction. The start value defines the starting position of the arc and the size defines the direction and size of the arc.

8.2.7 Arc2D

The Arc2D is actually an Ellipse 2D where radiusI and radiusII are the same and defined as radius. Be aware that Arc2D cannot be a subclass of Ellipse2D as radiusI and radiusII are non optional in Ellipse2D. However technically the code is the same and therefore Arc2D is redundant and could also have been defined in a separate ontology.

8.2.8 Line3D

The Line3D is a line explicitly defined in the 3D space. The property points consist always from 6 double values defining the starting and ending point of the line.

8.2.9 Polygon2D

A Polygon2D represents a polygon defined in 2D based on Arcs and Lines. The relation lineParts has at least 1 element of type Dim_1. In case the line is defined in 3D the third coordinate will be ignored, this is especially relevant when skewing operations cannot easily be applied via transformation matrices.

8.2.10 Polygon3D

Similar to Polygon2D Polygon3D exists of a collection of at least 1 lineParts referring to elements of type Dim_1. The difference is that when the third dimension is known in the referred object it is used and when not existing it is defined as 0. Note a Polygon3D with 1 lineParts relation on Polygon2D with lineParts XYZ has the same result (but in 3D space) as a Polygon2D with 1 lineParts on Polygon3D with lineParts XYZ (but then in 2D space).

8.2.11 PolyLine3D

The polyline3D consist or a set of at least 2 Point3D elements. Together the points form a line. The existence of this alternative way of representing
a 2D polygon can be important in cases where the definition of a solid
cannot be defined trivially in a BoundaryRepresentation based on vertex and index arrays. A
BoundaryRepresentation defined by Faces needs to know the link between points used in different faces,
this is of importance for several algorithms used for geometric operations, for Euler checks and therefore
also for data representation in (open) standards.

8.2.12 Dim_2
Dim_2 can be seen as an abstract class that is the parent of any class that describes a face. Dim_2
explicitly does not overlap with Dim_0, Dim_1 and Dim_3.

8.2.13 Face2D
A Face2D is a face defined in 2D space, defined by:
- outerPolygon (a non optional Dim_1)
- innerPolygons (an optional set of Dim_1)
When the outerPolygon is defined anti clock wise the normal is pointing upwards and only the top is
visible, this is also called the RHS (Right Hand Side) rule. InnerPolygons should be defined completely
within the outerPolygon and the direction should be opposite. For all referenced polygons counts that In
case the polygons are not closed an extra line will be added to close the polygon.

8.2.14 Dim_3
Dim_3 can be seen as an abstract class that is the parent of any class that describes a closed shell. The
internal area of a closed shell is interpreted as solid. Be aware that a solid does not have to be with
limited volume. For example the volume of a box defined inside out is unlimited. Dim_3 explicitly does
not overlap with Dim_0, Dim_1 and Dim_2.

8.2.15 Blend
A Blend is defined on top of two non optional closed polygons,
the polygons are referenced by:
- topPolygon (non optional Dim_1)
- bottomPolygon (non optional Dim_1)
In case the polygons are not closed an extra line will be added to close the polygon, note that it is also
possible to just add a circle or ellipse part via Ellipse2D/Arc2D.
The volume will be created to link the top and bottom polygon as good as possible, note that each
application could apply blend slightly different and the result is therefore not always uniquely defined by
the ontology in case blend is used.

8.2.16 BoundaryRepresentation
A Boundary Representation is typically used to define geometry coming from (other)
systems that lost its original description how it was defined. Note it can also be
used to store derived results from an ontology to prevent reapplying Boolean
operations that can be time consuming at the cost of loosing original definition and memory sparse representations. There are two ways to define a Boundary Representation:

- using a set of faces that together form a solid, for performance reasons it is advised to not duplicate original points in different faces whenever possible
- using a vertex and index buffer array. Within the index buffer polygons are terminated with -1 and openings are terminated with -2.

8.2.17 Box
A Box is a representation of one of the most basic forms with 6 faces, 8 points and 12 lines defined by the following properties:

- length is a non optional double value
- width is a non optional double value
- height is a non optional double value

In case one or more of the above values is zero the representation is empty. In case one or more of the above values is negative the normals of all faces will still point outside.

8.2.18 Cube
A Cube is a Box where the length is the same as the width and height. Note that this class actually should be in a library as it is a simple subclass from Box where the rules length = width and length = height apply.

