# Modeling vs Encoding for the Semantic Web

**Editors:** Krzysztof Janowicz, Pennsylvania State University, USA and Pascal Hitzler, Wright State University, USA
**Solicited reviews:** Thomas Lukasiewicz, Oxford University, UK and Giancarlo Guizzardi, Federal University of Espírito Santo, Brazil
**Open review:** Pascal Hitzler, Wright State University, USA

Werner Kuhn
*Institute for Geoinformatics (ifgi)*
*University of Münster, Weselerstr. 253, D-48151 Münster (Germany)*
*kuhn@uni-muenster.de*

**Abstract.** The Semantic Web emphasizes encoding over modeling. It is built on the premise that ontology engineers can say something useful about the semantics of vocabularies by expressing themselves in an encoding language for automated reasoning. This assumption has never been systematically tested and the shortage of documented successful applications of Semantic Web ontologies suggests it is wrong. Rather than blaming OWL and its expressiveness (in whatever flavor) for this state of affairs, we should improve the modeling techniques with which OWL code is produced. I propose, therefore, to separate the concern of modeling from that of encoding, as it is customary for database or user interface design. Modeling semantics is a design task, encoding it is an implementation. Ontology research, for applications in the Semantic Web or elsewhere, should produce languages for both. Ontology modeling languages primarily support ontological distinctions and secondarily (where possible and necessary) translation to encoding languages.

Keywords: ontology modeling and encoding, semantic engineering, expressiveness, algebra, functional languages, Haskell.

## 1. Introduction

The Semantic Web transcends all previous attempts at enriching data with explicit semantics. Yet, the modeling languages brought to the task are weaker than those for conceptual modeling at smaller scales, such as databases or user interfaces. As a consequence, the Semantic Web rests on the premise that it is possible to produce and understand ontologies in OWL, using editors like Protégé1. Many of those who have tried this doubt the premise, especially if they have also done other kinds of conceptual modeling. Their experience in over three decades of conceptual modeling does not support the conclusion that description logic statements adorned with syntactic sugar and design patterns are sufficient (or even necessary) to capture what people mean when they use a vocabulary.

Modeling semantics is a design task, encoding it is an implementation. With the former we explore how to constrain human and machine interpretations of vocabularies, with the latter we support automated reasoning. Expressiveness is an essential criterion for the former, decidability for the latter. Mixing the two concerns is harmful for both tasks, but routinely done. While there are complementary approaches to encode semantics (for example, through machine

[1] Witness the online guide to Protégé: "The Protégé platform supports two main ways of modelling ontologies - frame-based and OWL" (http://protege.stanford.edu/doc/owl/getting-started.html).

learning), the scope of this note is limited to conceptual modeling.

## 2. Can we support our own goals?

Architects do not start a design by constructing geometric figures, database administrators do not start a project by creating relational tables, and user interface designers do not model interfaces in a user interface toolkit. Each of these fields has its modeling languages and environments, allowing, for example, to sketch a building, draw diagrams, or compose storyboards.

What does the Semantic Web offer in support of design? How does it assist modelers in choosing ontological distinctions, testing their implications, exploring their varieties, experimenting with applications?

Ontology engineers can choose from informal or semi-formal techniques, such as twenty question games, sorting tasks, concept mapping, or document analysis[2]. They can follow best practice[3] and get advice on pitfalls to avoid[4]. The more formally inclined designers can use methods like OntoClean [5], which ask key questions, for example, about identity and rigidity.

Yet, the gap between informal knowledge elicitation techniques, design patterns, and design methods on the one hand, and useful, tested OWL axioms on the other often remains too wide to jump across without breaking a leg or two. Evidence for this comes in the form of typical problems found in OWL ontologies:

- confusing instance-of with subclass-of;
- confusing part-of with subclass-of;
- leaving the range of an OWL property unspecified;
- introducing concepts and properties that are not sufficiently distinguished from others (a.k.a. "ontological promiscuity").

These and other well-known problems may just be attributed to sloppiness in modeling. However, if it is too easy to be sloppy without noticing it, the Semantic Web will have a serious quality and reputation problem. Also, some of these problems occur in prominent spots, such as Protégé's *Guide to Creating Your First Ontology* [5], which teaches us, for example, to model Côtes d'Or region as a class (!) and furthermore as a subclass-of Bourgogne region.

