The VLDB Journal (2004) 13: 120–147 / Digital Object Identifier (DOI) 10.1007/s00778-004-0120-x

ViSWeb – the Visual Semantic Web: unifying human and machine knowledge representations with Object-Process Methodology

Dov Dori[1,2]

[1] Technion, Israel Institute of Technology, Haifa 32000, Israel;
e-mail: dori@ie.technion.ac.il
[2] Massachusetts Institute of Technology, Cambridge, MA 02139, USA;
e-mail: dori@mit.edu

Edited by V. Atluri. Received: December 14, 2002 / Accepted: November 28, 2003
Published online: February 6, 2004 –

Abstract. The Visual Semantic Web (ViSWeb) is a new paradigm for enhancing the current Semantic Web technology. Based on Object-Process Methodology (OPM), which enables modeling of systems in a single graphic and textual model, ViSWeb provides for representation of knowledge over the Web in a unified way that caters to human perceptions while also being machine processable. The advantages of the ViSWeb approach include equivalent graphic-text knowledge representation, visual navigability, semantic sentence interpretation, specification of system dynamics, and complexity management. Arguing against the claim that humans and machines need to look at different knowledge representation formats, the principles and basics of various graphic and textual knowledge representations are presented and examined as candidates for ViSWeb foundation. Since OPM is shown to be most adequate for the task, ViSWeb is developed as an OPM-based layer on top of XML/RDF/OWL to express knowledge visually and in natural language. Both the graphic and the textual representations are strictly equivalent. Being intuitive yet formal, they are not only understandable to humans but are also amenable to computer processing. The ability to use such bimodal knowledge representation is potentially a major step forward in the evolution of the Semantic Web.

Keywords: Semantic Web – Visual Semantic Web – Object-Process Methodology – Conceptual graphs – Knowledge representation

1 The human-machine language orientation dilemma

A major assumption underlying the development of the Semantic Web is that humans and machines must each use a different format of knowledge representation. The first sentence of the introduction of the Resource Description Framework (RDF) [8] reads: "The World Wide Web was originally built for human consumption, and although everything on it is *machine-readable*, this data is not *machine-understandable*" (emphasis in source). Berners-Lee et al. [5] have noted that "*the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation*." Indeed, a major challenge in constructing a comprehensive Web-based knowledge management system is the ability to reconcile the apparent human-machine language orientation dilemma. On the one hand is the bulk of knowledge that has been published and continues to be gathered on the Web at an ever accelerating rate. This knowledge is expressed in free natural language, which in its raw form is currently indigestible to machines. Indeed, as the Cyc project [11] has demonstrated, analysis of unconstrained natural language by computers is far too difficult. On the other hand, current technologies that are aimed at enabling Web-based knowledge management are developed based on the premise that while humans prefer natural language as the primary means of recording, communicating, and disseminating knowledge, machines must be fed with barely human-intelligible, XML-like scripts, and if humans wish to extract their meaning, they have to sweat to mentally compile and decipher these scripts.

Based on experience gained over the past half century in developing artificial programming languages, the common wisdom has been that human-oriented and machine-oriented languages are necessarily mutually exclusive in terms of their understandability: natural languages, which are intuitively understood by humans, are much too difficult to be processed by computers, while (programming) languages, which are easily "understood" by computers, require much training and effort to be deciphered by humans. The underlying assumption has been that the syntax of human-understandable language must necessarily be totally different than that of machine-digestible language. This view is clearly reflected by Berners-Lee and Hendler [4], who have noted that "*... instead of asking machines to understand people's language, the new technology, like the old, involves* ***asking people to make some extra effort****, in repayment for which they will get substantial* ***new functionality***." The ViSWeb approach was born as a result of an attempt to provide the promised new functionality without having to ask humans to make the extra effort required by the current Semantic Web man-machine knowledge representation dichotomy. Achieving this requires that people who invent these new environments make some extra effort in try-

ing to cater to human intuition while ensuring that computers too will be able to process the same representation.

The implicit, conventional wisdom assumption, according to which human-readable and machine-readable formats are bound to be different, is the basis not only of RDF, but also of OWL, the Web Ontology Language [32]. OWL is intended to describe the classes and relations between them that are inherent in Web documents and applications. Appendix B of [32] contains references to many Semantic Web and ontology-related works. The introduction to OWL [32] reads: "*The World Wide Web, as it is currently constituted, resembles a poorly mapped geography.... In order to map this terrain more precisely, computational agents require machine-readable descriptions of the content and capabilities of web accessible resources. These descriptions* ***must be in addition*** *to the human-readable versions of that information*."

Challenging this assertion, according to which machine-readable descriptions must be added on top of the human-readable ones, this paper proposes a fresh approach to the issue of knowledge representation over the World Wide Web through ViSWeb – the Visual Semantic Web. ViSWeb is founded on the premise that human and machine Web-based knowledge need not be represented by two distinct formats. The ViSWeb (Visual Semantic Web) approach is based on Object-Process Methodology (OPM) [14]. Using a bimodal representation of graphics and text, OPM models knowledge about systems of various types and different complexity levels in a single model that integrates structure and behavior. OPM, described in more detail below, combines a subset of natural language, called Object-Process Language (OPL), with a formal yet intuitive graphic model, a set of one or more Object-Process Diagrams (OPDs) of exactly the same knowledge expressed in OPL. This dual graphic-textual representation constitutes a solid foundation for generic knowledge representation over the Web. Since the ultimate objective of the Semantic Web is enhancing the human ability to extract knowledge from the Web and understand it, it should involve human-understandable language. And since a combination of graphics and text is highly effective as a knowledge modeling language [14], we follow this paradigm in designing ViSWeb, in which the human and machine representations are effectively identical, enabling humans to benefit from the advantages of a dual, text-graphic knowledge representation.

Human vs. machine understanding and language readability

Before examining various approaches to graphic and textual knowledge representation, it is worth pointing out that the terms *machine-understandable* and *human-understandable* are different. *Understanding* is the result of accumulated and mentally digested knowledge, which enables building a mental model, specifying cause and effect, and predicting future phenomena based on the model. While knowledge is sometimes attributed to computers, understanding, at least in its original sense, is a human trait that requires natural human intelligence to piece together bits of knowledge, information, and even data to make sense of and generalize them. Computers cannot really understand in the human-oriented sense of the concept. At best, they can exhibit behavior, such as generating sentences in natural language, as we shall see in this paper, which, to an outside human observer, superficially suggests that they understand.

Synthesis of sentences that make perfect sense to humans using a subset of natural language while still being amenable to reliable computer processing and compilation is not only feasible, it has already been achieved to an astonishing level of sophistication by Object-Process Language (OPL), a subset of English generated automatically on the fly in response to graphic human input into an Object-Process Diagram (OPD). OPL is the textual modality of Object-Process Methodology (OPM) designed to favor the human over the machine in its closeness to being natural yet formal to a sufficient degree to enable concise, unambiguous machine processing.

The following sections contain a survey and assessment of various graphic and/or textual knowledge representation approaches and methods and expose the reader to principles of Object-Process Methodology. OPM with the appropriate Visual Semantic Web extensions is then elaborated upon as a human-understandable layer on top of RDF [8, 20] for specifying knowledge over the Web.

