Towards Ontological Support for Principle Solutions in Mechanical Engineering

Thilo BREITSPRECHER [a], Mihai CODESCU [b], Constantin JUCOVSCHI [c], Michael KOHLHASE [c], Lutz SCHRÖDER [d], and Sandro WARTZACK [a]

[a] *Department of Mechanical Engineering, FAU Erlangen-Nürnberg*
[b] *Department of Computer Science, Otto-von-Guericke-Universität Magdeburg*
[c] *Computer Science, Jacobs University Bremen*
[d] *Department of Computer Science, FAU Erlangen-Nürnberg*

Abstract. Among the standard stages of the engineering design process, the principle solution can be regarded as an analogue of the design specification, fixing the way the final product works. It is usually constructed as an abstract sketch where the functional parts of the product are identified, and geometric and topological constraints are formulated. Here, we outline a semantic approach where the principle solution is annotated with ontological assertions, thus making the intended requirements explicit and available for further machine processing; this includes the automated detection of design errors in the final CAD model, making additional use of a background ontology of engineering knowledge.

Keywords. Knowledge-based engineering, document-oriented processes

1. Introduction

Much like software engineering design (in an ideal world), design processes in mechanical engineering proceed in multiple stages successively refining abstract requirements into a final solution. This process of *systematic engineering design* is standardized in models that bear substantial resemblance to the V-model, such as the German VDI 2221 [16]. However, only the last stage in this process, corresponding to the actual implementation in software engineering, has well-developed tool support, in the shape of CAD systems that serve to document the final design. Other stages of the design process are typically documented in natural language, diagrams, or drawings. There is little or no support available for interconnecting the various stages of the design, let alone verifying that decisions made in one stage are actually implemented in the next stage.

Here, we embark on a program to fill this gap, focusing for a start on the last step in the development process, in which we are given a *principle solution* and need to implement this solution in the final design, a CAD model. The principle solution fixes design decisions regarding physical layout, materials, and connections but does not normally carry a commitment to a fully concrete physical shape. It is typically represented by a comparatively simple drawing, produced using plain graphics programs or even by hand. As such, it has a number of interesting features regarding the way it does, and also does not, convey certain information. The basic issue is that while one does necessarily indi-

cate only one concrete shape in the drawing, not all aspects and details of this sketch are actually meant to be reflected in the final design. While some of this is obvious, other aspects are less straightforward; e.g. symmetries in the drawing such as parallelism of lines or equal lengths of certain parts, right angles, and even the spatial arrangement and ordering of certain components may constitute integral parts of the principle solution or mere accidents of the sketch (work on sketch maps in GIS [7] may eventually help make automatic distinctions here).

The approach we propose in order strengthen and explicate the links between the stages of the design process is, then, to integrate the documents associated to each stage into a unified document-oriented engineering design process using a shared background ontology. This ontology should be strong enough to not only record mere hierarchical terminologies but also, in our concrete scenario of principle solutions, to capture as far as possible the qualitative design intentions reflected in the principle sketch as well as the requisite engineering knowledge necessary for its understanding. Such an ontology will in particular support the tracing of concepts and requirements throughout the development process; we shall moreover demonstrate on an example how it enables actual *verification* of a final design against constraints indicated in the principle solution.

An extended version of this work is available [3].

2. A Document-Oriented Process with Background Knowledge

We recall the stages of the *engineering design process* according to VDI 2221 [16].

S1 Problem: a concise formulation of the purpose of the product to be designed.
S2 Requirements List: a list of explicitly named properties of the envisioned product.
S3 Functional Structure: a document that identifies the functional components of the envisioned product and relates them to each other.
S4 Principle Solution: an abstract sketch capturing the core ideas of the design.
S5 Embodiment Design: a CAD design that specifies the geometry of the final product.
S6 Documentation: accompanies all steps of the design process.

An approach to vertical semantic integration of this process is outlined in [2]. Here, we describe step **S4** in more detail, as it offers the most obvious handles for adding value using semantic services, and discuss the structure of the ontology that drives them.

According to Pahl and Beitz [14], one can develop a principle solution for a product by combining working principles that correspond to the sub-functions identified in the function structure of the product. The search for applicable working principles and their ensuing combination in the principle solution is essential for the further product development. For example, the manufacturing costs are determined to a large extent by these decisions. However, a combination of working principles cannot be fully evaluated until it is turned into a suitable representation. At this stage of the design process, the engineer does not want to consider the formalities inherent to a full-fledged CAD system. For this reason, probably the most common representations of principle solutions are old-fashioned hand-drawn sketches. Developing the principle solution mainly involves the selection of materials, a rough dimensional layout, and other technological issues.