OWL ontologies cannot be expected to be directly written or understood by modelers, because OWL is optimized to support machine reasoning, not human thought. The Semantic Web is today at a stage of maturity that databases had passed in the 1970's and user interfaces in the 1980's, when they abandoned using a single paradigm for encoding and modeling. Relational algebra for database encoding and logical devices for user interfaces have been complemented by conceptual modeling languages like entity-relationship or state-transition diagrams, and subsequently by much more elaborate design techniques.

A likely consequence of the relatively immature state of modeling support is that today's Semantic Web contains assertions, whose implications have never been understood by anybody, and which may have been tested for satisfiability at best, but not for correctness or relevance.

In the face of this situation, some commonly encountered claims about the goals of the Semantic Web appear rather bold. For

---

2 see http://www.semanticgrid.org/presentations/ontologies-tutorial/GGFpart4.ppt for an excellent overview

3 for example, http://www.w3.org/2001/sw/BestPractices/OEP

4 http://www.ontologyportal.org/Pitfalls.html

5 http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

example, in a classical paper introducing OWL, it has been said that

"ontologies are expected to be used to provide structured vocabularies that explicate the relationships between different terms, allowing intelligent agents (and humans) to interpret their meaning flexibly yet unambiguously". [7]

The goal of *unambiguous interpretation* is a formidable one, and variants of the idea that ontologies contain "precisely defined meanings" are propagated throughout the Semantic Web literature and in countless project proposals. A recent paper co-authored by the Semantic Web's father, Tim Berners-Lee, even carries it forward to linked data, whose

"meaning is explicitly defined"

according to the authors [1]. Claims like these, if not taken with a very large grain of salt, vastly overstate the achievable goals of the Semantic Web and create expectations that are bound to be disappointed, at least with the currently available modeling support.

In the rest of this note, I will first suggest that the Semantic Web community should adopt a more modest *engineering view* of semantics. Then, I will argue why OWL is too weak for modeling. Finally, I will propose to use and develop modeling languages to complement today's encoding languages.

## 3. An Engineering View of Semantics

Neither linguists nor philosophers have so far been able to define *meaning* as an object of scientific study in a way that would capture what people mean when they use vocabularies. Thus, specifying particular "meanings", or targeting unambiguous interpretations, rests on shaky grounds, no matter how it is attempted. Yet, while philosophers figure out what meaning really means, information scientists and engineers can use ontologies pragmatically to *constrain interpretations*.

Ontology engineers have recommended striving for *minimal* ontological commitments [3], rather than for any kind of completeness in ontological specifications. In this spirit, I have recently proposed a pragmatic view of concepts and their specifications [8]. It treats *meaning as a process* to be constrained, rather than as an object to be defined. As the saying "words don't mean, people do" expresses, it is people who mean something when they use a vocabulary, rather than the words having a fixed meaning. Such a process view of meaning can be captured by the threefold view of concepts in the meaning triangle (Figure 1), involving people using

- *words* to
- express *conceptualizations* and to
- refer to *something* in the world.

For example, the community of English speakers uses the word "star" to refer to shapes like that in the bottom-right corner of figure 1. Note that a speaker's or listener's idea of a star may not exactly match the instance referred to.

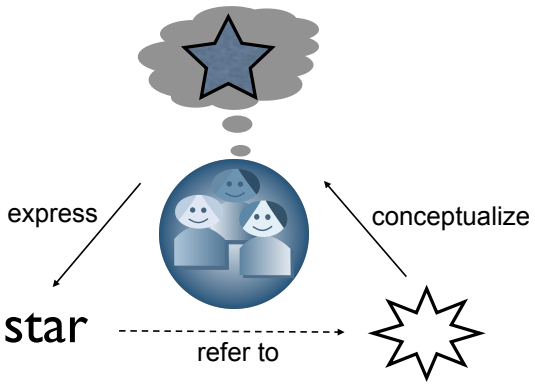


Figure 1: The meaning triangle

If one adopts such a triadic view of concepts, ontologies do not need to specify "meanings", much less the existence of something in reality. They simply provide humans or machines with constraints on how to apply and interpret vocabularies. These constraints support reasoning, while not removing all ambiguities.

