

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 D1.2.3.1



Ontology for Product Design – 3rd version

Circulation:	¹PU
Partner(s):	IGD (Leader), IMATI, INPG, INRIA, ITI, SINTEF
Authors:	N. Sevilmis, C. Catalano, E. Camossi, V. Cheutet, T. Dokken, R. Ferrandes
Contributors:	V. Skytt, M. Pitikakis
Version:	03
Stage:	Final (100%)
Date:	Thursday, March 1, 2007

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 2007 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 Institute – 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

Version	Issue Date	Stage	Content and changes
01	31 January 2007	80%	3 rd version of the Product Design Ontology
02	28 February 2007	Completed	Finalization of the validation process

Executive Summary

This document contains the deliverable **D1.2.3.1** of the IST NoE AIM@SHAPE.

The deliverable ***D1.2.3.1 – Ontology for Product Design – 3rd Version*** – is intended to provide a third version of the Product Design Ontology, which is part of the ontology development process in the network.

The task leader is **IGD** and has been actively supported by all involved partners.

For a better understanding of this document it is recommended to have read the first and the second version of the Product Design Ontology (D1.2.3.1).

This document is structured as follows:

In the introduction we shortly summarize the evolution steps taken in the different versions of the Product Design Ontology. Section 2 describes new ontological elements that have been chosen and integrated in the ontology, yielding its evolution; in section 3 we present our contribution concerning the evolution of the common ontologies, while in section 4 we validate the structure of the evolved cluster ontology; section 5 summarizes our dissemination activities; finally, section 6 gives the outlook concerning the future work for our cluster.

Table of Contents

1	INTRODUCTION	7
2	ONTOLOGY EVOLUTION	8
2.1	Modelling different condition types	8
2.2	Formalizing the relations between <i>ShapeRole</i> and <i>ConditionType</i>	11
2.3	Quality check of CAD models	13
3	COMMON ONTOLOGIES	18
4	ONTOLOGY VALIDATION	19
5	DISSEMINATION ACTIVITIES FOR THE PDO	24
6	FUTURE ACTIVITIES	24

Table of Figures

Figure 1: Partial visualisation of the ConditionType taxonomy.....	8
Figure 2: New relations between ShapeRole, ConditionType, and GeometricConditionType	12
Figure 3: Restrictions of the allowed conditions for a Finite Element Mesh	13
Figure 4: Some quality checks modeled as instances of <i>ConceptDesignEvaluation</i>	17
Figure 5: Instance graph showing that the self-intersection has to be checked before performing the tasks: Product Styling, Product Design, Layouting, Simulation and Post-Processing , Meshing, Solving, Part Design, Product Design, and Shape Simplification.	17
Figure 6: Interlinking concrete software tools and quality checks of CAD models on the instance level.	18
Figure 7: Screenshot for the CQ “Which quality checks do I have to perform for the task Solving?”	21
Figure 8: Metadata of the instance <i>CheckForSelfIntersection</i>	21
Figure 9: Screenshot for the CQ “Which software tools support the quality check for self- intersection”	22
Figure 10: Results for CQ “Which types of geometric condition are necessary for a simplification model for analysis?”	23
Figure 11: Metadata of the instance <i>NoSelf-Intersection</i>	23

1 INTRODUCTION

The general objective of the Product Design Ontology (PDO from now on) is to assist in the development of shape processing tools for design. In this document we present the 3rd version of the PDO.

For a better understanding of the ontology's evolution we have summarized the evolution steps that correspond to the different versions of the PDO:

- The focus of the first version of the PDO was the modeling process, tool and shape know-how (in general) relevant to the phases of the product development process, i.e., the free-form shape modeling for styling and the engineering analysis.
- In the second version of the PDO we mainly focused on the simulation task, formalizing the entities involved in the process. In particular, the shape role, the geometric and the task-oriented conditions have been investigated and described in order to address the scenarios “FEA model preparation and analysis of the details effects” and “Acquisition of test data for FEA post-processing”.

The third version of the PDO deals with the open issues that arose during the development of the second version of the PDO. Following is a list of evolution steps taken in the third version of the PDO:

- When the shape role refers to the simplification task, the additional information needed is related to some conditions which can drive or constrain the simplification process, where a model simplification consists either in decreasing opportunely the number of elements of a mesh or de-features a complex CAD model represented by B-Rep. Examples of geometric constraints driving a mesh simplification are the edge size or ratio, and the minimum triangle angle. There could also be mechanical criteria guiding such a process, e.g., if volume or area variation are fixed according to the specific mechanical setting. Simplification conditions have been identified and included in the 3rd version of the PDO in association to the *SimplificationShape*.
- To be able to recover a certain shape role, a shape type needs to verify some geometric conditions. Therefore, the need of formalizing the concept of conditions related to a shape emerged, and the class *GeometricConditionType* has been introduced. At the moment, this class has two subclasses: *BRepConditionType* and *MeshConditionType*. To give an example, a mesh needs to fulfil some additional conditions (e.g. conformity) to fit the concept of *FiniteElementMesh*. We have introduced several instances of *GeometricConditionType* and of *ShapeRole* requiring such geometric properties.
- We populated the PDO with several instances related to *ShapeRole*, *ConditionType*, and *Task*.
- Finally, the scenario “Quality check of CAD-models” has been tackled, refocusing the CQs such that they are able to drive a significant extension of the PDO to integrate such a scenario.

The rest of the deliverable is organized as follows: in section 2 we describe new ontological elements that have been chosen and integrated in the PDO, yielding its evolution. Section 3 outlines our cluster contribution to the common ontologies. Section 4 addresses the PDO validation process for a quality check of the evolution steps: in fact, here we can verify if the evolved PDO is able to answer more CQs associated with cluster specific scenarios. Section 5 summarizes our dissemination activities; finally, section 6 gives the outlook concerning the future work for our cluster until the end of the project.

2 ONTOLOGY EVOLUTION

This section describes new ontological concepts that have been chosen and integrated in the PDO, yielding its evolution.

2.1 Modelling different condition types

In the second version of the PDO, we introduced the concepts of shape role and PD model to formalize the additional information usually associated to the shape along the development process. In the second version of PDO, the class *ShapeRole* was related with the class *ShapeConditionType* through the relation *hasShapeCondition*, used for specifying the geometric conditions that have to be satisfied by a shape in a certain design phase, whereas the relation *hasTaskSetting* was used in order to define necessary conditions required to perform a certain mechanical task (see D1.2.3.1 second version).

In this version, all these concepts have been grouped, so that any kind of condition a shape role must or should satisfy is placed in the same class. As a first consequence, we reorganized the conditions related to a shape role.