Our main case study concerns an assembly crane for lifting heavy machine components in workshops. The assembly crane to be designed (Fig. 1)

can be divided into modules performing various functions. The modules are indicated by numbers in the figure: the main frame with a vertical beam, a cantilever, and parallel horizontal base profiles (1); and a lifting system, consisting of an electrically powered winch unit (2), connected via a cable (3), which is guided via deflection rollers, to a crane hook (4). We are going to use the design decision that the legs of the frame should be parallel as a running example.

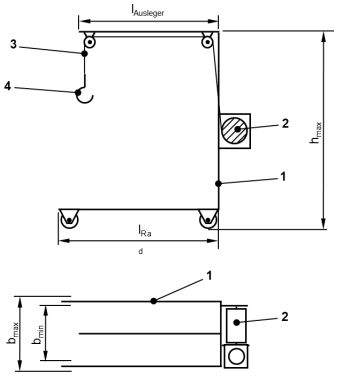

Figure 1.: The Assembly Crane

3. The Federated Engineering Ontology

The *Federated Engineering Ontology* (FEO) acts as the central repository of background knowledge. It serves as a synchronization point for semantic services, as a store for the properties of and relations between domain objects, and as a repository of help texts. As it has to cover quite disparate aspects of the respective engineering domain at different levels of formality, it is unrealistic to expect a homogeneous ontology in a single representation regime. Instead, we use the heterogeneous OMDoc/MMT framework [8] that allows representing and interrelating ontology modules via meaning-preserving interpretations. In particular, OMDoc/MMT supports the notion of meta-theories so that we can have ontology modules represented in OWL2 [6] alongside modules written in first-order logic, as well as informal modules given in natural language. Reasoning support is provided by the verification environment of the Heterogeneous Tool Set HETS [13], a proof management tool that interfaces state-of-the-art reasoners for logical languages. Within these frameworks, we employ the Distributed Ontology, Modeling and Specification Language DOL [12], which provides specific support for heterogeneity in ontologies.

A Verification Methodology We propose a general methodology for the verification of qualitative properties of CAD assemblies against principle solutions. While the checking of explicit *quantitative* constraints in principle solutions is supported by a number of research tools (e.g. the ProKon system [9]; in fact, some CAD systems themselves include constraint languages such as CATIA Knowledge Expert, which however are not typically interrelated with explicit principle solutions), there is to our knowledge currently no support for checking *qualitative* requirements given by the principle solution.

The first step is to provide a formal terminology for expressing the qualitative properties that a CAD design should fulfill. Here, we concentrate on geometric properties of physical objects and therefore we tackle this goal by developing an ontology of geometric shapes. We then need to have means to formally describe the aspects of a CAD design that are relevant for the properties that we want to verify. Since we want to verify geometric properties, we are going to make use of an ontology of CAD features. We then need to formulate general rules regarding geometric properties of objects constructed by repeated applications of CAD features. This gives us a new ontology, of rules relating geometric properties and CAD features.

We now come to the task of verification of a concrete CAD design against the requirements captured by a given principle solution. In a first step, we generate a representation of the requirements as an ABox T_R over the ontology of rules, explained below.

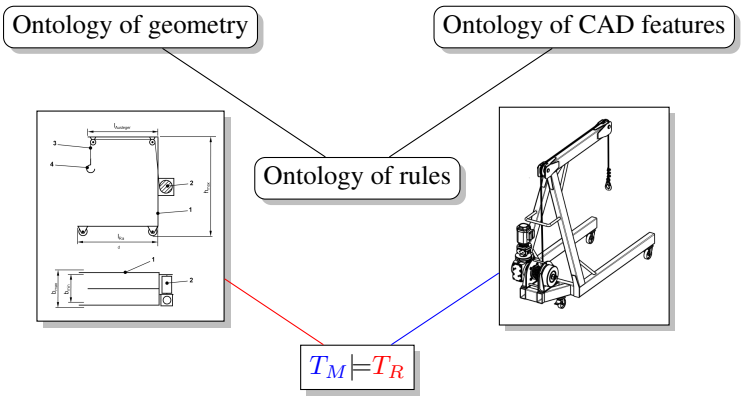

Figure 2. Verification of qualitative properties of CAD designs.