2 Combining graphic and textual knowledge representations

A powerful knowledge modeling and communication modality that is complementary to language is graphics. Diagrams are often invaluable for describing models of abstract things, especially complex systems. The fact that people from the early caveman days to date have been using some kind of sketching or diagramming technique to express their knowledge or ideas is testimony to the viability of graphic representation. However, such representation of our knowledge is valuable only if it is backed by a comprehensive and consistent modeling methodology. Such methodology is essential if we want to represent knowledge, understand complex systems in any domain, and communicate our understanding to others. An accepted diagramming method has the potential of becoming a powerful modeling tool if it constitutes an unambiguous language. In such visual formalism, each symbol must bear defined semantics and the links among the symbols must unambiguously convey some meaningful information that is clearly understood by the diagram readers.

Knowledge representation approaches

A number of knowledge representation approaches have been designed with the goal of graphically and/or textually representing knowledge aimed at facilitating human understanding and communication of knowledge. These approaches include concept maps, semantic networks, conceptual graphs (CGs), Knowledge Interchange Format (KIF), the Common Logic (CL) Standard initiative, Unified Modeling Language (UML), and Object-Process Methodology (OPM), which is the basis for the Visual Semantic Web presented in this paper. This section briefly surveys these knowledge representation approaches and assesses their potential as candidates for use in the Semantic Web. Since CGs and OPM came out as the two finalists, these two approaches are compared in terms of

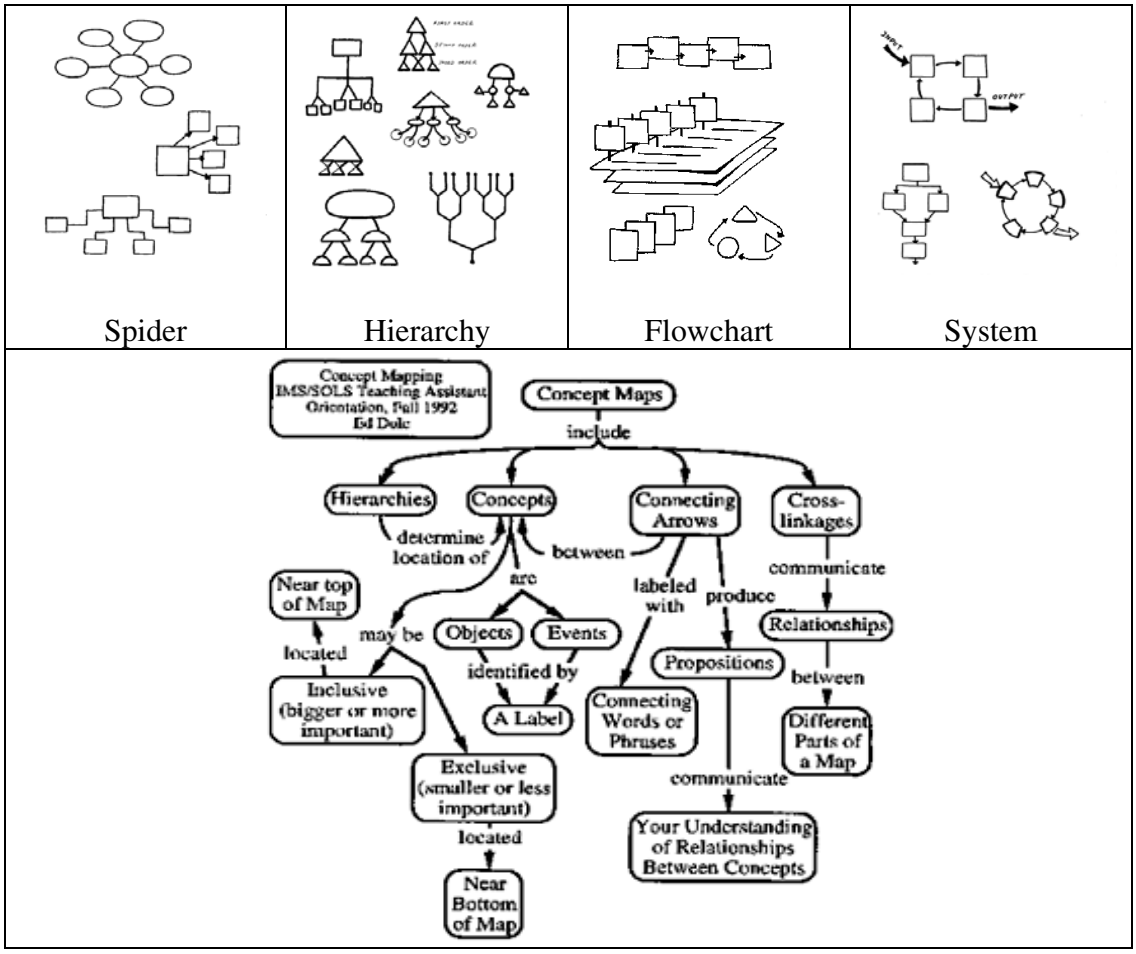


Fig. 1. *Top*: Four basic types of concept maps. *Bottom*: A hierarchy concept map by Ed Dole (adapted from http://classes.aces.uiuc.edu/ACES100/Mind/CMap.html)

their expressive power, level of formality, text-graphics equivalence, closeness to natural language, and applicability to system representation over the Web. The objective of this comparison is to argue for or against one method or the other as an enhancement of the Semantic Web that augments it with visual and linguistic modeling capabilities.

Concept maps

Concept maps [2] offer a method of representing information visually in a variety of ways shown in Fig. 1. Concept maps provide a complementary alternative to natural language as a means of communicating knowledge. Capitalizing on the fact that visual imagery plays a significant role in the creative processes of many people, concept maps are informal graphic representations that harness the power of the human vision processing to interpret and understand incoming complex information "at a glance".

As the top part of Fig. 1 shows, concept maps come in a number of variants. The bottom part of Fig. 1 can be considered a "metamodel" of concept maps, showing the concepts that constitute concept maps and how they interrelate. Concept maps have been used in education, policy studies, and the philosophy of science to provide a visual representation of knowledge structures and forms of argument. Under the concept map umbrella, Gaines and Shaw [18] have included semantic networks in artificial intelligence, bond graphs in mechanical and electrical engineering, Petri nets in communications, and category graphs in mathematics. Perhaps the most significant use of concept maps has been in education since 1977, when Novak [25] developed a system of concept maps that has been widely applied in evaluating students' learning outcomes [26].

Concept maps are human oriented, and, while being graphics-based, natural language sentences can readily be extracted from them, so their expressive power and closeness to natural language are quite powerful. However, the expressive power of concept maps and the many flexible ways they can be drawn come at the expense of formality, the lack of which precludes their being serious candidates for machine-processable system representation in the context of the Semantic Web.

Semantic networks

Semantic networks [21] represent information as a graph, in which a series of nodes that are connected to each other convey some semantics. An example semantic network is given in Fig. 2.