We introduced a general class *ConditionType* (see Figure 1), which groups *BoundaryConditionType* with its corresponding taxonomy and *GeometricConditionType*. The *BoundaryConditionType* class corresponds to the taxonomy of boundary condition types associated to a mesh during the analysis stage, which has been already defined in the previous version of the PDO. *GeometricConditionType* is an evolution of the previous *ShapeConditionType* class and models geometric conditions. It is further subdivided into *BRepConditionType* and *MeshConditionType* to distinguish the geometric properties applying to different shape types.

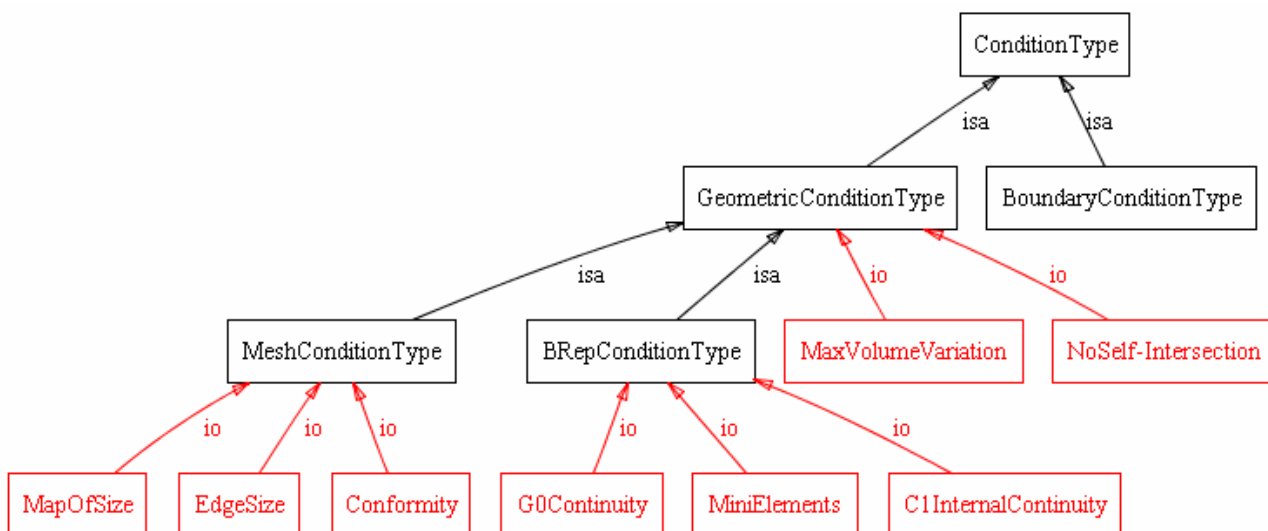


Figure 1: Partial visualisation of the ConditionType taxonomy.

Geometric Condition Type

In the formalization we introduced all conditions involving geometry for B-Rep and mesh types are placed in the *GeometricConditionType* class. The relation between the shape role and the condition type, which will be introduced in the next section, enables further differentiations. For example, an

instance of the *GeometricConditionType* will be also able to act as a mechanical condition during a simplification task.

The class *SimplificationCondition* has disappeared in the current version of the PDO. In fact, since candidate instances – in order to belong to this class – also include some geometrical meaning, they have been moved into the class *GeometricConditionType* or into the suitable subclasses.

Several instances have been identified for the class *GeometricConditionType*. As they in general may be referred to more than one shape representation type, they do not belong to a specific subclass. At the moment, all instances identified are properties that make sense only when two models are compared.

Deflection	<p>Maximum chordal deviation between two shape models. The two models could either have the same shape type, e.g. meshes, or have different shape types, e.g. a B-Rep NURBS model and its polyhedral approximation.</p> <p>It can be a condition supporting a <i>meshing</i> task.</p>
Max Area Variation	<p>Maximum variation of a surface area.</p> <p>This criterion may support the <i>simplification</i> task. Two examples where there is a strong need to maintain the surface area nearly constant are:</p> <ul style="list-style-type: none"> – During the shape adaptation of surface models based on shell or plate type elements (in case of volumetric models, the volume variation will be calculated instead); – When the area of a surface is loaded by a pressure boundary condition or a heat flux.
Max Variation of Center of Gravity	<p>Maximum variation of the center of gravity</p> <p>This criterion may support the <i>simplification</i> task. During the simplification of a geometric model, each transformation on curved areas introduces a displacement of the center of gravity. Then, transformations generating a displacement of the center of gravity larger than a prescribed threshold value should be avoided.</p>
Max Variation of Center of Inertia	<p>Maximum variation of the center of inertia</p> <p>This criterion may support the <i>simplification</i> task. During the simplification of a geometric model, each transformation on curved areas introduces a variation in inertia moments. Then, transformations generating a variation in some inertia matrix values larger than a prescribed threshold value should be avoided.</p> <p>This criterion, together with the previous one, is meaningful for volumes, free form surfaces, as well as planar sections when considering beam-shaped components.</p>
Max Volume Variation	<p>Maximum amount of mass variation</p> <p>This criterion may support the <i>simplification</i> task, e.g. helping to ensure that the mesh simplifications performed during a simplification task have no significant effect on the simulation results. Besides verifying local mass</p>

	changes, a stiffness criterion could be derived from this criterion, which evaluates the influence of local volume differences over the stiffness of the part.
No Self-Intersection	A self-intersecting B-Rep model does not represent a valid 3D object. The self-intersection test should be tolerance dependent.
No Duplicated Elements	The duplication of elements may introduce topological problems.

Mesh Condition Type

This class includes geometric conditions specific for meshes. Several instances have been identified.

Conformity	<p>If a mesh M of a field is composed of V_h elements, then M is conform if and only if it satisfies the following conditions:</p> <ol style="list-style-type: none"> 1. V_h has a non empty interior; 2. $M = \bigcup_{V_h \in M} V_h$; 3. The intersection of two elements of V_h is reduced either to the empty set or to a common part of their respective boundary. <p>This is a necessary property of the mesh in a <i>solving</i> task, i.e. when performing a mechanical analysis with FE methods.</p>
Edge Size	<p>Edge length in a mesh</p> <p>It could be a measure important to fix during a <i>meshing</i> and a <i>simplification</i> task in order to have edges with approximately the same length.</p>
Aspect Ratio	<p>Ratio between two characteristic lengths of one element. For triangles, the aspect ratio is defined to be either the ratio of the maximum horizontal length of the element to the maximum vertical length of the element, or the ratio of the diameter of a circumscribed circle to the maximal distance between vertices. For tetrahedrons, the aspect ratio can be defined as</p> $4\sqrt{\frac{3}{2}} \frac{P_k}{h_k}$ <p>where P_k denotes the diameter of the sphere circumscribed about the tetrahedron and h_k is the maximum distance between two vertices. These formulations yield a value of 1 for an “isosceles” tetrahedron (triangle) and a value of 0 for a degenerate (flat) element.</p> <p>This property of mesh elements could be interesting to check during a <i>meshing</i> and a <i>simplification</i> task, whenever a <i>solving</i> task follows. Usually, the closer this quantity to the value of 1, the better. Anyway, this is not a general rule for all mechanical analysis problems: for example, it is often better to distort the aspect ratio of the element when treating high speed fluid flow.</p>
Min Element Angle	<p>Minimum angle of mesh elements (i.e. triangles, tetrahedrons)</p> <p>This quantity can support a <i>simplification</i> and a <i>meshing</i> task.</p>