The next step is to generate a representation of the CAD design as another ABox T_M over the same ontology of rules, and then to make use of the rules to formally verify that T_M logically implies T_R. This process is illustrated in Figure 2.

Ontology of Shapes We begin setting up our verification framework by developing an ontology of abstract geometric objects, with their shapes and properties. The shape of a geometric object would seem to be a well-understood concept; however, the task of formalizing the semantics of shapes and reasoning about them is difficult to achieve in a comprehensive way. For a broader discussion, including some attempts to develop ontologies of geometric shapes, see, e.g., the proceedings of the Shapes workshop [10].

Our ontology, inspired by CYC [11], concentrates on geometric primitives of interest for CAD design . The central concept is that of *PhysicalObject*, which may be of an unspecified shape or can have a 2-dimensional or 3-dimensional shape. The object and data properties of the ontology are either parameters of the geometric shapes (e.g. diameter of a circle, or length of the sides of a square) or general geometric properties, like symmetric 2D- and 3D-objects and parallel lines.

We present the fragment of the ontology[1] of shapes that is relevant for asserting that two objects are parallel, a DOL specification that extends our OWL formalization of geometry with the axiom that two lines are parallel if the angles of their intersections with a third line are equal. Since the intersection of two lines is a three-place relation, the two intersecting lines and the angle between them, we use reification to represent it as a concept *Intersection*, together with a role *intersectsWith* that links to the first constituent line, a class *LineAngle* for pairs of lines with angles (with associated projection roles) and a role *hasLineAngle* that links to the pair of the second line of an intersection and the angle between the two lines. We denote the inverses of $hasLineAngle$ and $intersectsWith$ by $lineAngleOf$ and $hasIntersection$, respectively.

Ontology of CAD Features Inspired by [4], our ontology of features contains information about the geometry and topology of CAD parts. It describes assemblies and their parts, feature constructors and transformers, 2D sketches and their primitives, and constraints. We present here a fragment of the ontology of features that is relevant for veri-

[1]The current version of the ontology is available at `http://ontohub.org/fois-ontology-competition/FormalCAD/`.

fying that two objects are parallel. We have a concept of *3DPart* of an assembly and each part has been constructed in a 3D space which has 3 axes of reference. We record this by an object property *hasAxis*, with the inverse *isAxisOf*. Furthermore, 3D parts can be *constrained* at the assembly level. The constraint of interest for us is an angle constraint that specifies the angle formed between two axes, two edges or two faces of two chosen parts. Since this is again a relation with three arguments, we reify again to obtain a class *AngleConstraint* and three roles, *firstConstrainedLine* and *secondConstrainedLine* giving the two lines that are constrained and *constrainedAngle* giving the specified angle.

Ontology of rules The next step is to relate via rules the concrete designs using feature transformers and constructors, given as elements of the ontology of features, to the abstract shapes in the ontology of geometry. We make use of DOL *alignments* to express semantic relations between the concepts in the two ontologies, e.g. that each part is a physical object and that lines and angles in the same ontologies are equivalent. The outcome is that we can use DOL *combinations* to put together the two ontologies while taking into account the semantic relations given by the alignment. We can then further state that an angle constraint in an assembly gives rise to an intersection between the constrained lines and that two parts of an assembly are parallel if their axes are parallel.

Generating the ABoxes and proving correctness The principle solution is available as an image file, together with a text document that records additional requirements introduced in the principle solution, thus further restricting the acceptable realizations of the design. Each part of the sketch has been identified as a functional part of the principle solution and given a name; this yields the required individual names for our ABox. The assertions regarding the individuals thus obtained are added as semantic annotations to the text that accompanies the image e.g. the fact that *leg1* is parallel with *leg2*.
The ABox of the CAD design is generated from its history of construction, using a plugin for the CAD system. Thus we extract that the two legs of the crane have been explicitly constrained to be perpendicular to the main frame and coplanar in the CAD model.
Following Figure 2, we have to show that all models of the ABox generated from the CAD design are models of the ABox generated from the principle solution. DOL uses *interpretations* to express this; their correctness can be checked using one of the provers interfaced by HETS, e.g. the Pellet reasoner for OWL [15]; as expected, for our simple scenario the reasoner makes short work of this.

