

AIM@SHAPE

Advanced and Innovative Models And Tools for the
development of Semantic-based systems for
Handling, Acquiring, and Processing knowledge
Embedded in multidimensional digital objects

IST NoE No 506766

Deliverable D1.5.1



Report on an integrated view of the domain ontologies – 1st version

Circulation:	¹ CO
Partner(s):	ITI, IMATI, DISI, EPFL, IGD, MPII, INRIA
Authors:	G. Vasilakis, M. Pitikakis, Riccardo Albertoni, Chiara Catalano, Simone Marini
Version:	03
Stage:	100%
Date:	Thursday, March 2, 2006

¹ Please indicate the dissemination level using one of the following codes:

PU = Public

PP = Restricted to other programme participants (including the Commission Services).

RE = Restricted to a group specified by the consortium (including the Commission Services).

CO = Confidential, only for members of the consortium (including the Commission Services).

Copyright

© Copyright 2006 The AIM@SHAPE Consortium

consisting of:

CNR-IMATI-GE	C.N.R. – Istituto di Matematica Applicata e Tecnologie Informatiche Dept. of Genova, Italy
DISI	Università di Genova – Dipartimento di Informatica e Scienze dell'Informazione, Italy
EPFL	École Polytechnique Federale de Lausanne, Switzerland
FhG/IGD	Fraunhofer Institut für Graphische Datenverarbeitung, Germany
INPG	Institut National Polytechnique de Grenoble, France
INRIA	Institut National de Recherche en Informatique et Automatique, France
ITI-CERTH	Informatics and Telematics Institut – Center for Research and Technology Hellas, Greece
UNIGE	Université de Genève, Switzerland
MPII	Max-Planck-Institut für Informatik, Germany
SINTEF	Stiftelsen for industriell og teknisk forskning ved Norges Tekniske Høgskole, Norway
TECHNION	Technion – Israel Institute of Technology, Israel
UU	Utrecht University, Netherlands
WEIZMANN	Weizmann Institute of Science, Israel

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the AIM@SHAPE Consortium. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.

This document may change without notice.

Document History

Vers.	Issue Date	Stage	Content and changes
1	22 November 2005	20%	First draft
2	18 January 2006	80%	Second draft
3	30 January 2006	100%	Final version

Executive Summary

This document contains a description of the deliverable D1.5.1 of the IST NoE AIM@SHAPE. The deliverable *D1.5.1 Report on an integrated view of the domain ontologies – 1st version* is intended to provide a first version of an integrated view of the ontology development process in the network.

The task leader is ITI-CERTH and is actively supported by all other AIM@SHAPE partners.

The document is structured as follows. Section 1 gives a short introduction on ontology development with OWL. Section 2 describes the development of domain ontologies in the network while section 3 describes the motivation behind the development of the common ontologies in the network. Section 4 gives some concluding remarks and section 5 provides some reference information.

Table of Contents

1	INTRODUCTION.....	5
1.1	INTRODUCTION TO OWL.....	5
2	ONTOLOGY DEVELOPMENT IN THE NETWORK.....	7
2.1	DOMAIN ONTOLOGIES.....	8
3	COMMON ONTOLOGIES	10
3.1	THE COMMON ONTOLOGY FOR SHAPES	11
3.2	THE COMMON ONTOLOGY FOR TOOLS	13
4	CONCLUDING REMARKS	17
4.1	SUMMARY OF CHANGES FROM THE PREVIOUS VERSION.....	17
5	REFERENCES.....	19

1 INTRODUCTION

An ontology necessarily entails or embodies some sort of world view with respect to a given domain. The world view is often conceived as a set of concepts (e.g. entities, attributes, and processes), their definitions and inter-relationships; this is referred to as a *conceptualisation*. Recording such a conceptualisation with an ontology is referred to as ontology development and has been described in broad terms in deliverable D1.4 – *Shape ontologies: Requirements and preliminary design*. Furthermore, a methodology for developing an ontology has been outlined. The methodology divided the process into four phases: Kick-off, Refinement, Evaluation and Maintenance & Evolution. We will be referring to these phases in this document in order to identify the current status of the various ontologies that are being developed within the network.

Each phase can also be distinguished by the degree of formality by which the vocabulary, that constitutes the ontology, is created and meaning is specified. Four, somewhat arbitrary degrees of formality can be identified:

- Highly informal: the vocabulary is expressed loosely in natural language;
- Semi-informal: the vocabulary is expressed in a restricted and structured form of natural language, greatly increasing clarity by reducing ambiguity;
- Semi-formal: the vocabulary is expressed in an artificial formally defined language;
- Rigorously formal: the vocabulary is meticulously defined with formal semantics, theorems and proofs of such properties as soundness and completeness.

1.1 Introduction to OWL

An important step in the ontology development in the network has been the adoption of the OWL language [1] as the ontology representation language. The reasons for choosing OWL are the same reasons which lead to the creation of OWL: *the identification of a number of characteristic use-cases for the Semantic Web which require much more expressiveness than that offered by older ontology languages*. OWL provides some new features that distinguish it from older ontology languages and facilitate reasoning both at schema and instance level. Some notable examples will be given in the following. Additionally, more specific information on developing ontologies with OWL, most of which also applies for other ontology languages, is given next.