8.2.19 Clipping
Clipping is like performing a Boolean Operation difference on an object with a Box of unlimited size on one side of the plane. Normally Clipping is much cheaper in terms of performance then a real Boolean Operation.

8.2.20 Cone
A Cone is defined by:

- radius is a non optional double value
- height is a non optional double value
- segmentationParts is a non optional integer value

The top of the Cone is placed on the orthogonal line through the centre of the circle with distance height between the circle and the top of the Cone. The segmentationParts defines the number of lineparts the circle is defined by.
8.2.21 Cylinder
A Cylinder is defined by:
- radius is a non optional double value
- height is a non optional double value
- segmentationParts is a non optional integer value
The segmentationParts defines the number of lineparts the circles are defined by.

8.2.22 ExtrudedPolygon
An Extruded Polygon is a very simple extrusion of a given polygon defined by a set of 2D points (double property points with at least 6 elements). Non optional double property extrusionLength defines the direction and gives the length of the extrusion.

8.2.23 ExtrudedAreaSolid(Set)
A ExtrudedAreaSolid(Set) defines an extrusion based on a polygon defined in Dim_2. It is allowed to define several opening polygons that have to be non-intersecting and completely inside the outer polygon. The extrusion direction can be defined and does not have to be orthogonal on the XY plane.

8.2.24 Prism
A Prism is defined by:
- radius is a non optional double value
- height is a non optional double value
- segmentationParts is a non optional integer value

8.2.25 Pyramid
A Pyramid is defined by:
- radius is a non optional double value
- height is a non optional double value
- segmentationParts is a non optional integer value

8.2.26 Sphere
A Sphere is defined by:
- radius is a non optional double value
- height is a non optional double value
- segmentationParts is a non optional integer value
8.2.27 SweptAreaSolid(Set)
A SweptAreaSolid(Set) is similar to ExtrudedAreaSolid(Set). However where the extrusion direction in ExtrudedAreaSolid could be set by a vector within SweptAreaSolid it is possible to use any Dim_2 polygon. Another difference is that the XY plane is rotating to be defined as orthogonal to the path defined.

8.2.28 Dummy
A dummy object can reference any Geometric Item via the optional relation object. It does not do anything and is purely a placeholder that can be relevant for certain modelling purposes. Be aware that an instance of Dummy becomes an instance of Dummy3D when object is of type Dim_3. Similar Dummy when object is of type Dim_2, etc.

8.2.29 Plane
A Plane is defined by:
- \(a\) is a non optional double value
- \(b\) is a non optional double value
- \(c\) is a non optional double value
- \(d\) is a non optional double value

The definition follows that for each point in the plane the following equation is true:
\[a \times x + b \times y + c \times z + d = 0\]
This can also be understood as that the plane has a normal vector \(\text{Normalize}(a, b, c)\) where the smallest distance from the plane towards the origin is \(d \times \text{length}(\text{Vector}(a, b, c))\).

8.2.30 Repetition
Often an object is placed in a set next to each other or as a circular repetition in a circle structure. To prevent the user from placing the same object several times in a Collection, not said that in such case the amount needs to be known, the Repetition concept is developed. A Repetition contains the following relations:
- matrix (non optional Matrix defining the transformation between each instance)
- object (non optional GeometryItem)

The same object will be placed \(n\) times with relative placement defined by matrix. The exact number of times is defined by property count being a non optional integer value.

8.2.31 Transformation
As objects rarely placed in the origin in the complete picture it is needed to be able to set a transformation of an arbitrary object. As these transformations are normally relatively to other objects and such transformations have different abstraction levels (i.e. a Window is placed within a Wall, relative to the Building Storey, relative to the Building, relative to the Site etc.).
The Transformation therefore has two relations:
object (non optional GeometryItem)
matrix (non optional Matrix)

The matrix itself can be a (nested) set of matrix multiplications, this enables reuse of more abstract transformations while still being able to set the transformation of a single item. Be aware that an instance of Translation becomes an instance of Translation3D when object is of type Dim_3. Similar Transformation2D when object is of type Dim_2, etc.