The Semantic Web does not need to and probably cannot raise the bar higher than this semantic engineering goal. Traditional formal semantics is compatible with it, as long as truth is not put before meaning. Truth is a consequence of meaning, just as much as it is a cause, as the cyclic nature of the triangle implies. Truth conditions do not "define" meaning, but constrain interpretations of vocabularies to the ones that make sentences "true". For natural or informal technical languages, where there are no logical truth criteria, this is equivalent to sentences being correctly interpreted in the language community. For example, English speakers need to be able to correctly interpret the word "star" when it is used to refer to the bottom-right shape in figure 1.

## 4. Does OWL support modeling?

Naked OWL code does not convey much insight. Consider how hard it is to understand what somebody else's OWL statements say about a collection of concepts. OWL editors like Protégé provide class and property hierarchies as useful overviews, but do not show how the stated properties interact. Some context and rationale for the statements may be guessed from annotations, but the processes in which concepts are used remain informal and often invisible. Graph representations, showing classes as nodes and relations as edges, can be produced with Protégé plug-ins, but these tend to work only one-way and show only parts of the story. Even professional (and costly) development environments provide only limited and fractioned support for exploring, communicating, and evaluating – in other words for *modeling* before, while, and after encoding.

In addition to these usability and understandability issues, OWL and the current tools around it are often *not expressive enough* for modeling. For example, they

- treat properties and relations as the same, though these are two rather different ideas in modeling;
- limit us to binary relations, though interesting relations often start out as ternary or more;
- provide a well-defined primitive for taxonomies (subclass-of), but not for partonomies and other formal ontological relations;
- make it hard to encode processes and events, though these are often essential elements of semantics.

OWL's expressiveness may be sufficient (or even overkill) for eventual encoding and machine reasoning; but human understanding and reasoning require more expressive languages and we do not seem to understand yet what these should be. Web searches for "ontology modeling languages", for example, lead either to OWL or to UML or to requirements for better modeling languages with experimental implementations at best. While it is possible and valuable to introduce ontological distinctions into diagrammatic modeling languages like UML [6], automated reasoning support at the modeling stage requires more formal languages. Note that a call for more powerful modeling languages does not contradict the idea of lightweight approaches in implementation. On the contrary, better modeling tends to produce leaner, simpler encodings.

## 5. Toward Formal Modeling Languages

Modeling ontologies involves tasks like

- *finding out* what should be said,
- *understanding* what has been said,
- *conveying* that understanding to others,
- *checking* whether it is what was intended,
- *spotting errors* in what has been said, and
- *testing* whether what has been said is relevant and useful.

These are human reasoning and communication challenges that are unlikely to be met by reasoning and visualization at the encoding level. As Nicola Guarino proposed in [4], they require ontological distinctions built into modeling languages and automated reasoning support during modeling. What exactly the ontological distinctions should be remains an important research question. The formal ontological distinctions proposed by Guarino (essentially, those of OntoClean [5]) are immensely useful, but appear difficult to build into modeling languages.

Here, I will propose some distinctions that are already available in existing languages. Their choice has been motivated by the work of Joseph Goguen on algebraic theories [2] and 25 years of conceptual modeling applying these ideas, 15 years of which using the functional language Haskell (http://haskell.org), into which many of them are built. Similar ideas and arguments with a larger scope can be found in [9].

A fundamental ontological distinction is that between objects, events, and properties (these labels vary, but the basic idea remains the same). It is quite natural for humans to think in terms of objects and events with properties, rather than just in terms of predicates. Description logics do not allow for this distinction. Functional languages offer *formal* distinctions, based on kinds of functions. For example, properties (say, temperature) can be modeled as functions mapping objects (say, air masses) to values; events (say, a storm) can be modeled as functions mapping between objects (say, air masses again).

With a distinction between objects and events comes the need and opportunity to model the *participation* relation (say, of air masses in storms). Since participation is fundamental for semantics (witness the idea of thematic roles), having it as a modeling primitive would be very useful. *Types* in any language provide it. Data types and operation signatures capture the possible participation of objects in operations. Thereby, typing also provides a theory for instantiation, which is an undefined primitive in description logics: instances of a type are the individuals who can participate in its operations.