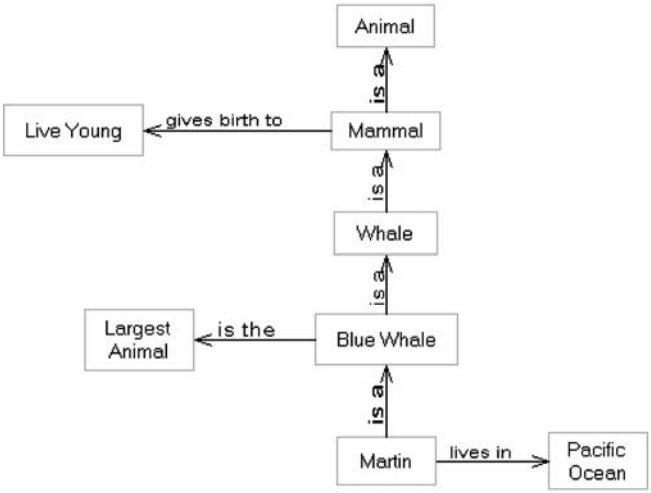


Fig. 2. Exmple of semantic network [24]

The relations between the entities, contained in nodes, are shown by the words along the arrows (directed arcs). Semantic networks can represent some relations in logic, but they are limited due to lack of quantifiers (such as "for all" or "exists").

In the 1970s, semantic networks were proposed in the AI community as the foundation for knowledge representation in humans [1,6]. Semantic networks can be considered a restricted, more formal class of concept maps. However, as candidates for enhancing the Semantic Web with visual capabilities, like concept maps, they still suffer from problems due to lack of a sufficient level of formality and expressive power.

Topic maps

An XML topic map (XTM) [31] is one or more interrelated documents employing a model and grammar for representing the structure of information resources used to define topics and the associations (relationships) between topics. The objective of topic maps is "to enhance the World Wide Web by leveraging the XML family of specifications". Names, resources, and relationships are said to be *characteristics* of abstract subjects, which are called *topics*. Topics have their characteristics within *scopes*. Scopes are the limited contexts within which the names and resources can be considered relevant. While the name "topic map" may suggest a graphical representation (like concept map), it is in fact a pure XML-based notation. The only graphics used in [31] is a variant of UML class diagram in an appendix that describes an informative conceptual model of XTM. Hence topic maps cannot enhance the Semantic Web graphically or visually.

Unified modeling language (UML)

UML [27], the OMG standard for object-oriented software systems, is an agglomeration of about nine sets of diagramming conventions known interchangeably as "models" or "views" that are aimed at different types of users (designers, end users, programmers, etc.). They show various aspects of the system that can be broadly classified into structure and behavior. The multiplicity of models makes the general problem of maintaining coherence among them intractable since each change in one model gives rise to potential changes in the rest of the models, which, in turn, ripples through [15] and may cause a circular effect. With respect to specifying just the structure of systems, which is the focus of the current efforts of the Semantic Web, the class and object diagrams are sufficient. Being the seed of almost all the UML ancestor OO modeling languages, these are probably the two best understood and least controversial types of diagrams. However, even these diagrams have no human-readable textual counterpart, let alone a subset of natural language, so they cannot be candidates for representing knowledge over the Semantic Web, where the textual part is mandatory. Moreover, when the Semantic Web starts moving from specifying just static knowledge to modeling system dynamics, the other UML models will need to be recruited as well, which will immediately raise the model multiplicity problem [30].

Conceptual graphs

Conceptual graph (CG) [9,33,34] is a system of logic based on semantic networks and the earlier existential graphs of Charles S. Peirce [29]. CGs express meaning in a form that is logically precise, human readable, and computationally tractable [13]. Since CGs bear direct mapping to language, they can potentially serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they may serve as a readable yet formal design and specification language. For example, while discussing the embedding of knowledge in Web documents and comparing CGs with XML metadata languages, Martin and Eklund [22] point to the advantages of the CG formalism for expressing metadata. Delteil and Faron [12] have proposed an expressive concept description language (CDL) for real-world applications that combines features of CGs and Description Logics (DLs).

CGs are formally defined by an abstract syntax that is independent of notation but can be represented in either graphical or character-based notations. A CG g is a bipartite graph whose two kinds of nodes are called concepts and conceptual relations. Every arc of g must link a conceptual relation r in g to a concept c in g. The CG g may have concepts not linked to any conceptual relation, but every arc that belongs to any conceptual relation in g must be attached to exactly one concept in g. As an example, Fig. 3, adapted from [4], shows the display form (DF) of a conceptual graph that represents the English sentence *John is going to Boston by bus*.

In DF, concepts are represented by names within rectangles: `[Go]`, `[Person: John]`, `[City: Boston]`, and `[Bus]`. Conceptual relations are represented by circles or ovals: `(Agnt)` relates `[Go]` to the agent John, `(Dest)` relates `[Go]` to the destination Boston, and `(Inst)` relates `[Go]` to the instrument bus. The arcs that link the relations to the concepts are represented by arrows. For relations with more than two arguments, the arcs are numbered. The linear form (LF) is intended as a more compact notation than DF, but with some human readability. It is exactly equivalent in expressive power to the abstract syntax and the DF. Following is the LF for Fig. 3:

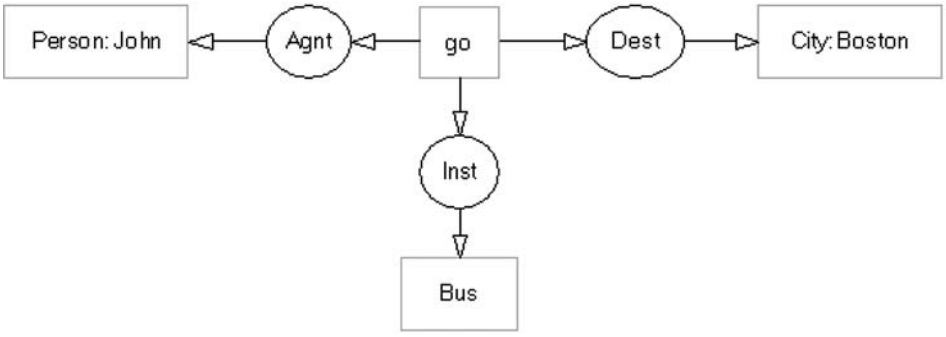


Fig. 3. A snapshot CG Display Form (DF) for "*John is going to Boston by bus.*".

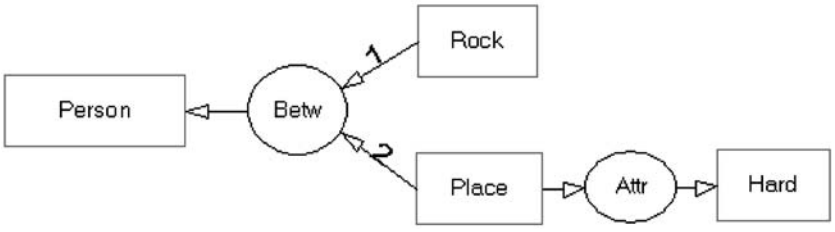
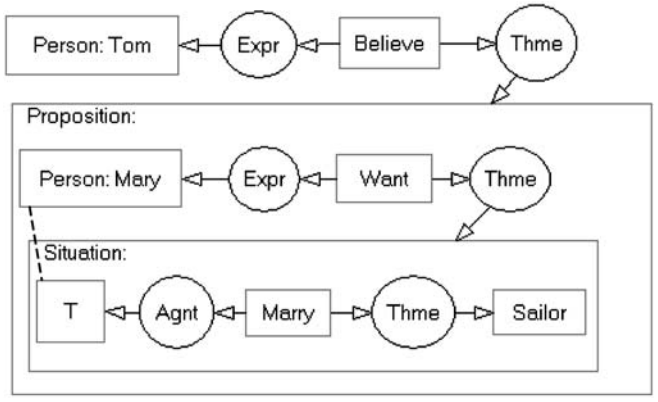