In ontology development *classes* are created to encapsulate the meaning of concepts in a specific target domain. A class describes all those features that define the concept in the knowledge domain. An *individual*, or an *instance*, of that class in the ontology represents an object whose type is characterized by the concept in the domain. For example the class *Animation Sequence* in the Virtual Humans ontology describes the concept of an animation sequence that is used to show actions of virtual humans. An instance of the *Animation Sequence* class in this case would be a specific animation sequence, with all the necessary metadata information (i.e. format, description, associated files, etc.), used for example to visualize a virtual human in the action of running.

Relations between classes are defined by *object properties*. Object properties describe the kind of associations that are possible between classes. On the other hand, *datatype properties* describe features of the class and represent attributes that have a specific type such as *boolean*, *integer*, *string*, etc. Examples of datatype properties are *phone*, *title*, *age*, etc.

Restrictions can be imposed both on classes and on properties in order to refine their semantics. Suppose for example that we wish to declare, that a class C satisfies certain conditions, that is, all instances of C satisfy the necessary conditions. Using an OWL *Restriction* in an ontology describing a University, for example, we can easily describe the axiom that says that *all academic staff members must teach at least one undergraduate course*. This imposes a restriction on the class *Academic Staff*, thus making it more specific. In this example we state that every professor must teach an undergraduate course. In terms of logic we have a *universal quantification*.

Properties can also be defined to have special characteristics. For example, sometimes it is useful to say that a property is *transitive* (like “greater than”), *unique* (like “is mother of”), *symmetric* (like “is sibling of”), or the *inverse of* another property (like “eats” and “is eaten by”).

In OWL it is possible to talk about boolean combinations (*union*, *intersection*, *complement*) of classes. In the University example, we can say that courses and staff members are *disjoint* and we can find all the people at the University by finding the union of the *Staff Member* and the *Student* classes.

These are some of the features that make OWL a good choice for specifying ontologies. Its advanced semantics make automated reasoning support possible and also provide the necessary expressiveness needed both in specifying complex schema restrictions and in making interesting queries.

2 ONTOLOGY DEVELOPMENT IN THE NETWORK

In deliverable D1.4 – *Shape ontologies: Requirements and preliminary design*, four clusters were identified, representing different areas in the domain of Computer Graphics and motivating the development of four domain ontologies in the respective research areas: Virtual humans, Beyond STEP, Design, Acquisition and Reconstruction. During the ontology development process it became apparent that there was not enough motivation for the development of the Beyond STEP ontology. As a result it was decided that the development of this ontology will not proceed any further.

Ontology development in the network has so far been focused on domain ontologies for the various clusters that have been defined. However, needs for defining metadata for tools and shapes have also been expressed which have led to the creation of the metadata for Tools and Shapes respectively. This metadata have so far been stored in the Tools and Shapes repositories, respectively, but they also represent concepts that are shared by the domain ontologies. The next logical step was therefore the transformation of this metadata from their current format to an ontology driven format that is compatible with that of the domain ontologies. The goal is to create higher level ontologies which can be extended by the domain ontologies to express metadata for each cluster. These will constitute common ontologies to be shared and reused by each cluster. A first version of such a common ontology was presented in deliverable D1.4.

In the case of the common ontologies we did not try to immediately define ontology structures for the associated metadata (for tools and shapes) as we had to follow a *bottom-up* approach. The necessity for this approach came from the existence of metadata for tools and shapes which had to be taken into account. So the preliminary version was based on a simple structure, mostly following the Dublin Core metadata format [2], in order to capture the existing metadata.

Furthermore, in all the clusters we identified associations to metadata involving shapes and tools. Therefore, we also had to respect the needs of all clusters in creating the ontologies for shapes and tools because a large part of these ontologies was expected to be common and shared by the domain ontologies. This does not mean that all domain ontologies are expected to use this common metadata in the same way. The opposite is true: the common ontologies are expected to define a basic structure for this metadata and each domain ontology is expected to specialise and use the metadata information according to its needs. For example, this might involve extending the common structure or associating it in an appropriate way with domain-specific metadata. Keeping the common ontologies structure simple is a vital prerequisite for ensuring their reusability.

To ensure that the structure for the metadata for shapes and tools respects the needs of all clusters and so it is indeed common, a second phase took place that involved the validation of the metadata structure for shapes and tools. In this validation process all partners involved in the domain ontologies took part, as well as partners who are experts in the relevant research field and have the necessary expertise. The result of the validation phase was a metadata structure that is common and shared by all domain ontologies but was still not ontology-driven. The next step involved the development of

the common ontologies for shapes and tools that were based on the common metadata structure and are presented in section 3 of this deliverable.