Map of FE sizes	<p>Map of FE sizes of a mesh</p> <p>It is an instance of <i>MeshConditionType</i> with a strong mechanical meaning, since it acts as a geometric mapping of the mechanical behaviour of the component.</p> <p>It is normally used for generating a FE mesh during a <i>meshing</i> task and is based on the user's expertise. It permits to characterize over the entire model the gradients of mechanical parameters for a given FEA problem, e.g., localising small FE where stress concentrations take place, and large FE in areas where the stress values stay constant.</p> <p>Some approaches exist which use the Map of FE sizes as a priori criterion to support and to drive a <i>simplification</i> task.</p>
------------------------	--

B-Rep condition Type

These conditions are related to the type of geometrical and topological information of the shape representation, which can become very critical during the transfer of a B-Rep Representation either from a CAD system to another or into the downstream applications.

Gi Continuity (i = 0, 1, 2)	G0, G1 or G2 continuity between different entities of the B-Rep object, to preserve topological links
X Internal Continuity (X = C1, G1, C2, G2)	C1, G1, C2 and G2 continuity within a NURBS curve or surface
Mini Elements	Small curves, small surfaces, narrow surfaces, and small distances between edge loops in a surface may introduce inconsistencies.
No Singularity	Singular points may result in algorithm failures
Polynomial Degree	High polynomial degrees can be a problem for geometry processing

Other condition types

Not only geometrical conditions and boundary conditions support shape roles. For example, a post-processing task could be driven by some conditions directly related to mechanical criteria, even if acting on the geometry. Currently, no specific class is dedicated to this kind of conditions, which have been modelled as instances of the *ConditionType* class. So far we defined:

- *APosterioriCriterionforLinearStaticAnalysis*. This condition enables to evaluate the influence of shape simplifications on the accuracy of FEA results. It is useful during a post simulation task and can be used for Linear Static Analyses.
- *APosterioriDiscretizationErrorCriterion*. This condition enables to evaluate the error generated by a bad object discretization on the FEA results. It is useful during a post simulation task.

2.2 Formalizing the relations between *ShapeRole* and *ConditionType*

In this new version of the PDO, *ShapeRole* has only one property, which connects it to the *ConditionType*, named *hasCondition*. To specify the relationship between *ShapeRole* and the different types of conditions, we used sub-properties modelled according to the following hierarchy:

- *hasConditions*:
 - *hasShapeCondition*
 - *hasNecessaryShapeConditions*
 - *hasPreferableShapeConditions*
 - *hasTaskCondition*
 - *hasNecessaryTaskCondition*
 - *hasPreferableTaskCondition*

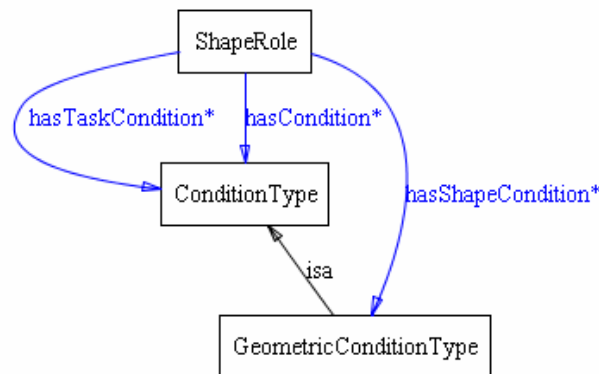


Figure 2: New relations between ShapeRole, ConditionType, and GeometricConditionType

The relation *hasShapeCondition* refers to a geometric condition, whereas *hasTaskCondition* refers to conditions required to fulfil a given task (cf. Figure 2). In detail, the property *hasShapeCondition* refers to the geometric conditions that shapes with type **ShapeRepresentationType** of the given **ShapeRole** must or should verify (cf. *hasNecessaryShapeCondition* and *hasPreferableShapeCondition*, respectively) *before* performing the task supported by the shape role. The range of the relation is **GeometricConditionType**. On the contrary, the property *hasTaskCondition* refers to conditions required or suggested (*hasNecessaryTaskCondition* and *hasPreferableTaskCondition*, respectively) *while* performing the task related to the shape role. These conditions can be either geometric conditions (e.g., edge size), as in the case of simplification tasks, or not (e.g., boundary conditions), as it happens, for instance, when performing simulation.

As already mentioned, the property *hasTaskSetting* of the previous PDO version has been substituted by *hasTaskCondition*, which takes a value in **ConditionType**, to clarify the kind of relation between the role and the task. Some restrictions have been fixed: for instance, if the shape role of a mesh is a **FiniteElementMesh**, then the *hasNecessaryTaskConditionType* contains at least values from **BoundaryConditionType** (cf. Figure 3); *hasNecessaryShapeCondition* contains at least *Conformity*, which is an instance of **MeshConditionType**. If necessary, other types of conditions could be added.

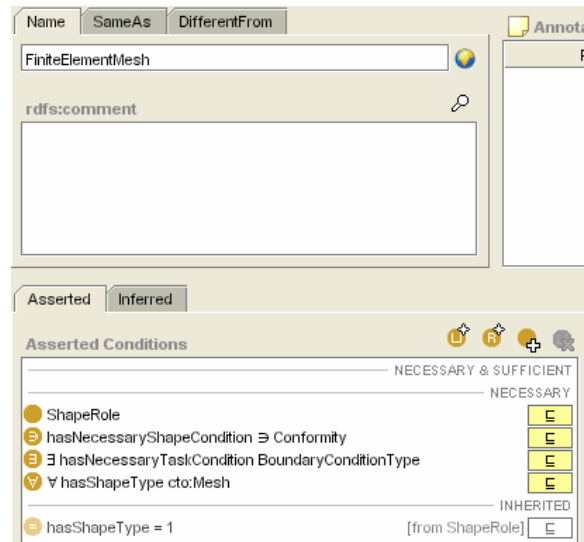


Figure 3: Restrictions of the allowed conditions for a Finite Element Mesh

We have created the class *SimplificationModelforAnalysis*, which is a sub-class of *SimplificationShape*. It is helpful to specify further shape simplifications performed during the preparation of a model for a FE analysis and to distinguish this simplification role from other types of simplification tasks (e.g. for visualisation). A *SimplificationModelforAnalysis* has an additional property in comparison with a *SimplificationShape*: it can be equipped with some boundary conditions, which drive the simplification task. Some instances have been added: *CADforVolumePreservingSimplification*, *MeshForEdgeSizeSimplification*, *MeshForIdealization*, *SimplificationMeshwithStaticBC*. Finally, we added a relationship between *ShapeRole* and *Task*, *supportsTaskInput*, and a relationship from *ConditionType* to *ShapeRole*, *isConditionOf* (with its subproperties *isTaskConditionOf*, *isPreferrableTaskConditionOf*, *isNecessaryTaskConditionOf*, *isShapeConditionOf*, *isPreferrableShapeConditionOf*, *isNecessaryShapeConditionOf*): in this way, we are able to answer CQs asking for the conditions required to perform a given task, like “Which are the necessary shape conditions to perform the Simulation task?”