Fig. 4. A snapshot CG DF for "*A person is between a rock and a hard place.*"




```
[Person: Tom]←(Expr)←[Believe]→(Thme)-
   [Proposition: [Person: Mary *x]←(Expr)←[Want]→(Thme)-
      [Situation: [?x]←(Agnt)←[Marry]→(Thme)→[Sailor]]].
```

Fig. 5. A CG DF (*top*) and LF (*bottom*) for "*Tom believes that Mary wants to marry a sailor.*"

```
[Go]-
(Agnt)->[Person: John]
(Dest)->[City: Boston]
(Inst)->[Bus].
```

In this form, the concepts are represented by square brackets instead of boxes, and the conceptual relations are represented by parentheses instead of circles. The hyphen on the first line indicates that the relations attached to `[Go]` are continued on subsequent lines.

Both DF and LF are designed for communication with humans or between humans and machines. For communication between machines, the Conceptual Graph Interchange Form (CGIF) has another, shorter syntax. Following is the CGIF for Fig. 3:

```
[Go *x] (Agnt ?x [Person 'John']) (Dest
?x [City 'Boston']) (Inst ?x [Bus])
```

CGIF is intended for transfer between computer systems that use CGs as their internal representation. For communication with systems that use other internal representations, CGIF can be translated to the Knowledge Interchange Format (KIF) [19], yielding the following KIF expression for the above CGIF format:

```
(exists ((?x Go) (?y Person) (?z City)
(?w Bus))
(and (Name ?y John) (Name ?z Boston)
(Agnt ?x ?y) (Dest ?x ?z) (Inst ?x ?w)))
```

Figure 4 shows the DF adapted from [13] of the sentence "*A person is between a rock and a hard place.*"

A CG *context* [36] is a concept with a nested conceptual graph. In Fig. 5, adapted from [13], the concept of type `Proposition` is a context that describes a proposition that Tom believes. Inside that context is another context of type `Situation`, which describes a situation that Tom believes

Mary wants. The resulting CG represents the sentence "*Tom believes that Mary wants to marry a sailor*."

Harnessing the CG formalism to RDF querying

Corby et al. [10] have utilized RDF, which is elaborated upon below, for expressing metadata and interpreting them in conceptual graphs to exploit query and inference capabilities enabled by CG formalism. They show that CGs can be used as a means to exploit RDF metadata to handle metadata-based search queries and present mapping of RDF into CG and its application in the context of the Semantic Web. The principle of the mapping relies on considering an RDF description as an instance of a resource CG concept type and the associated properties as relations of this concept. The designator of a resource concept is the URI of the resource itself. For example, consider the following knowledge in RDF/XML syntax:

```
<rdf:Description about='http://
  www.bookstore.org/id1971'>
<author>John Rawls</author>
<title>A theory of Justice</title>
<date>1971</date>
< /rdf:Description>.
```

This specification can be interpreted in CG as

```
[Resource :http://
www.bookstore.org/id1971]-{
  ->(author)->[Literal :John Rawls]
->(title)->[Literal :A theory of
  Justice]
->(date)->[Literal :1971]}.
```

This mapping is quite straightforward, and it demonstrates the fact that XML/RDF tags are converted to CG relation nodes. For example, the RDF line "`<date>1971</date>`" is translated to "`...(date)->[Literal :1971]`", showing that the RDF tag "`<date>` was translated into the CG relation node `(date)`, linking the concept nodes `[Resource :http://www.bookstore.org/id1971]` and `[Literal :1971]`". The work does not address the issues involved in graphic representations of CGs and DRF graphs.

The Common Logic Standard initiative

Although DF, LF, CGIF, and KIF look different, their semantics is defined by the same logical foundations. Semantic information expressed in any one of them can be translated to the others without loss or distortion, but formatting and stylistic information may be lost in translations between these formats. Based on this similarity, the Common Logic (CL) Standard initiative [35] aims at reducing the bias toward its two starting notations, KIF and CGs, and to emphasize the common basis in first-order logic, as it was originally developed during the late 19th and early 20th centuries by Frege, Peirce, Schröder, Peano, and many others. CL is in the process of being defined by an abstract syntax that specifies the major categories, such as Quantifier, Negation, and Conjunction, without specifying any concrete symbols for writing them. At the abstract level, even the ordering is left undefined, so that there is no bias toward a prefix notation, such as KIF, an infix notation, such as predicate calculus, or a graph notation, such as CGs. Although the CL project grew out of collaboration between the KIF and CG communities, the initiators hope that the CL standard will be used for many other languages that have a declarative semantics such as RDF, UML, DAML, or topic maps. Any language that can be mapped to and from the CL abstract syntax would automatically inherit the same model-theoretic semantics and would therefore be semantically compatible and interoperable with software based on any other languages that adopt the CL semantics.

Object-Process Methodology

Most interesting and challenging systems are those in which structure and behavior are highly intertwined and hard to separate. Motivated by this observation, Object-Process Methodology (OPM) [14] is a holistic approach to the study and development of systems that integrates the object-oriented and process-oriented paradigms into a single frame of reference. Structure and behavior, the two major aspects that each system exhibits, coexist in the same OPM model without highlighting one at the expense of suppressing the other. Due to its structure-behavior integration, OPM provides a solid basis for modeling complex systems in general and those documented through the Semantic Web in particular.

The elements of the OPM ontology are entities and links. Entities, the basic building blocks of any system modeled in OPM, are of three types: *objects* with *states* and *processes*. *Objects* are (physical or informatical) things that exist, while *processes* are things that transform objects. *Links* can be structural or procedural. *Structural links* express static, time-independent relations between pairs of entities. The four fundamental structural relations are aggregation-participation, generalization-specialization, exhibition-characterization, and classification-instantiation. *Procedural links* connect entities (objects, processes, and states) to describe the behavior of a system.

The behavior is manifested in three major ways: (1) processes can *transform* (generate, consume, or change the state of) objects; (2) objects can *enable* processes without being transformed by them; and (3) objects can *trigger* events that invoke processes if some conditions are met. Accordingly, a procedural link can be a transformation link, an enabling link, or an event link. A *transformation link* expresses object transformation, i.e., object consumption, generation, or state change. An *enabling link* expresses the need for a (possibly state-specified) object to be present in order for the enabled process to occur. The enabled process does not transform the enabling object. An *event link* connects a triggering entity (object, process, or state) with a process that it invokes. The event types that OPM supports include state entrance, state change, state timeout, process termination, process timeout, reaction timeout, and external events. External events include clock events and triggering by environmental entities such as a user or an external device.