2.1 Domain Ontologies

With respect to the ontology development methodology that is chosen, as described in section 1, the current status of the domain ontologies is the Evaluation phase. Due to the iterative nature of the methodology (see deliverable D1.4), there is a continuous shifting between the Evaluation phase and the Refinement phase. This simply means that the ontologies undergo refinements followed by evaluation phases in order to expand and capture more domain knowledge to be able to answer more CQs. In essence all the ontologies have passed the Kick-off phase which is the initial phase where the ontology structure takes some form. The degree of formality in all ontologies is *rigorously formal*.

However, identifying the development stage each ontology is currently in only reveals part of the status information that is needed. What is also needed is a qualitative evaluation of each ontology. Such an evaluation is necessary and constitutes a technical judgment of the ontologies and their associated environment with respect to a frame of reference. This frame of reference consists of the requirements specifications, the competency questions, expert knowledge of the domain, etc., for each of the ontologies being developed.

Up to this point ontology development has been more or less intuitive as a basic structure had to be established. This meant that a rigorous evaluation could not be performed. Despite this fact, such a qualitative evaluation has taken place, at least partly, and the various ontologies have been tested to answer some of the competency questions initially defined, and progress in describing some of the domain-specific scenarios has been shown in the respective deliverables.

However, a more rigorous evaluation procedure needs to be imposed at the ontology development phases that follow. For this purpose the notion of an *evolution step* with regards to ontology development will be introduced. An evolution step signifies a well-defined progress of the ontologies which can be qualitatively justified by taking into account an evaluation frame of reference. Each ontology must be shown to take a concrete step towards completeness by capturing a wider spectrum of CQs and adhering to the requirements specifications that have been specified. Evolution steps will have a constant duration of two months.

Measuring progress can only be done with respect to competency questions and use-case scenarios. Both CQs and scenarios constitute requirements on the expressiveness and the amount of knowledge that needs to be captured by the ontologies. Therefore, the evolution of domain ontologies must necessarily be justified through the presentation of new CQs that can be answered by each ontology and the modification of its structure in order to capture the new knowledge that is required. Obviously, the ontology should also be able to answer the CQs it already answered.

Competency questions have so far been specified in an informal way in natural language. To be able to use them as a qualitative measure of the ontology development process, however, we need them to be logically consistent and unambiguous. This implies that a

formal way of describing CQs is necessary. Indeed, formal competency questions, as was also described in D1.4, constitute a required step towards the definition of formal axioms (axioms, rules and restrictions in OWL), which facilitate the validation of the ontology.

3 COMMON ONTOLOGIES

A common ontology, also known as meta-ontology or higher-level ontology, is an ontology that is created to integrate ontologies across domains in order to support some common purpose. In this sense a common ontology has the potential to achieve interoperability across domains and encapsulate the fragmented knowledge, captured by the different ontologies, under a unified ontological framework. In this case, the common goal is the ability to reason, to re-use existing knowledge and to extend and create new knowledge about shape resources and tools.

As promising as this might seem, the overall integration of existing ontologies is one of the biggest problems we face in developing a comprehensive methodology for building common ontologies. It is easy enough to identify synonyms and to extend an ontology where no concepts readily exist. However, when there are obviously similar concepts defined in different contexts, it is rarely clear how and whether such concepts can be adapted and reused. The difficulty lies in achieving the necessary integration and also keep the structure simple in order to increase its potential for reuse.

This is the first version of the common ontologies (a very first draft version has already been presented in deliverable D1.4). Unlike the development process regarding the domain ontologies which is based on a combination of *top-down* and *middle-out* approach, the methodology used for the development of the common ontologies is a *bottom-up* approach. For a description of the different ontology development approaches see [3]. The common ontologies aim to capture the metadata information that is related to shapes and tools and is specified in deliverables D1.5 and D1.6.

In essence, we tried to integrate all the metadata information from the *Shape Repository*, and the *Tools Repository*. Our intent is to provide an ontology-driven specification for objects representing a shape (*Shapes Repository*) or a tool (*Tools Repository*), including other secondary information (like creation info, tools info etc.). There is some metadata information which can be shared in all domain ontologies because all deal with this kind of information (e.g. geometrical information). Each domain ontology could build on this common structure and extend it or focus on a specific part of it and expand it (i.e. add classes, properties, relationships). In a specific domain, a shape is expected to have this common metadata information and potentially some domain-specific metadata information as well. As a result all metadata (common and domain-specific) will be ontology-driven, specified using the OWL language and stored in the *Ontology & Metadata Repository*.

Clearly, this is not the final version of the common ontologies as there are several elements that have not been integrated yet and some extensions to the already existing metadata that have not been finalized yet. Also a full integration with the domain ontologies has not been achieved yet. There are some concepts and properties from the *Cluster Ontologies* that are also captured in the common ontologies, albeit in a slightly different manner, so another step is needed in order to fully integrate the domain ontologies with the common ontologies. Furthermore, there is also the possibility to include in the common ontologies concepts related to the *Digital Library*, or even to fully integrate the DL with the common ontologies.