Distinguishing *properties* from *relations* is straightforward through unary functions for properties and n-ary (Boolean) functions for n-ary relations. Any interesting ontological relations (such as part-of or location) can then be specified through equational axioms, well known from algebraic specifications. For example, one can specify that an object that was put into another object is in that object as long as it has not been taken out again. Function composition allows for defining semantic constraints involving such sequences of events, which abound in practice. Some recent examples of these and other modeling capabilities of Haskell can be found in [11].

Modeling *roles* is much harder. Their antirigidity may be captured through so-called dependent types, whose instances depend on their values. Haskell type classes (generalizations over types) offer a useful model without explicit typing. For example, the types Person and University can be declared

as belonging to a type class which provides the functions of enrolling and unenrolling. Students are then modeled by terms like enroll (person, university), rather than by explicit typing.

Haskell's most useful support for modeling, however, stems from the fact that, as a programming language, it allows for *simulation*: ontological specifications can be tested through their constructive model, while and after being developed [12].

## 6. Conclusions

As Einstein is said to have pointed out, everything should be made as simple as possible, but not simpler. Conceptual modeling in the Semantic Web has been made to look simpler than it is. This carries the risk of yet another turn against Artificial Intelligence in some of its latest incarnations (not only the Semantic Web, but also Linked Data).

Yet, the idea of allowing information sharing with minimal human intervention at run time is too good to be discredited. Therefore, our challenges as researchers in this field are to

- promise only what we honestly believe we can achieve;
- work hard to achieve what we promised;
- validate what we have achieved;
- improve our theories and tools.

These challenges give us some re-thinking and re-tooling to do, and the new journal is a welcome place to report progress. It would be a pity if encoding-biased views about admissible approaches to ontology engineering precluded alternatives that support conceptual modeling.


## Acknowledgments

Among the many colleagues who helped shape these ideas over decades in discussions (without necessarily agreeing) are Andrew Frank, Joseph Goguen, Nicola Guarino, Krzysztof Janowicz, Ronald Langacker, David Mark, and Florian Probst, as well as the three reviewers.


## 7. References


[1] Bizer C., Heath T., Berners-Lee T., 2009. Linked Data - The Story So Far, http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf (preprint of a paper to appear in: Heath, T., Hepp, M., and Bizer, C. (eds.). Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS).

[2] Burstall, R. M., Goguen, J. A., 1977. Putting theories together to make specifications, Proceedings of the 5th International Joint Conference on Artificial intelligence, Cambridge, USA: 1045-1058.

[3] Gruber, T.R., 1995. Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, 43(5-6): 907-928.

[4] Guarino, N., 2009. The Ontological Level: Revisiting 30 Years of Knowledge Representation. Conceptual Modelling: Foundations and Applications. Essays in Honor of John Mylopoulos. Edited by A. Borgida, V. Chaudhri, P. Giorgini, E. Yu. Springer-Verlag: 52-67.

[5] Guarino, N. and Welty, C., 2002: Evaluating Ontological Decisions with OntoClean. Communications of the ACM, 45(2): 61-65.

[6] Guizzardi, G., 2005: Ontological Foundations for Structural Conceptual Models. Telematica Instituut Fundamental Research Series No. 015.

[7] Horrocks, I., Patel-Schneider, P.F., and Harmelen, F. van, 2003. From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics, 1(1).

[8] Kuhn, W., 2009. Semantic Engineering. In G. Navratil (Ed.): Research Trends in Geographic Information Science. Springer-Verlag, Lecture Notes in Geoinformation and Cartography: 63-74.

[9] Lüttich, K., T. Mossakowski, and B. Krieg-Brückner, 2005. Ontologies for the Semantic Web in CASL. In J. Fiadeiro, editor, WADT 2004, Springer-Verlag, LNCS 3423: 106–125.

[10] Ogden, C.K., Richards, I.A., 1923. The Meaning Of Meaning. Harcourt Brace Jovanovich.

[11] Ortmann, J., and Kuhn, W., 2010. Affordances as Qualities. 6th International Conference on Formal Ontology in Information Systems (FOIS 2010), IOS Press: 117-130.

[12] Raubal, M. and W. Kuhn, 2004. Ontology-Based Task Simulation. Spatial Cognition and Computation 4(1): 15-37.