2.3 Quality check of CAD models

When a CAD-model is received from a CAD-system, the feasibility of the use of the CAD-model for down stream processes or long time storage is to be determined. Two levels of problems are to be addressed:

- Defects that make the CAD-model non-manifold (within specified tolerances) and consequently invalid.
- Configurations that make down stream applications difficult.

There are a number of standards or standard proposals related to the quality control of CAD-models, which indicate the main criteria to be respected when transferring a model either from one phase of the development process to the other or from one CAx system to the other:

- AECMA-STAN LOTHAR – Long term archiving and retrieval of digital technical product documentation such as 3D, CAD and PDM data within the aerospace industry “ – Part 110 Explicit Geometry
- VDA/VDMA - CAD/CAM Working Group in the VDA Raw Material Committee (VDA-AK “CAD/CAM”) Scope and Quality of CAD/CAM data

- JAMA – Japan Automobile Manufacturers Association, PDQ Guideline
- SASIG – Product Data Quality Guideline for the Global Automotive Industry.

In the 1st version of the PDO (cf. the first version of D1.2.3.1) the scenario about the quality check of CAD models listed simply the typical set of evaluation tests to perform. Herein, we refined the scenario, specifying for the different criteria their importance related to the following transition between phases in product development:

- CAD → CAD
- CAS → CAD
- CAD → CAS
- CAD → FEM
- CAD → CFD

According to the above-mentioned standards and standard proposals, the most relevant quality checks are listed below, specifying that some of them can be further refined into sub-criteria:

1. **Identity between objects.** Check for duplication of elements.

Transition	Importance
All transitions	Important

2. **Mini elements.** Test for small curves, small surfaces, narrow surfaces, and small distances between edge loops in a surface.

Transition	Importance
CAD → CAD	Medium
CAS → CAD	Medium
CAD → CAS	Low
CAD → FEM	High
CAD → CFD	High

3. **Continuity between objects.** Check for G0, G1 or G2 continuity within defined tolerances.

Transition	Importance
CAD → CAD	G0 important
CAS → CAD	G0 important
CAD → CAS	Not important
CAD → FEM	G0 essential
CAD → CFD	G0 essential
All transitions	Problem and process dependent importance

4. **Continuity internal to objects.** Tests for C1, G1, C2 and G2 continuity within a NURBS curve or surface within defined tolerances.

Transition	Importance
All transitions	Problem and process dependent importance

5. **Polynomial degree.** High polynomial degrees can be a problem for geometry processing.

Transition	Importance
All transitions	System dependent

6. **Knots.** This test relates to the internal distance of knots describing the internal continuity of the NURBS curve or surface.

Transition	Importance
All transitions	Important

7. **Self-Intersections and Singularities.** A self-intersecting CAD-model, curve or surface does not represent a valid 3D object. The self-intersection test should be tolerance dependent.

Transition	Importance
CAD → CAD	Important
CAS → CAD	Important
CAD → CAS	Important
CAD → FEM	Essential
CAD → CFD	Essential

8. **Curvature.** The radius of curvature should not be below a predefined length.

Transition	Importance
All transitions	Problem dependent

9. **Knife Edges.** Models with sharp edges cannot be physically produced and such models must be avoided.

Transition	Importance
All transitions	Problem dependent

10. **Consistent orientation.** Tests are performed for edge loops and for the orientation of the surface normals. The test does not depend on a tolerance.

Transition	Importance
CAD → CAD	Moderate
CAS → CAD	Moderate

CAD → CAS	Moderate
CAD → FEM	Important
CAD → CFD	Important

11. **Topological consistency.** Check if the CAD model is manifold or not.

Transition	Importance
CAD → CAD	Moderate
CAS → CAD	Moderate
CAD → CAS	Moderate
CAD → FEM	Important
CAD → CFD	Important

12. **Topological Distances.** Check that the geometric description of a face (a surface) and an edge (the curve representing the edge) is within a predefined tolerance.

Transition	Importance
CAD → CAD	Moderate
CAS → CAD	Moderate
CAD → CAS	Moderate
CAD → FEM	Important
CAD → CFD	Important

13. **Additional model quality.** Tests here can be related to the number of elements describing the model, or the quality of the surface, e.g. if the surfaces are class A.

The evaluation of such properties usually requires tests involving tolerances, i.e. comparing calculated values with a tolerance. The current ontology tools used in AIM@SHAPE do not support neither automatic integration of processing tools nor testing involving \geq , \leq , $>$ or $<$. For this reason, we modelled only the type of check to perform and linked it to the proper transition phase in order to formalize at least the importance and the necessity of these evaluations in the process.

Ontology Modelling

When we started to integrate this scenario in the PDO we first evaluated the CQs already listed in the 1st version of the PDO concerning their suitability for the further development of the PDO. We discovered that most of the CQs can indeed be answered by available software tools (e.g., *Is the CAD model 3-manifold within specified tolerances* or *Are there self-intersections in the part of a surface used for describing a patch with trimmed away edges?*). Thus we did not modify the structure of the PDO on the basis of these CQs, but we used the Common Tool Ontology and modelled the available tools that are suitable to perform a specific quality check in a certain task. After analyzing the scenario we have identified three types of CQs regarding quality checks that are considered as meaningful concerning the further development of the ontology.

The first type of CQ refers to the relation *hasGeometricCondition* between instances of *ShapeRole*

and instances of *ConditionType*:

What type of conditions should a model have before performing a specific task?

For instance: What type of conditions should a model have before performing the task Solving?

Refining further, this type of CQ can also be specialized asking for geometric or task conditions, either necessary or preferable: for instance, what type of geometric conditions should a model have before performing the task *ShapeSimplification*? or what are the necessary geometric conditions that have to be satisfied before performing the task *ShapeSimplification*?

The second type of CQ is related to the task concept *ConceptDesignEvaluation* already present in the PDO from the beginning. The task *ConceptDesignEvaluation* in the Product Design Process is generally designed for verifying the feasibility of the design of the major parts (components) of a product (see the first version of D1.2.3.1). The different kinds of quality checks are part of the concept design evaluation and are run in different phases of the Product Design Process. According to this, we have extended the PDO by modelling the different quality checks as concrete instances of the task concept *ConceptDesignEvaluation*. For the sake of clarity, only some quality checks are presented in Figure 4.