3.1 The common ontology for shapes

The common ontology for shapes captures shape related metadata and constitutes an ontology-driven evolution of the metadata in the Shapes repository. A high level view of the ontology is shown in Figure 1.

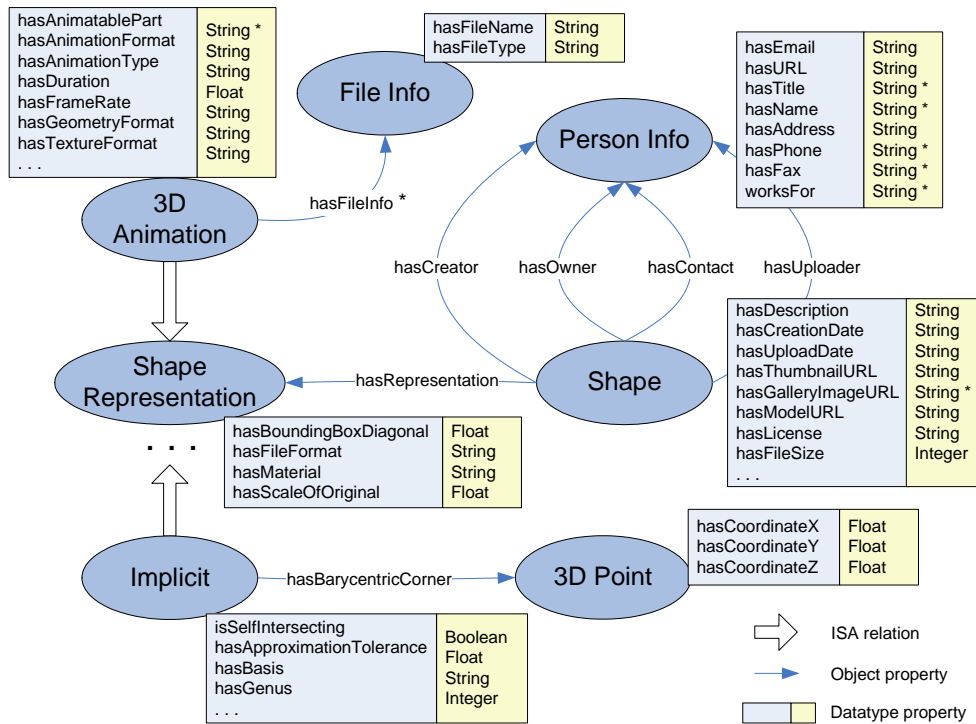


Figure 1. The common ontology for shapes.

The structure of the ontology as can be seen is fairly simple. The complexity of an ontology structure increases with the ontology depth (the number of class levels) and the number of relations between classes (object properties). In this case, the number of object properties is small. The ontology depth is not so small but this occurs only with respect to the *Shape Representation* class. This ensures that there are no relations between classes in the lower levels of the hierarchy – as the hierarchy involves only one level one class. The most important datatype properties for each class are shown in the diagram.

The *Shape Representation* hierarchy is shown in Figure 2. Different class levels are painted in different colour in the diagram. First level classes are shown in blue colour, second level classes are shown in light blue colour and third level classes are shown in yellow colour in the hierarchy. The attributes (datatype properties) of the first and third level classes in the hierarchy are shown in Figure 3. The datatype properties of the second level classes in the hierarchy are depicted in Figure 4.

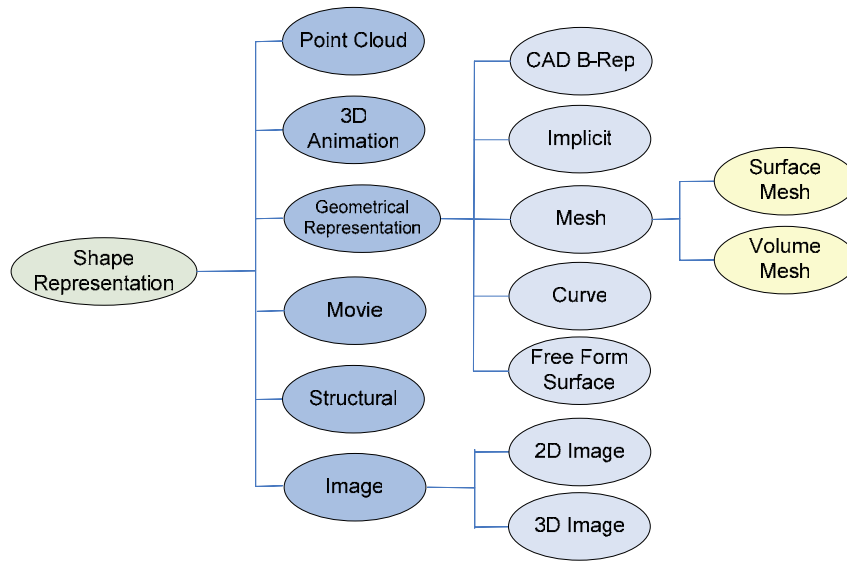


Figure 2. The Shape Representation hierarchy.