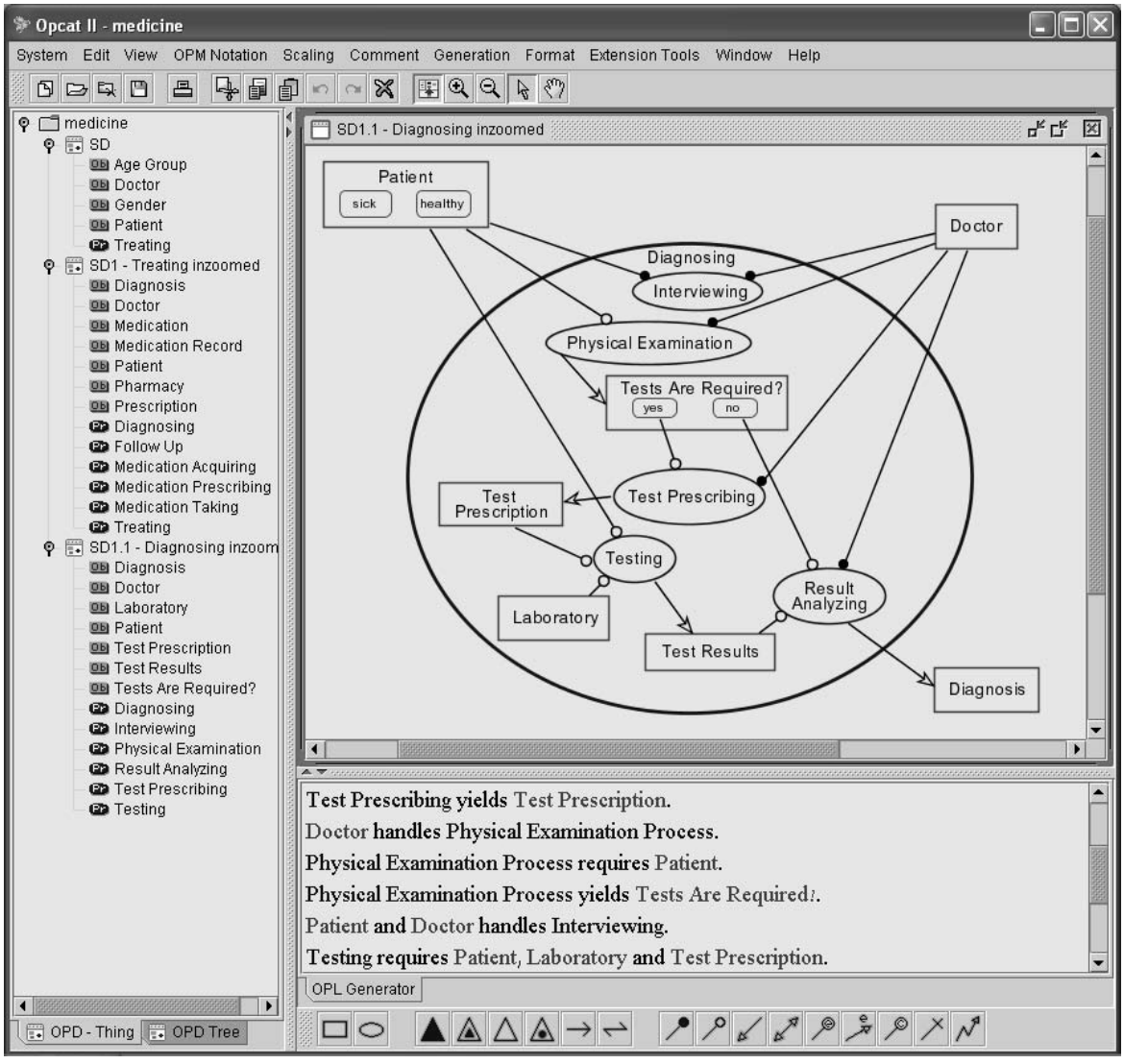


Fig. 6. A snapshot of OPCAT 2 GUI showing the OPD window (*top*), part of the corresponding OPL window (*bottom*), and the OPD tree (*left*)

The bimodal graphic-text representation of OPM

Two semantically equivalent modalities, one graphic and the other textual, jointly express the same OPM model. A set of interrelated Object-Process Diagrams (OPDs) showing portions of the system at various levels of detail constitute the graphical, visual OPM formalism. Each OPM element is denoted in an OPD by a symbol, and the OPD syntax specifies correct and consistent ways in which entities can be connected via structural and procedural links, each having its specific, unambiguous semantics. OPM assigns special graphical symbols for a selected set of relations (similar to UML class diagrams, only for a larger set of relations).

OPCAT (Object-Process CASE Tool) [16] is a Java-based software environment that supports OPM system modeling and evolution. As shown in the toolbox at the bottom of OPCAT's GUI in Fig. 6, a triangular graphical symbol along the line connecting two things (objects or processes) is assigned to each of the four fundamental structural relations mentioned above, whose symbols are black triangle for aggregation, white triangle, as in UML, for generalization, black on white triangle for characterization, and black circle in white triangle for instantiation. Like associations in UML class diagrams, other structural relations become textual tags (labels) recorded along the arrow connecting the two entities such that concatenating the source entity with the tag with the destination entity yields a meaningful sentence.

In principle, all relations could be expressed textually, requiring no special symbol. For example, instead of the white triangle along the arrow pointing from the generalizing thing to the specialized one, the text would read "generalizes" (or "is-a" if the direction of the arrow is reversed). The same applies to assigning symbols to a small set of procedural relations, such as the agent and instrument link. The motivation for defining this small set of graphic symbols is that they help eliminate much of the text that might otherwise clutter the diagram. Choosing exactly this particular link set is rooted in the observation that the relations these links symbolize occur frequently in models, so denoting them by special symbols makes the diagram more visually expressive and less loaded

with text strings. It reflects an attempt to strike a balance between the tendency to reduce the symbol set on the one hand and to enhance the visual appearance of the diagrams on the other.

The Object-Process Language (OPL), defined by a context-free grammar, is the textual counterpart modality of the graphical OPD set. OPL is a dual-purpose language oriented toward humans as well as machines. Catering to human needs, OPL is designed as a constrained subset of English, which serves domain experts and system architects, jointly engaged in analyzing and designing a system, such as an electronic commerce system or a Web-based enterprise resource planning system. Every OPD construct is expressed by a semantically equivalent OPL sentence or phrase. This dual representation of OPM increases the processing capability of humans according to the cognitive theory of multimodal learning proposed by Mayer [23]. Catering to the modality principle of this cognitive theory, OPM enables modeling systems both graphically and textually.

Figure 6 is a snapshot of OPCAT's GUI. OPCAT translates on the fly each OPD construct into its equivalent OPL sentence, yielding an OPL paragraph, a collection of OPL sentences that specify the knowledge represented graphically in the OPD. Conversely, typing an OPL sentence complements the OPD with the text added to the OPL paragraph such that at any point in time the graphic and textual representations are completely equivalent and reconstructible from each other. Using OPCAT, users can model complex systems and express and query knowledge. The example in Fig. 6 deals with medicine and the particular OPD zooms into the diagnosing process. The GUI of OPCAT in Fig. 6 shows the hierarchy of diagrams and things (i.e., objects and processes) on the left, the graphic (OPD) window at the top right, the text (OPL) window at the bottom right, and the palette with the various OPM entities and links at the bottom. Some of the OPL sentences, all of which were generated automatically by OPCAT, are shown in the OPL window. To get the feeling for their closeness to natural English, some of these sentences are listed below. To the right of each OPL sentence is its type.