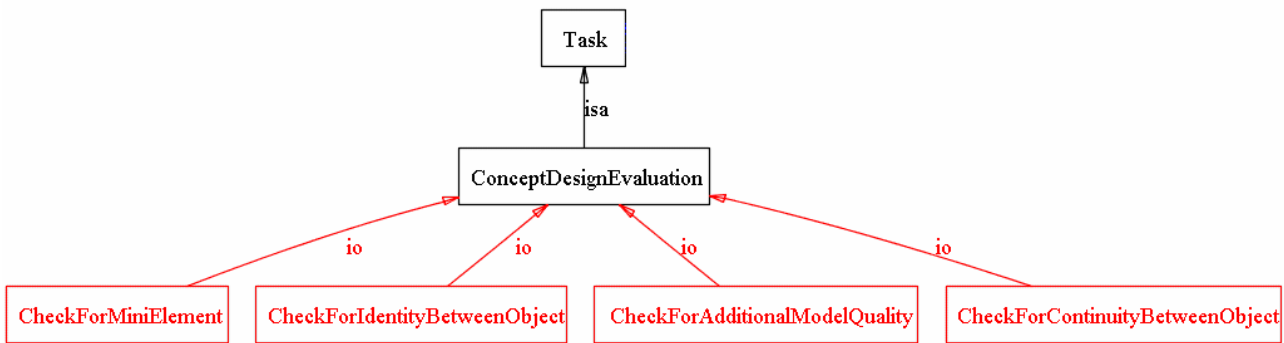


Figure 4: Some quality checks modeled as instances of *ConceptDesignEvaluation*.

The corresponding type of CQ is:

Which kind of checks do I have to consider when performing a specific task?

For instance, we can ask: which quality checks should I perform in the Concept Design Evaluation phase?

For any criterion, the design phase is specified before which a certain quality check has to be evaluated. For example, the instance *CheckforSelfIntersection* has as successor the tasks *Solving*, *PartDesign*, *Simulation_Post_Processing*, *Layouting*, *Meshing*, *ProductStyling*, *ShapeSimplification*, and *ProductDesign*. This means that to perform any of the listed tasks, the model is required to be tested for self intersections (see Figure 5).

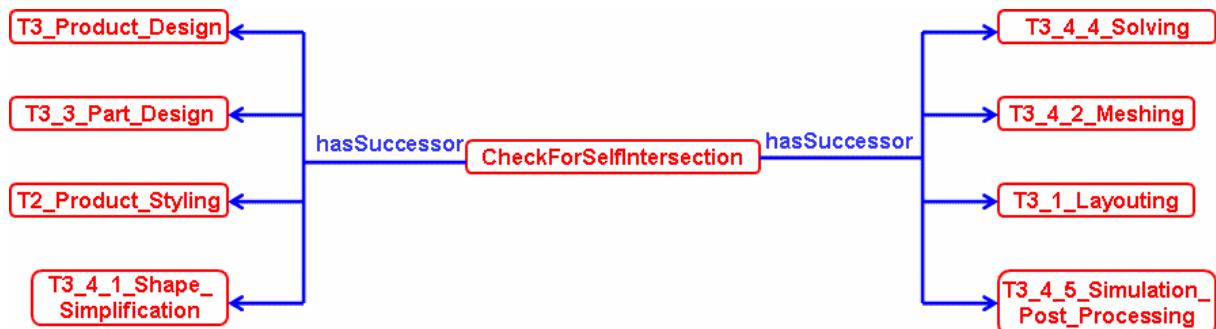


Figure 5: Instance graph showing that self-intersection has to be checked before performing the tasks: Product Styling, Product Design, Layouting, Simulation and Post-Processing, Meshing, Solving, Part Design, and Shape Simplification.

The third type of CQ refers to the relation between software tools and instances of the task concept *ConceptDesignEvaluation*.

Which software tools support the task *ConceptDesignEvaluation*?

For instance, which software tools support the task *CheckForSelfIntersection*?

In the 1st version of the PDO we already formalized the relation between the concepts *SoftwareTool* and *Task*. Both concepts were related via the relation *supports* and its inverse *isSupportedBy*, which permit to ask for tools, possibly present in the AIM@SHAPE Tool Repository, suitable for a specific task activity. As an example (by way of illustration) we show in Figure 6 that the tool package *GoTools* can be used to perform the quality check for self-intersection.

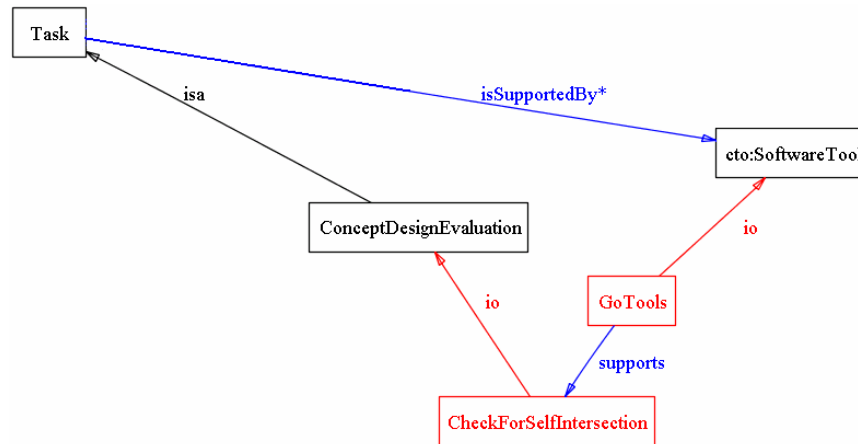


Figure 6: Interlinking concrete software tools and quality checks of CAD models on the instance level.

In AIM@SHAPE many functionalities for quality checks of CAD-models exist within the GoTools packages provided by SINTEF (one of our cluster members). In the first quarter of 2007 a new Module of GoTools for surface intersection and self-intersection will be made available under the GNU GPL license (http://www.sintef.no/math_software).

Through these modifications we are able to answer scenario-specific CQs in different variations. In section 4 we present a full list of the CQs that we can answer.

3 COMMON ONTOLOGIES

The development and the validation of the Common Ontologies through the three clusters in the AIM@SHAPE network is mandatory in order to align the Shape Repository and Tool Repository metadata and the ontological elements (classes and properties) of the Common ontology for Shapes (SCO) and Tools (TCO). The Common Ontology validation activity is related with *Task 1.5: Ontology and Metadata Integration and Validation*.

In the reporting period, we specialized the concepts *BRep* and *Mesh* and reorganised *ParametricRepresentation* and *ImplicitRepresentation*, the last two specialised for curves and surfaces, in the SCO to specify the shapes of interest for our cluster and to make the structure of the SCO more intuitive and explicit.