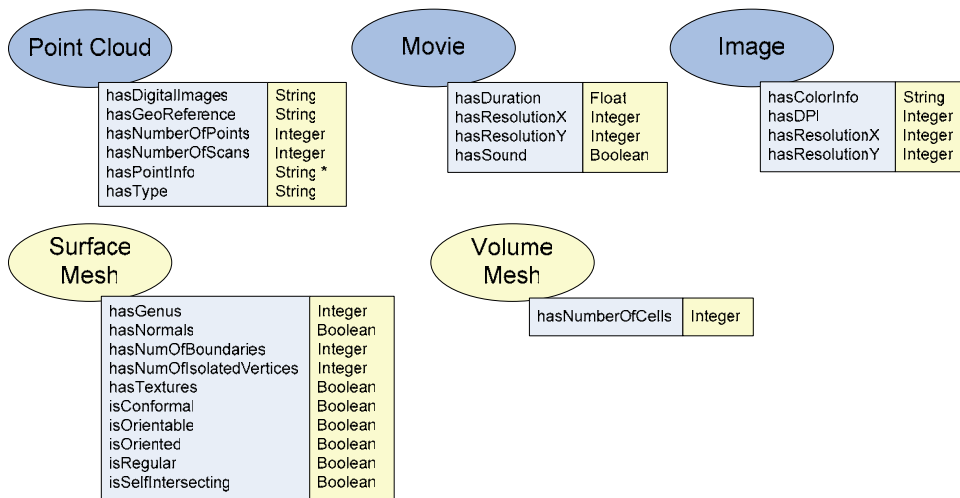


Figure 3. Attributes of the first- and third-level classes in the Shape Representation hierarchy.

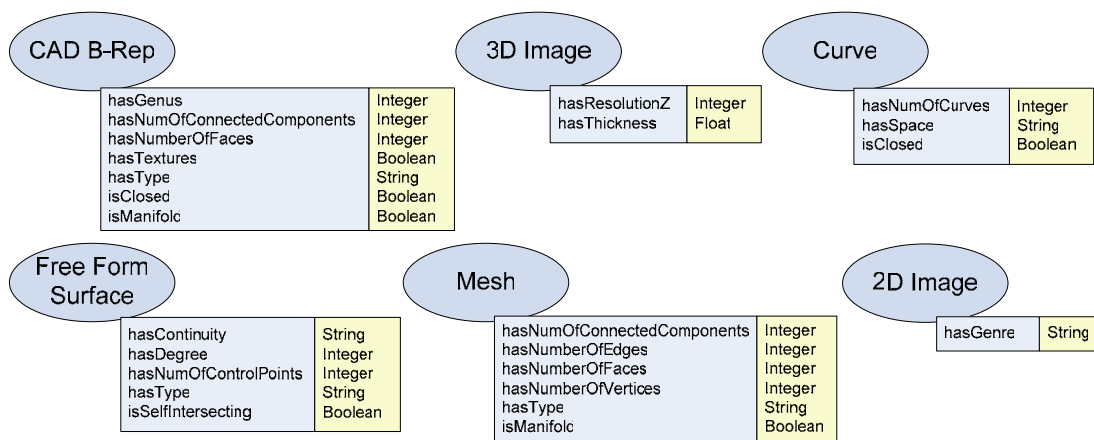


Figure 4. Attributes of the second-level classes in the Shape Representation hierarchy.

3.2 The common ontology for tools

The common ontology for tools captures tools related metadata and constitutes an ontology-driven evolution of the metadata in the Tools repository. A high level view of the ontology is shown in Figure 5.

The different color used for the *Person Info* class denotes that the description of this class is imported from the common ontology for shapes. This class is shared between the two ontologies and it makes no sense to specify it twice.

On a first look the tools ontology seems much more complex than the shapes ontology due to the existence of more concepts needed to describe the common vocabulary. However, all of the object properties, which define relations between concepts, exist in the first class level only. Furthermore, out of the 3 class hierarchies that exist in the ontology the 2 that are related to classes *Algorithm Category* and *Shape Type* simply capture type information for the respective concepts (algorithms and shapes).

This means that there are no additional attributes defined in these hierarchies and there is a specific amount of instances that will be created – those that represent the most specific types for each concept. For example, in the *Algorithm Category* hierarchy a leaf class is defined which is named *Intersection Algorithm*. The only allowed instances for this class are *Intersection Compute Algorithm*, which represents the algorithms computing the