Patient can be **healthy** or **sick.**	*(State enumeration sentence)*
Doctor handles **Test Prescribing.**	*(Agent sentence)*
Test Prescribing yields **Test Prescription.**	*(Result sentence)*
Doctor handles **Physical Examination** Process**.**	*(Agent sentence)*
Physical Examination Process requires **Patient.**	*(Instrument sentence)*
Patient and **Doctor** handle **Interviewing.**	*(Agent sentence)*
Testing requires **Patient, Laboratory,** and **Test Prescription.**	*(Instrument sentence)*
Testing yields **Test Results.**	*(Result sentence)*
Doctor handles **Result Analyzing.**	*(Agent sentence)*

These sentences show how knowledge that combines structure and behavior, including state transitions, can be represented in both intuitive (yet formal) graphics and human-intelligible text. Users who are not familiar with the graphic notation of OPM can verify their specifications by inspecting the OPL sentences. For example, comparing the OPL sentence "**Doctor** handles **Test Prescribing**." with the relevant part of the OPD shows that the link from the object **Doctor** to the process **Test Prescribing**, which gives rise to the OPL reserved word handles, is the agent link, which originates from the agent (**Doctor**) and points to the destination process (**Test Prescription**). States are situations in which an object can exist. For example, **healthy** and **sick** are situations of **Patient**, as expressed by the OPL state enumeration sentence "**Patient** can be **healthy** or **sick**."

As this example shows, the knowledge that OPM can represent is not restricted to just structural knowledge, as in CGs and the other knowledge representation formats described above. It can also be procedural, showing temporal order and enabling cause-and-effect analysis. Designed also for machine interpretation through a well-defined set of production rules, OPL provides a solid basis for automating the generation of the designed application. Indeed, OPCAT currently generates solid skeleton Java code from OPL script and enables the generation of any other formal language.

The OPM text-graphic equivalence principle

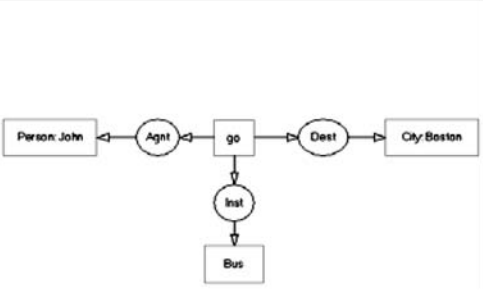
Unlike the graphic representation in the Semantic Web, which is an auxiliary means of illustrating the machine-oriented, XML-based content, OPDs constitute a complete and consistent visual formalism that goes hand in hand with OPL. A basic OPM principle, demonstrated in Fig. 6, is the *text-graphic equivalence principle*:

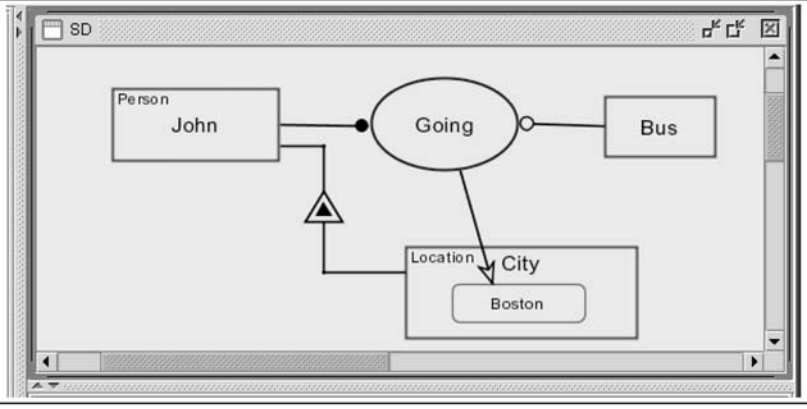
> *Anything that is expressed graphically by an OPD is also expressed textually in the corresponding OPL paragraph, and vice versa.*

Following this principle, our goal in developing ViSWeb, the Visual Semantic Web, as in OPM in general, is to specify a system by a set of interrelated Object-Process Diagrams and their completely equivalent corresponding OPL paragraphs. This equivalence implies that both modalities, the graphic and the textual, contain exactly the same information, albeit in two different forms of expression. Due to this complete equivalence, each can be reconstructed from the other. As noted, in spite of the apparent graphics-text redundancy, from a human factors engineering viewpoint, these two modalities activate different cognitive processes and therefore reinforce the understanding of each other and of the system as a whole.

The option of choosing between natural language text and graphics to specify some modeling artifacts and alternating between the two at the user's discretion is a unique feature of OPM and OPCAT. The relatively small set of OPD symbols and corresponding OPL sentence types increases the accessibility of OPM to both system architects and domain experts. System architects who are familiar with OPD syntax can use the visual OPD symbols, while domain experts can read the English-like OPL script to understand the specification and verify that the system is designed to meet their requirements. The automatic translation into an OPL script also improves the documentation quality of the developed system. The automatic implementation (code and database schema) generation ensures that the specification designed by the system archi-

Table 1. Comparison between the CG of Fig. 3 (*left*) and the OPM model (*right*) of "*John is going to Boston by bus.*"






```
[Go]-
(Agnt)->[Person: John]
(Dest)->[City: Boston]
(Inst)->[Bus].
```

The **Person John** exhibits the **Location City.**
City is **Boston.**
John handles **Going.**
Going requires **Bus.**
Going changes **City** to **Boston.**

tects and endorsed by the domain experts is indeed reflected without any translational gap in the actual system.

OPM scalability and complexity management

A major problem with most graphic modeling approaches is their scalability: as system complexity increases, the graphic model becomes loaded with shapes and cluttered with links that cross each other in all directions. The limited channel capacity [23] is a cognitive principle that states that there is an upper limit on the amount of detail a human can process before being overwhelmed. This principle is addressed by OPM and implemented in OPCAT with three abstraction/refinement mechanisms. These enable complexity management by providing for the creation of interrelated OPDs (along with their corresponding OPL paragraphs) that are limited in size, thereby avoiding information overload and enabling comfortable human processing. The three refinement/abstraction mechanisms are: (1) *unfolding/folding*, which is used for refining/abstracting the structural hierarchy of a thing and is applied by default to objects; (2) *in-zooming/out-zooming*, which exposes/hides the inner details of a thing within its frame and is applied primarily to processes; and (3) *state expressing/suppressing*, which exposes/hides the states of an object. Using flexible combinations of these three abstraction/refinement mechanisms, OPM enables specifying a system to any desired level of detail without losing legibility and comprehension of the resulting specification. The complete OPM system specification is expressed graphically by the resulting set of consistent, interrelated OPDs and textually by the corresponding OPL script, which is the union of the information expressed in the OPL paragraphs. Like OPD, each OPL paragraph is a collection of sentences that span no more than a single page, enabling humans to comfortably read and digest the knowledge it expresses.