8.3 Matrix

A matrix is defined by:

_11 is an optional double value (if not defined it is expected to be 1)
_12 is an optional double value (if not defined it is expected to be 0)
_13 is an optional double value (if not defined it is expected to be 0)
_21 is an optional double value (if not defined it is expected to be 0)
_22 is an optional double value (if not defined it is expected to be 1)
_23 is an optional double value (if not defined it is expected to be 0)
_31 is an optional double value (if not defined it is expected to be 0)
_32 is an optional double value (if not defined it is expected to be 0)
_33 is an optional double value (if not defined it is expected to be 1)
_41 is an optional double value (if not defined it is expected to be 0)
_42 is an optional double value (if not defined it is expected to be 0)
_43 is an optional double value (if not defined it is expected to be 0)

8.3.1 InverseMatrix

An inverse matrix references exactly one matrix via relation matrix and applies the following operations:

```java
double a = orMat._11 * orMat._22,
b = orMat._12 * orMat._23,
c = orMat._13 * orMat._21,
d = orMat._22 * orMat._31,
e = orMat._21 * orMat._33,
f = orMat._23 * orMat._32,
determinant = a * orMat._33 + b * orMat._31 +
c * orMat._32 - d * orMat._13 -
e * orMat._12 - f * orMat._11;

if (determinant) {
    iMat._11 = (orMat._22 * orMat._33 - f) / determinant;
    iMat._12 = (orMat._13 * orMat._32 - orMat._12 * orMat._33) / determinant;
    iMat._13 = (b - orMat._13 * orMat._22) / determinant;
    iMat._21 = (orMat._23 * orMat._31 - e) / determinant;
    iMat._22 = (orMat._11 * orMat._33 - orMat._13 * orMat._31) / determinant;
    iMat._23 = (c - orMat._11 * orMat._23) / determinant;
    iMat._31 = (orMat._21 * orMat._32 - d) / determinant;
    iMat._32 = (orMat._12 * orMat._31 - orMat._11 * orMat._32) / determinant;
    iMat._33 = (a - orMat._12 * orMat._21) / determinant;
    iMat._41 = 0;
    iMat._42 = 0;
    iMat._43 = 0;
}
```
VECTOR3 tmp;
tmp->x = -orMat->.41;
tmp->y = -orMat->.42;
tmp->z = -orMat->.43;
Vector3Transform(&tMat._41, &tmp, iMat);
}

Where:

- orMat is the original matrix
- iMat is the inverse matrix

The result is that in case the determinant of a matrix is non zero the multiplication of a matrix with its inverse as well as the multiplication of the inverse with its original results in the identity matrix. It is an often used variant of the matrix and very relevant, however it could also have been part of an external library as CMO is able to represent InverseMatrix with its behaviour as user defined subclass of Matrix.

8.3.2 MatrixMultiplication

A matrix multiplication applies the following operation on two input matrices referenced by non optional relations firstMatrix and secondMatrix:

\[
\begin{align*}
\mathbf{rM}_{.11} &= \mathbf{fM}_{.11} \times \mathbf{sM}_{.11} + \mathbf{fM}_{.12} \times \mathbf{sM}_{.21} + \mathbf{fM}_{.13} \times \mathbf{sM}_{.31}; \\
\mathbf{rM}_{.12} &= \mathbf{fM}_{.11} \times \mathbf{sM}_{.12} + \mathbf{fM}_{.12} \times \mathbf{sM}_{.22} + \mathbf{fM}_{.13} \times \mathbf{sM}_{.32}; \\
\mathbf{rM}_{.13} &= \mathbf{fM}_{.11} \times \mathbf{sM}_{.13} + \mathbf{fM}_{.12} \times \mathbf{sM}_{.23} + \mathbf{fM}_{.13} \times \mathbf{sM}_{.33}; \\
\mathbf{rM}_{.21} &= \mathbf{fM}_{.21} \times \mathbf{sM}_{.11} + \mathbf{fM}_{.22} \times \mathbf{sM}_{.21} + \mathbf{fM}_{.23} \times \mathbf{sM}_{.31}; \\
\mathbf{rM}_{.22} &= \mathbf{fM}_{.21} \times \mathbf{sM}_{.12} + \mathbf{fM}_{.22} \times \mathbf{sM}_{.22} + \mathbf{fM}_{.23} \times \mathbf{sM}_{.32}; \\
\mathbf{rM}_{.23} &= \mathbf{fM}_{.21} \times \mathbf{sM}_{.13} + \mathbf{fM}_{.22} \times \mathbf{sM}_{.23} + \mathbf{fM}_{.23} \times \mathbf{sM}_{.33}; \\
\mathbf{rM}_{.31} &= \mathbf{fM}_{.31} \times \mathbf{sM}_{.11} + \mathbf{fM}_{.32} \times \mathbf{sM}_{.21} + \mathbf{fM}_{.33} \times \mathbf{sM}_{.31}; \\
\mathbf{rM}_{.32} &= \mathbf{fM}_{.31} \times \mathbf{sM}_{.12} + \mathbf{fM}_{.32} \times \mathbf{sM}_{.22} + \mathbf{fM}_{.33} \times \mathbf{sM}_{.32}; \\
\mathbf{rM}_{.33} &= \mathbf{fM}_{.31} \times \mathbf{sM}_{.13} + \mathbf{fM}_{.32} \times \mathbf{sM}_{.23} + \mathbf{fM}_{.33} \times \mathbf{sM}_{.33}; \\
\mathbf{rM}_{.41} &= \mathbf{fM}_{.41} \times \mathbf{sM}_{.11} + \mathbf{fM}_{.42} \times \mathbf{sM}_{.21} + \mathbf{fM}_{.43} \times \mathbf{sM}_{.31} + \mathbf{sM}_{.41}; \\
\mathbf{rM}_{.42} &= \mathbf{fM}_{.41} \times \mathbf{sM}_{.12} + \mathbf{fM}_{.42} \times \mathbf{sM}_{.22} + \mathbf{fM}_{.43} \times \mathbf{sM}_{.32} + \mathbf{sM}_{.42}; \\
\mathbf{rM}_{.43} &= \mathbf{fM}_{.41} \times \mathbf{sM}_{.13} + \mathbf{fM}_{.42} \times \mathbf{sM}_{.23} + \mathbf{fM}_{.43} \times \mathbf{sM}_{.33} + \mathbf{sM}_{.43};
\end{align*}
\]