4. Conclusions

We have described a framework for semantic support in engineering design processes, focusing on the step from the principle solution to the final CAD model. We base our framework on a flexiformal background ontology, the FEO, that combines informal and semiformal parts serving informational purposes with formalized qualitative engineering knowledge and formal semantic annotation of principle sketches. The latter serve to separate contingencies of the sketch from its intended information content, and enable *automated* verification of the CAD model against aspects of the principle solution.

In the future, we plan to deepen and extend the FEO, and include in particular suitable feature ontologies originally developed to support interoperability and data interchange between CAD systems, such as OntoSTEP [1]. Moreover, we will extend the

FEO to cover the full development process, including requirements and function structure, taking into account existing work on knowledge-based systems for the latter [5].

We currently use OWL as the logical core of our verification framework. In principle, our approach is logic-agnostic, being based on heterogeneous principles, in particular through use of the Heterogeneous Tool Set HETS and the Distributed Ontology, Modeling and Specification Language DOL [12]. It is thus possible to go beyond expressivity boundaries of OWL where necessary, e.g. by moving parts of the ontology into first-order logic– this will increase the complexity of reasoning but HETS will localize this effect to those parts of the ontology that actually need the higher expressive power.

Acknowledgements. We acknowledge support by the German Research Foundation (DFG) under grant KO-2484/12-1 / SCHR-1118/7-1 (FormalCAD).

References

[1] R. Barbau, S. Krima, R. Sudarsan, A. Narayanan, X. Fiorentini, S. Foufou, and R. D. Sriram. OntoSTEP: Enriching product model data using ontologies. *Computer-Aided Design*, 44:575–590, 2012.

[2] T. Breitsprecher, M. Codescu, C. Jucovschi, M. Kohlhase, L. Schröder, and S. Wartzack. Semantic support for engineering design processes. In *Int. Design Conf., DESIGN 2014*. To appear.

[3] T. Breitsprecher, M. Codescu, C. Jucovschi, M. Kohlhase, L. Schröder, and S. Wartzack. Towards ontological support for principle solutions in mechanical engineering. In *Formal Ontologies Meet Industry, FOMI 2014*. To appear.

[4] G. Brunetti and S. Grimm. Feature ontologies for the explicit representation of shape semantics. *J. Comput. Appl. Technology*, 23:192–202, 2005.

[5] M. Erden, H. Komoto, T. van Beek, V. D'Amelio, E. Echavarria, and T. Tomiyama. A review of function modeling: Approaches and applications. *AI EDAM*, 22:147–169, 2008.

[6] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *J. Web Semantics*, 1:7–26, 2003.

[7] S. Jan, A. Schwering, M. Chipofya, and J. Wang. Qualitative representations of schematized and distorted street segments in sketch maps. In *Spatial Cognition 2014*, LNCS. Springer, 2014. To appear.

[8] M. Kohlhase. OMDOC – *An open markup format for mathematical documents [Version 1.2]*, vol. 4180 of *LNAI*. Springer, 2006.

[9] M. Kratzer, M. Rauscher, H. Binz, and P. Göhner. Konzept eines Wissensintegrationssystems zur benutzerfreundlichen, benutzerspezifischen und selbständigen Integration von Konstruktionswissen. In *Design for X, DFX 2011*. TuTech Innovation, 2011.

[10] O. Kutz, M. Bhatt, S. Borgo, and P. Santos, eds. *The Shape of Things, SHAPES 2013*, vol. 1007 of *CEUR Workshop Proc.*, 2013.

[11] D. Lenat. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *CACM*, 38:33–38, 1995.

[12] T. Mossakowski, O. Kutz, M. Codescu, and C. Lange. The distributed ontology, modeling and specification language. In *Modular Ontologies, WoMo 2013*, vol. 1081 of *CEUR Workshop Proc.*, 2013.

[13] T. Mossakowski, C. Maeder, and K. Lüttich. The Heterogeneous Tool Set, HETS. In *Tools Alg. Constr. Anal. Systems, TACAS 2007*, vol. 4424 of *LNCS*, pp. 519–522. Springer, 2007.

[14] G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote. *Engineering Design*. Springer, 3rd ed., 2007.

[15] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. Web Semantics*, 5:51–53, 2007.

[16] VDI. *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte (Systematic approach to the development and design of technical systems and products) – VDI 2221*, 1993.