More specifically, the class *Mesh* has been modelled through the subclasses *ManifoldSurfaceMesh*, *ManifoldVolumeMesh*, and *NonManifoldMesh*. Consistently, we have introduced for the class *BRep* the subclasses *ManifoldSurfaceBRep*, *ManifoldVolumeBRep*, and *NonManifoldBRep*. Through encoding manifoldness within the concept names this crucial property of the shapes becomes more explicit in the SCO and cluster-specific CQs can be performed more intuitively. Moreover, specific

metadata have been validated to capture more detailed characteristics of this kind of shapes. In the 2nd version of D1.5.1 all the metadata already fixed for the mentioned classes are listed. Meanwhile, the PDO cluster is going to propose specific metadata for *NonManifoldBRep*, which are consistent with the ones of *NonManifoldMesh*, for the next validation step. Also metadata for *ManifoldVolumeBRep* will be selected.

Since the PDO imports both Common Ontologies, we had to reorganize affected instances in the PDO in order to be consistent with the evolved structure of the SCO. For example, instances of the class *BRep* which had an attribute-value pair *isManifold=false* became instances of the class *NonManifoldBRep*. However, the reorganization of the affected instances in the PDO was done manually which is time-consuming. We continuously monitor the activities of the Semantic Web community concerning a semi-automatic support for this problem. To our knowledge, there is currently no tool we could employ to make it automatically.

4 ONTOLOGY VALIDATION

The validation of the PDO through cluster-specific CQs is mandatory in order to check if the PDO fulfils its objective. In this section we present a table with **new CQs** that have validated the new structure of the PDO. The right column in the table indicates the status of the CQs (answered or not) and related comments.

No	Competency Questions (CQs) related to the <i>Quality Check of CAD models scenario</i>	Status
1	<ul style="list-style-type: none"> a. Is the CAD-model tested for self-intersections? b. Is the model without self-intersecting faces? c. Is the single surface/patch of model without ridges or vanishing normals? d. Does any hull in the CAD-model self-intersect? e. Do different hulls in the CAD-models intersect? f. Is there large variation in patch size within the model? g. Are there long and very narrow patches within the model? h. Are there extremely small patches in the model? i. Are there extreme curvatures in the model? j. Is each single face of the model 2-manifold? k. Do any of the surfaces used for describing the faces have self-intersections outside of the part used by the patch description? l. Are self-intersections in the part of a surface used for describing a patch trimmed away edges? m. Is the CAD-model tested for self-intersections? n. Is the CAD-model 3-manifold within specified tolerances? 	<p>As already mentioned in section 2.3, we propose to use available software tools for these CQs (suggested in the first version of the “Quality check” scenario) because an ontology is not appropriate.</p> <p>However, the PDO can provide these tools as far as they are in the Tool Repository.</p>
2	Is the model without self-intersecting faces?	answered
3	Which quality checks do I have to perform in the concept design evaluation phase?	answered
4	Which software tools are helpful to perform a specific quality check on a given CAD model?	answered
5	What type of geometric conditions should a model have before	answered

	performing a specific task ?	
6	Which kind of checks do I have to consider when performing a specific task?	answered
7	Which software tools support the task <i>ConceptDesignEvaluation</i> ?	answered
8	Which software tools support a specific quality check for self-Intersection?	answered
9	What types of geometric conditions are necessary for a given Shape Role?	answered
10	Which Shape Roles have to fulfill a given geometric condition type (e.g. conformity).	answered
11	What are the PDMODELS having a given Shape Role?	answered
12	What is the Shape Role needed as input of a specific task?	answered

To test the CQs and to validate the new structure of the PDO we have used the semantic-based AIM@SHAPE Search Engine. The above CQs are formulated in a general way. In fact, they can be formulated in different variations depending on the number of modeled instances.

For instance, Figure 7 shows the results for the concrete CQ “Which quality checks do I have to perform for the task Solving?” (cf. CQ No 6). As a result, we get a list of all quality checks that have to be considered before performing the task Solving. The quality check for self-intersection is one of those. Figure 8 shows the corresponding metadata related with the quality check for self-intersection.