Where:

- fM is the first matrix
- sM is the second matrix
- rMat is the resulting multiplied matrix

It is an often used variant of the matrix and very relevant, however it could also have been part of an external library as CMO is able to represent MatrixMultiplication with its behaviour as user defined subclass of Matrix.
9. Appendix II - Hardware and Software

9.1 Hardware

The hardware used is divided in main development hardware and testing hardware. To do the main development and testing the following systems are used:

- **DELL Precision type M6500**
  - Intel i7 quad core processor
  - 16 Mb internal memory
  - nVidia QUADRO FX 3800M videocard
  - Windows 7 Professional

- **DELL Precision type M6600**
  - Intel i7 quad core processor
  - 16 Mb internal memory
  - nVidia QUADRO 5010M videocard
  - Windows 7 Professional

The hardware system to apply testing:

- **SONY Vaio Pro 13**
  - Intel i5 dual core processor
  - 8 Mb internal memory
  - Integrated Intel HD graphics (4000 series)
  - Window 8.1

During development of large models we experienced performance issues and were hitting the boundaries of non high-end video cards.
9.2 Software

During developments we used the following software
- Windows 7 Professional (64 bit) and Windows 8.1 (64 bit) from Microsoft
- Visual Studio 2010 Professional and Visual Studio 2013 Professional from Microsoft
- TopBraid Composer Meastro Edition and Free Edition from Top Quadrant
- Protégé from Stanford University
- VM VirtualBox from ORACLE
- Linux Ubuntu 12.04 LTS (64 bit)
- GCC 4.7/4.8 compiler
- Eclipse Standard 4.2/4.3 for Linux

The following already software is embedded in the products
- IFC Engine DLL library
- CSG Engine library
- dotNetRDF library
- DirectX 9.0c libraries
- WebGL libraries (only on the client)

For the embedded products means that they are ‘open’, well known and free (like DirectX and WebGL) or the source code is available for free or at reasonably price and the interfaces are clearly defined.

9.2.1 IFC Engine DLL

This product is used to be able to apply conversions between IFC and CMO. It includes a large part of internal support for geometry conversion between the standards and enables conversion for both IFC 2x3 TC1 and IFC4.

9.2.2 CSG Engine

The CSG engine is able to store interlinked RDF Class and Property structures. It also is able to apply all kind of geometry operations towards an internal geometry format that is very close to CMO with Extensions. All any geometrical visualization send to the screen is generated by this component.

For stand alone applications the 32 bit Windows version is used, it is expected this will be ported to the 64 bit version later in the project. For WebGL client/server solution the 64 bit Linux version is used.

9.2.3 doNetRDF library

One of the view non-JAVA parsers for RDF based content. The use of this library will probably be temporary as the library cannot be used under Linux and OSX. A good alternative seems to be the Redland RDF library.