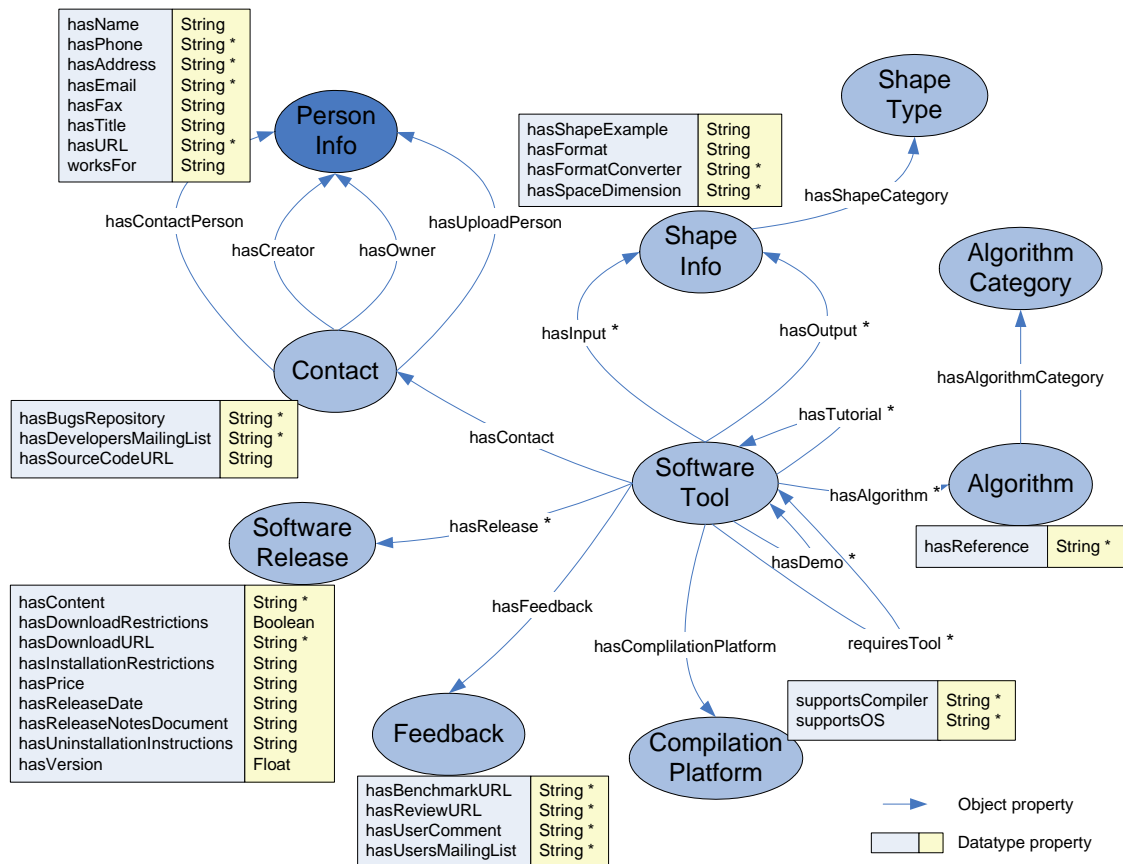


Figure 5. The common ontology for tools.

common intersection between the input objects, and *Intersection Detect Algorithm*, which represents the algorithms testing whether the input objects have a non empty common intersection. These two instances are predefined and no other instances may be defined for that class. This does not necessarily mean that this class cannot be extended when some potential subclasses may be found – in such a case the class will be sub-classed and new instances may be created by the ontology architect. What is meant here is that the *Intersection Algorithm* class is not meant to be instantiated by an ontology user. The two predefined instances are considered as part of the ontology schema and not metadata information. This might seem strange but in fact the instances have been defined to ensure that the ontology is logically consistent. For completeness, the following restriction should also have been defined for the class *Intersection Algorithm*:

$(\exists \text{Intersection Compute Algorithm}) \sqcup (\exists \text{Intersection Detect Algorithm}),$

which states that the only possible instances are the ones that have already been created.

The two hierarchies are shown in figures 6 and 7. In Figure 7 the predefined instances of the various classes are also shown. The instances are omitted from Figure 6 as there are too many to be displayed in the diagram.

It is evident from the diagram in Figure 7 that the *Shape Type* hierarchy shares a lot of concepts with the *Shape Representation* hierarchy in the common ontology for shapes.