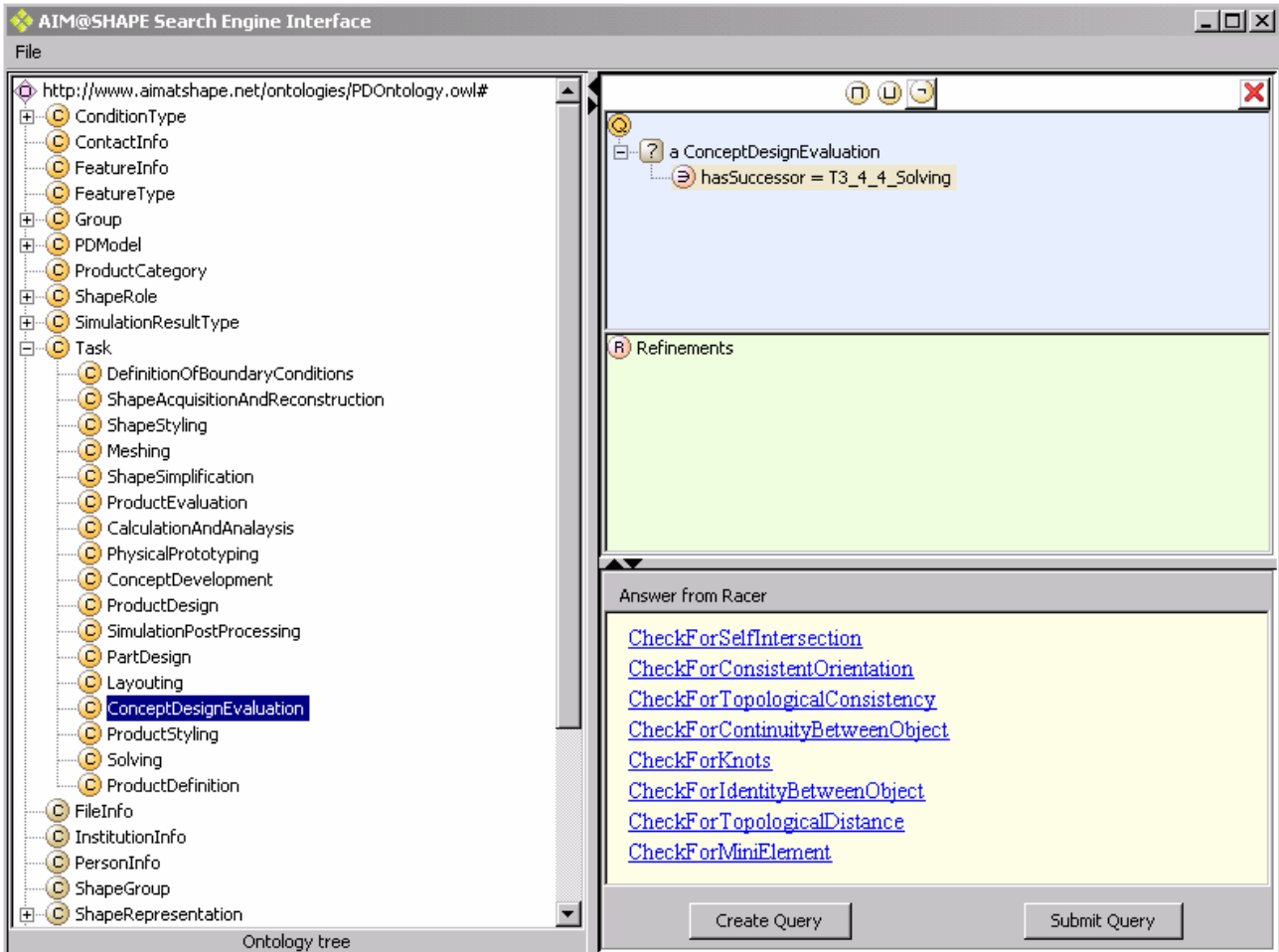


Figure 7: Screenshot for the CQ “Which quality checks do I have to perform for the task Solving?”

Metadata for - CheckForSelfIntersection [ConceptDesignEvaluation]		
Property type	Property name	Object
Object Properties	hasSuccessor	T3_4_1_Shape_Simplification
	hasSuccessor	T2_Product_Styling
	hasSuccessor	T3_4_2_Meshing
	hasSuccessor	T3_4_5_Simulation_Post_Processing
	hasSuccessor	T3_1_Layouting
	hasSuccessor	T3_3_Part_Design
	hasSuccessor	T3_4_4_Solving
	hasSuccessor	T3_Product [http://www.aimatshape.net/ontologies/PDOntology.owl#T3_4_4_Solving]
Datatype Properties	taskDescription	(7) Self-Intersections and Singularities. A self-intersecting C...

Figure 8: Metadata of the instance *CheckForSelfIntersection*

The next step is to retrieve some software tools that are able to perform these quality checks. Figure 9 shows the results for the CQ “Which software tools support the quality check for self-

intersection” (cf. CQ No 4 and 8). The result is the tool package *GoTools* provided by our cluster member SINTEF. In the final version we will populate the PDO with more software tools performing different quality checks on CAD models.

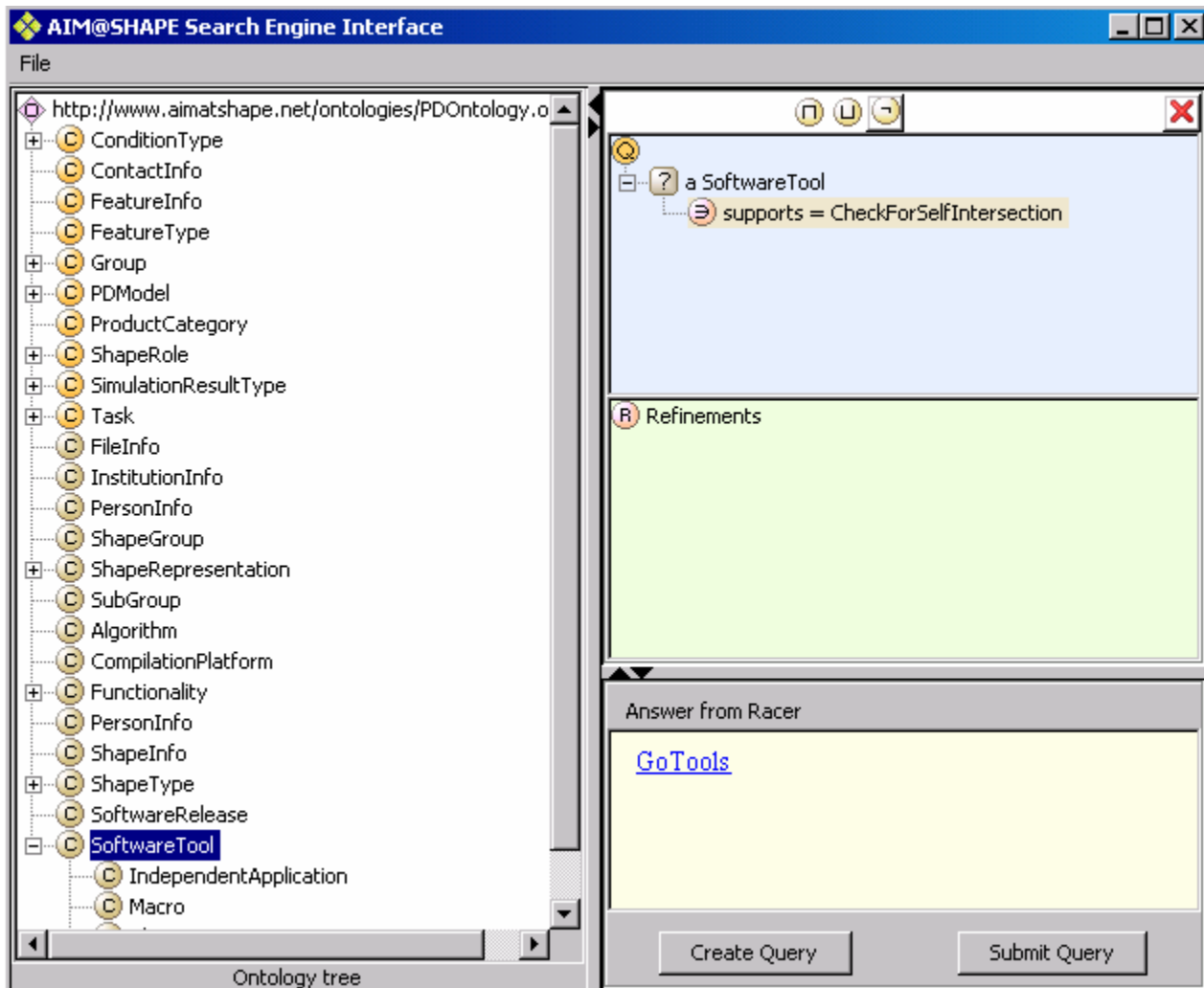


Figure 9: Screenshot for the CQ “Which software tools support the quality check for self-intersection”

Figure 10 and Figure 11 show the results of the CQ “Which types of geometric condition are necessary for a simplification model for analysis?” (cf. CQ No 5).