3 Concept graphs vs. object-process diagrams

The dual graphic and equivalent natural language representation of the single OPM model is both human understandable and machine processable. The modeling system that comes closest to OPM in its dual graphic-textual representation is conceptual graphs (CGs). Having introduced both CGs and OPM, we are now in a position to compare these two bimodal graphic-textual approaches to knowledge representation and their adequacy as a basis for visually and textually modeling the Semantic Web.

We start by showing the OPM counterparts of the three CG examples presented in Sect. 2. Table 1 compares both the graphic and the textual CG and OPM model representations of the system represented by the natural language sentence "*John is going to Boston by bus*." Graphically, the CG model has a compact set of symbols: boxes for concepts, ovals for relations, and arrows for the directed links between concepts and relations. OPM has a richer set of symbols, which allows it to be more expressive. While OPDs use boxes and ovals, like CGs, their semantics is different, denoting, respectively, objects and processes rather than CG concepts and relations. OPM relations are expressed via the various link types. For example, the link from **John** to **Going**, which ends with the black circle, is the agent link, denoting that the **Person** called **John** is the agent of (the human who executes) the process **Going**. Similarly, the link from **Bus** to **Going**, which ends with the white circle, is the instrument link, denoting that the **Bus** is the instrument of the process **Going**. As noted earlier, these special symbols obviate the need to annotate these links textually. Agent and instrument links are procedural links: they connect an object and a process.

Other OPD procedural links are the result, consumption, and effect links. In addition to procedural links, OPDs feature a family of structural links, each of which connects two objects. An example of a structural link in the OPD in Table 1 is the exhibition-characterization relation, denoted by the black-

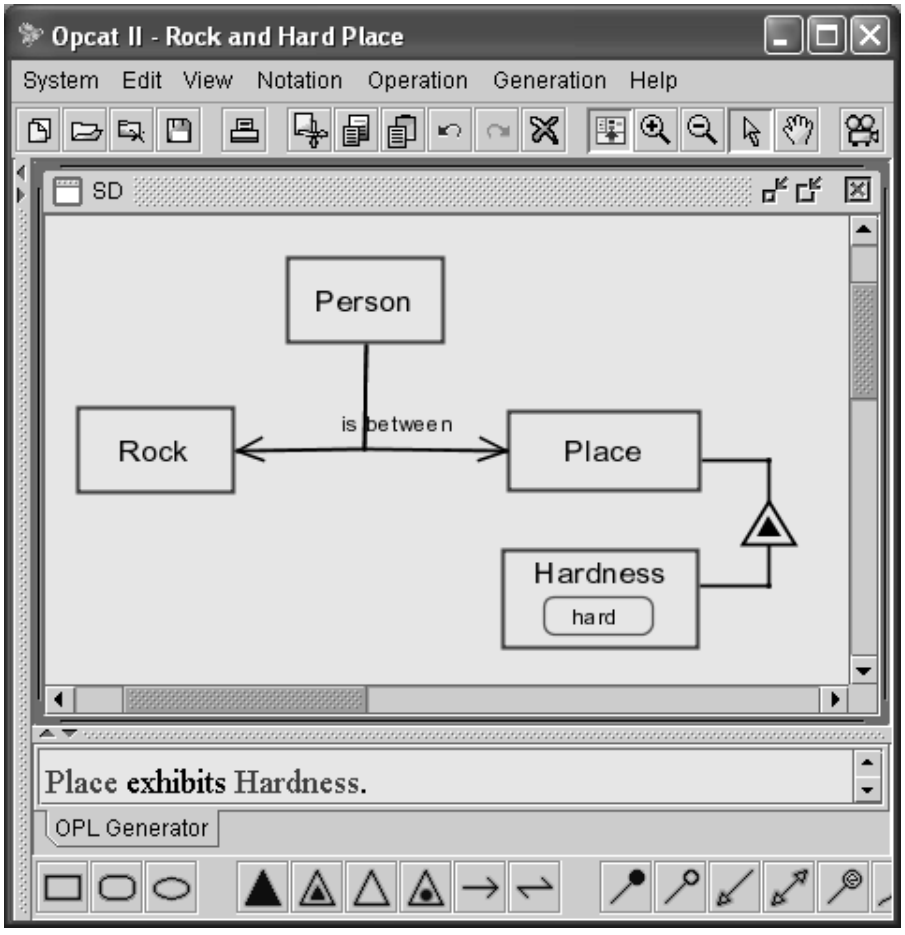


Fig. 7. OPD (*left*) and the OPL paragraph (*right*) of the system "*Person is between a rock and a hard place*" of the CG in Fig. 4

in-white triangle along the line connecting the **Person John** to the **City**. The exhibition-characterization symbol from **Person** to **Location** states that **Location** is an attribute of **Person**.

The CG makes no underlying semantic distinction between "`John`", "`Boston`", and "`Bus`" on the one hand and "`Go`" on the other: all are concepts. In OPM there is a principal difference between these two entity types: the OPM ontology stipulates that objects are things that exist, while processes are things that transform objects by changing their state or by generating/consuming them. The inability of CGs to distinguish between objects and processes is a major hindrance to enhanced expressive power with respect to system dynamics: CGs may be fine for declarative assertions, i.e., statements about what exists in the world and how what exists relates to other things that exist. However, when it comes to describing the dynamics of the system, namely, its time-dependent behavior, CGs lacks the basic concept of process and makes no distinction between objects and processes, relating to all as concepts. A somewhat similar difference exists between OPM and the OO paradigm, in which Object is the only top-level concept and processes can only be expressed as operations that objects own.

A comparison of the two textual representations in Table 1 reveals that, while CGs may bear direct mapping to language, they do not translate to any subset of natural language but rather to a symbolic representation. The OPL sentences, on the other hand, are understandable, and the OPL paragraph above can indeed be summarized by the original sentence. Note that the OPL script unfolds a small five-sentence "story". To set up the framework for this story, **John** was assigned the attribute **City**, which is an instance of the class **Location** and is assigned the value **Boston**. OPL sentences are written in plain English that is easily readable and understandable to humans with no prior training whatsoever. The same cannot be said about the LF script of CGs, as demonstrated in the bottom left of Table 1. Another quantum leap is still required to convert this script to a natural language sentence like "*John is going to Boston by bus.*"

Moving on to the next example, Fig. 7 shows the OPD of the system "*Person is between a rock and a hard place*" of the CG in Fig. 4 and its corresponding OPL paragraph. Note that even though this system is purely declarative, static, and with no dynamic element, the three-sentence OPL paragraph clearly conveys the semantics of the system in plain English. Like the previous example, **Hardness** is set as an attribute of **Place**, with **hard** being its value. The structural relation "is between" is a fork relation [14] whose handle is linked to **Person**, and two teeth are linked to **Rock** and **Place**.

The third example concerns the OPM model in Fig. 8 of the system "*Tom believes that Mary wants to marry a sailor*", which appeared in the CG in Fig. 5.