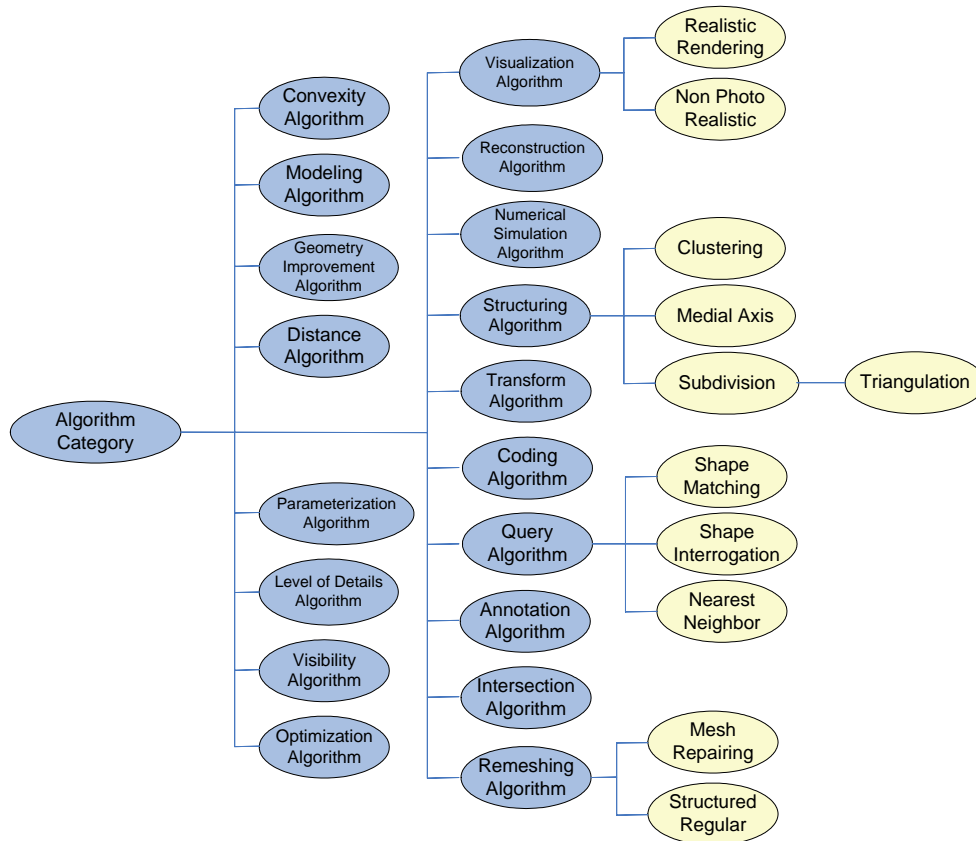


Figure 6. The *Algorithm Category* hierarchy.

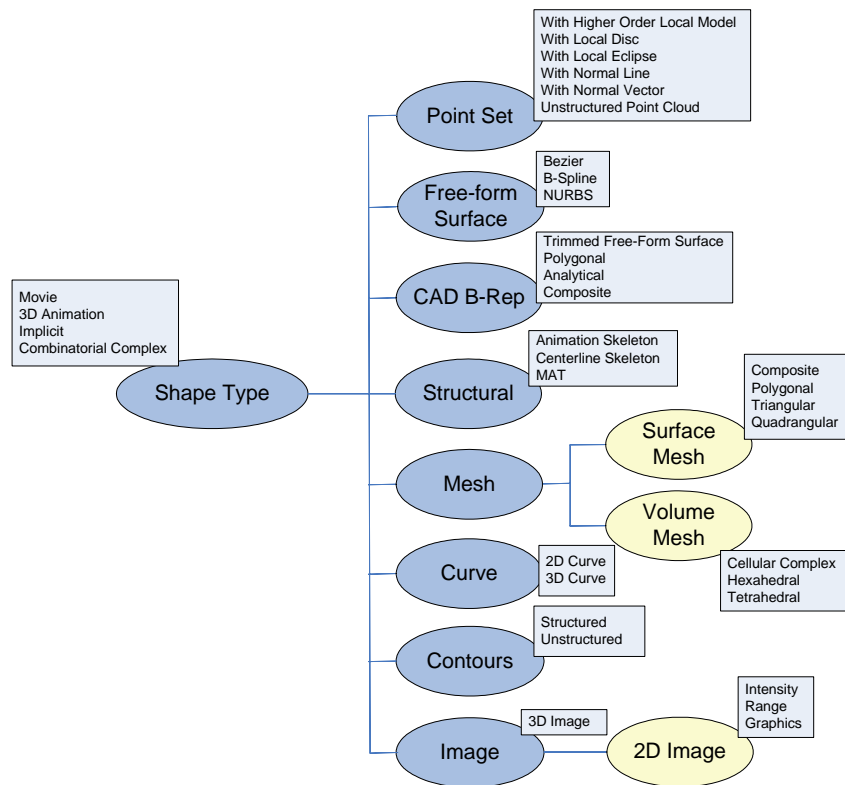


Figure 7. The *Shape Type* hierarchy.

The reason for specifying a large part of the hierarchy twice, and not importing it like the *Shape Info* class, is that the two hierarchies represent the same concepts under different contexts. In essence, the *Shape Type* hierarchy is in a higher level of generality. It represents a particular shape type or category, whereas the *Shape Representation* hierarchy represents the more specific shape objects that fall under the categories specified by *Shape Type*.

To better understand why this is necessary consider an instance of the *Software Tool* class. Furthermore, suppose we need to specify a shape type that constitutes an acceptable input parameter for the tool. In this case we simply need the type information about the shape, so we need to know for example that it represents a mesh. If we were to use the *Mesh* class of the *Shape Representation* hierarchy we would have to fill in much more information besides that given by the class (that it is of type *Mesh*). So we would have to specify values for attributes like *hasNumberOfEdges*, *hasNumberOfVertices*, *isManifold*, *hasNumberOfConnectedComponents*, etc. In doing so we would create a very specific *Mesh* instance, one that cannot be reused in any other case where we would need to specify that the type of a shape is a mesh. This is precisely why we cannot reuse the *Shape Representation* hierarchy to specify the type of a shape.