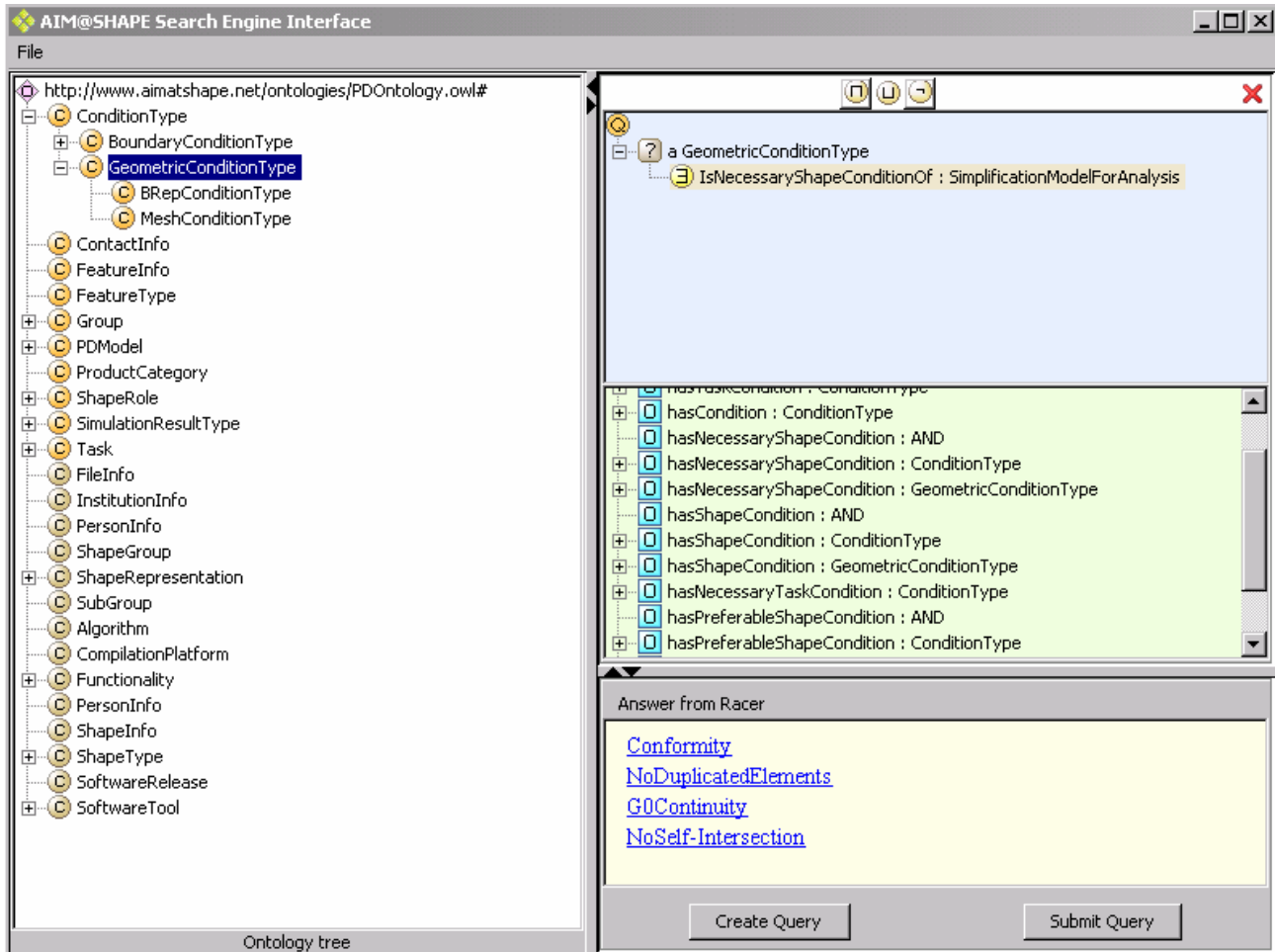


Figure 10: Results for CQ “Which types of geometric condition are necessary for a simplification model for analysis?”

Metadata for - NoSelf-Intersection [GeometricConditionType]		
Property type	Property name	Object
	IsConditionOf	CADModelForFEMCase1
	IsConditionOf	Digital2DSketch
	IsConditionOf	CADMockUp
	IsConditionOf	CADModelForTesselation
	IsNecessaryShapeConditionOf	CADforVolumePreservingSimplification
	IsNecessaryShapeConditionOf	Digital3DSketch
	IsNecessaryShapeConditionOf	CADModelForFEMCase1
	IsNecessaryShapeConditionOf	Digital2DSketch
	IsNecessaryShapeConditionOf	CADMockUp
	IsNecessaryShapeConditionOf	CADModelForTesselation
	IsShapeConditionOf	CADforVolumePreservingSimplification
	IsShapeConditionOf	Digital3DSketch
	IsShapeConditionOf	CADModelForFEMCase1

Figure 11: Metadata of the instance *NoSelf-Intersection*

5 DISSEMINATION ACTIVITIES FOR THE PDO

During the past months of research and developing activities, we have also worked on the preparation of material showing the results of the PDO to the industrial and research Community.

In particular, material for the DVD on AIM@SHAPE has been prepared (This activity is related with *Task 8.7: General and Large audience dissemination activities*):

- 1) a short presentation in HTML of the PDO
- 2) a long presentation in both HTML and PPT format of the PDO
- 3) two scenario-specific videos showing how the styling and simulation scenarios became realized with test instances.

The DVD material will be continuously updated with the latest results of our cluster activities.

And also:

- 4) a long presentation in PPT format for the International Summer School of AIM@SHAPE held in Tallinn, July 19-25 2006 (This activity is related with *Task 4.4: Annual School*)
- 5) Preparation of the paper “A Product Design Ontology supporting E-Science” (C.E. Catalano, E.Camossi (IMATI), R.Ferrandes (INPG/IMATI), V.Cheutet (INPG), N.Sevilimis (IGD)), and submission to ESWC 2007, 4th European Semantic Web Conference, June 3-7, 2007, Innsbruck (Austria).

6 FUTURE ACTIVITIES

For the 4th and final version of the PDO we aim to deliver a solid version of the PDO and to make it representative for the public.

We will further develop the PDO tackling grouping mechanisms. Several shape grouping concepts are thought off: shapes that represent the same object (in different representations, different formats), shapes that have been derived from each other via a chain of shape processing tools, shapes belonging to the same product category, shapes representing different variants of the same product, and shapes belonging to an assembly. We started to consider assemblies since they are commonly used in our research activity. Assemblies can be considered as groups but their complexity is not supported by the current notion of Group in the Shape Repository. Some preliminary discussions have been made already within the cluster, but a comprehensive modelling solution has not been reached yet.

Moreover, we will focus our work on an extensive validation of the PDO in combination with the evolving common Ontologies (close cooperation with Task 1.5). The PDO will be validated through cluster-specific scenarios and related CQs and by researchers involved in Task 7.9.

Moreover we will concentrate our efforts in populating the shape and tool repository with corresponding data (and metadata) in order to demonstrate how cluster-specific scenarios are becoming realized with real data.