Of course, there is a way – possibly, more than one, actually – of circumventing the problems that arise and define just one hierarchy to serve both purposes. However, it is not a good design choice, it is not intuitive and it does not scale well; therefore, we have not opted for such a solution.

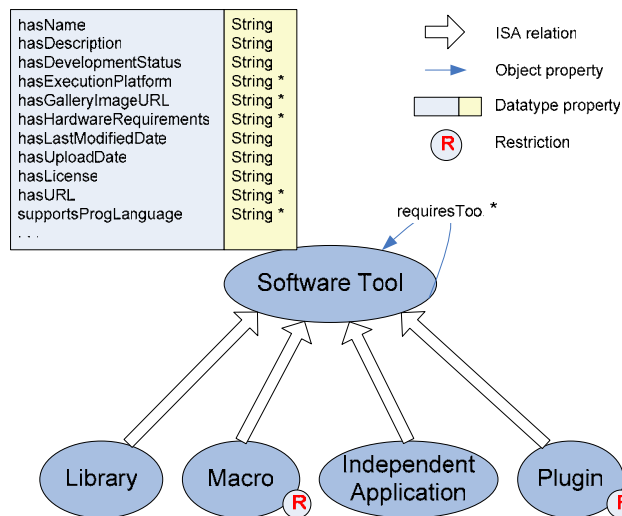


Figure 8. The *Software Tool* hierarchy.

The third hierarchy has the *Software Tool* class at its root and is shown in Figure 8. This is another simple hierarchy as no extra relations are specified by the four subclasses. There would be no need for sub-classing at all, however, there is one restriction specified in classes *Macro* and *Plugin* to differentiate them from the other two:

$$\exists \text{ requiresTool } \textit{Software Tool}$$

This restriction simply states that each *Macro* or *Plugin* instance should be associated with a *Software Tool* instance. In other words, a plugin or a macro only have meaning and can be applied with some reference to a tool.

4 CONCLUDING REMARKS

In this deliverable we mainly described the evolution of the metadata for shapes and tools that are currently stored in the respective repositories, from a semi-structured to an ontology-driven format. This has resulted in the creation of the first version of the common ontologies for shapes and tools respectively.

A metadata validation phase has preceded the creation of the common ontologies in order to harmonize the shapes and tools metadata according to the needs of the various clusters. However, a full integration of the common ontologies with the domain ontologies has yet to be achieved. This is expected to take place before the next version of the ontologies is released.

We have tried to keep the common ontologies as simple as possible in order to increase their potential for reusability. It is possible that future refinements may simplify this structure even more and move some concepts from the common ontologies to the domain ontologies. In general, such refactorings, even though they do not provide more functionality, do accomplish to keep information close to where it will be needed, and thus make the ontology structure more flexible.

Another potential modification, which seems to contradict the aforementioned refactoring pattern, involves moving the *Shape Type* hierarchy from the Tools common ontology to the Shapes common ontology. Despite the fact that this hierarchy is used in the tools ontology it makes sense to define it in the shapes ontology as the concept of the type of a shape is very close to the concept that the *Shape Representation* hierarchy represents. This will make the two different contexts described by the two hierarchies more clear and possibly reduce the confusion that might be created by having two hierarchies representing two very close concepts in different ontologies.

A general description of the status and the progress made in the development of the domain ontologies has been given. More specific information can be found in the associated deliverables. A methodology for the qualitative evaluation of the domain ontologies has been outlined and the need for the definition of formal competency questions has been stressed. The evaluation procedure is not going to be used for the common ontologies as their evolution is expected to be small – it is expected to correspond to a time span of three to six months.

4.1 Summary of changes from the previous version

The following modifications and improvements were made since the first version of this deliverable (D1.4):

- Ontology development for cluster *Beyond STEP* has been stopped.
- The structure for the metadata for Tools and Shapes has been expanded and the repositories have been populated with new metadata.
- The metadata for Tools and Shapes have been validated by all clusters.

- The first version of the common ontologies has been produced. These ontologies provide a common structure that captures the metadata information found in the Shapes and Tools repositories.
- The validation procedure for the common ontologies has started, the result of which will be the full integration with the domain ontologies. Furthermore, the common ontologies will be extended as needed to capture metadata information that has not been specified yet.
- The common ontologies have been improved to capture more domain knowledge. Progress on each of the domain ontologies can be found in the respective deliverables. The documents that constitute the latest version of these deliverables are: D1.2.1.1, D1.2.2.1 and D1.2.3.1.

5 REFERENCES

1. OWL Web Ontology Language. <http://www.w3.org/TR/owl-features/>
2. Dublin Core Metadata Initiative. <http://dublincore.org/>
3. Y. Sure and R. Studer: A Methodology for Ontology-based Knowledge Management. In: *On-To-Knowledge: Semantic Web enabled Knowledge Management*. J. Davies, D. Fensel, F. van Harmelen (eds.), ISBN: 0-470-84867-7, Wiley, 2002, pages 33-46.