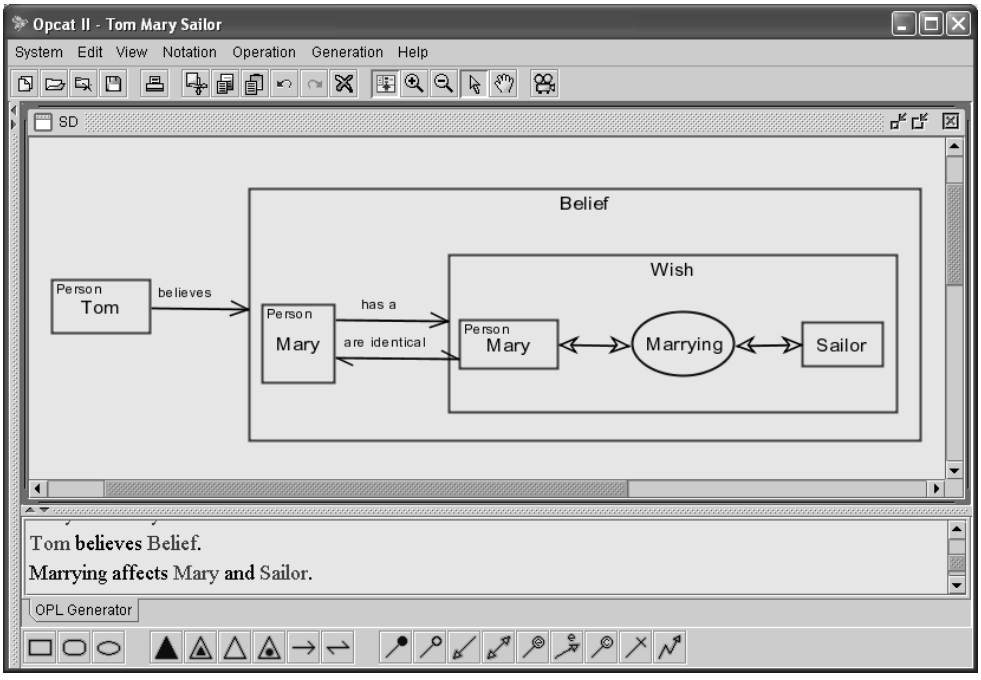


Fig. 8. An OPD of the system "*Tom believes that Mary wants to marry a sailor*" of the CG in Fig. 5

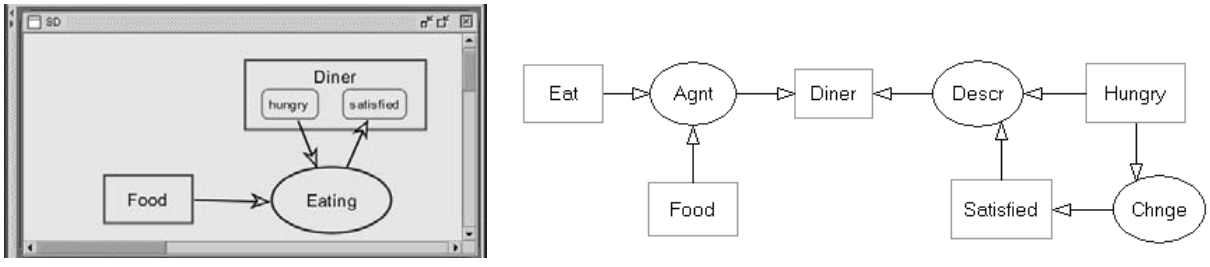


Fig. 9. An OPD (*left*) and a CG (*right*) of the system "*The food that the hungry diner eats makes him satisfied*."

This example includes CG contexts. The OPM analog is the embedded objects, which define scopes. The OPL of this system is written below.

The **Person Tom believes Belief**.
Belief zooms into the **Person Mary** and **Wish**.
Mary has a Wish.
Wish zooms into the **Person Mary** and **Sailor**, as well as **Marrying**.
Mary of **Belief** and **Mary** of **Wish are identical**.
Marrying affects **Mary** and **Sailor**.

This syntax makes use of the scaling options of OPM as analogs of the context of CGs. In this case, the in-zooming of objects was utilized as a "container" for nesting the parts of **Belief** and the parts of **Wish** within **Belief**.

While the three examples above were done originally in CG and translated to OPM, let us now consider a reverse example. Both the OPD and the CG in Fig. 9 describe the system "The food that the hungry diner eats makes him satisfied." The description requires 16 symbols in CG and only 8 in OPD, reflecting the 2:1 ratio stated in the theorem below.

The three OPL sentences making up the paragraph of the OPD in Fig. 9 are:

Diner can be **hungry** or **satisfied**.
Eating consumes **Food**.
Eating changes **Diner** from **hungry** to **satisfied**.

Examining either the OPD or the OPL paragraph one can tell that **hungry** and **satisfied** are states of **Diner**, since they are embedded in it, and that the **Eating** process consumes the **Food** and makes the **hungry Diner satisfied**. All this is much less obvious in the CG. The LF offers no extra help either, while the OPL does provide clarifications to those not versed in the OPD symbols.

Text in CGs and OPDs

The objective of the textual modality in both CG and OPM is to be immediately human comprehensible. Being as close as possible to natural language guarantees such comprehension. Comparing the text of OPM with that of CG reveals a major difference at the semantic level. While the OPL sentence "**Making** yields **Car**" is a legal English sentence that makes perfect sense, the LF expression "`[Make]->(Rslt)->[Car]`" is cryptic and makes very little sense to humans not versed in the idiosyncrasies of CGs. Another major translation phase is required to convert this mechanical text to one that can be readily understood by untrained humans.

Table 2. Comparison between CGs and dynamic OPDs as two types of bipartite graphs

Bipartite graph type	Node type I	Node type II	Edge
Conceptual Graph (CG)	Concept	Relation	Between Concept and Relation
	Diagram: Car ← Rslt ← Make Text: `[Make]->(Rslt)->[Car].`		
Dynamic OPD	Object	Process	Between Object and Process, which are CG concepts
	Diagram: Car ← Making Text: Making yields Car.		

Comparing the size of CGs and dynamic OPDs as bipartite graphs

As Table 2 shows, there is an interesting syntactic analogy between CGs and the dynamic OPDs (i.e., OPDs that involve only procedural links): both yield bipartite graphs whose two node types are denoted by boxes (rectangles) and ovals (ellipses). However, contemplating the meanings conveyed by these two modeling paradigms, one observes a principal semantic difference. In terms of CGs, both "Object" and "Process" of OPM, e.g., **Car** and **Making**, are the CG concepts `[Car]` and `[Make]`. The "Relation" `(Rslt)` in CGs is a node type, like "Concept", and these two node types are linked by edges (arrows). In dynamic OPDs, however, the semantics of relations are embedded in the graph's edges, such as the result link from **Car** to **Making**, while Object and Process are the two node types. In CG terms, on the other hand, both Object and Process are concepts, i.e., nodes of the same type.

CG edges have no counterpart OPDs; they are the (invisible) connecting points between the Object or Process and the procedural link. In OPDs, relations are the links, or graph edges. Conveying the same semantics in CGs therefore requires significantly more graphic symbols than OPDs. Consider, for example, expressing the fact that "*Bus is an instrument of Going*." In CG we need the concepts `[Go]` and `[Bus]` and the relation `(Inst)`. We also need two edges, one between `[Go]` and `(Inst)` and the other between `(Inst)` and `[Bus]`, a total of five graphic symbols. In OPD, the same semantics is conveyed by three symbols: the object **Bus**, the process **Going**, and the instrument link from **Bus** to **Going**.

For the above toy examples, the ratio of CG to OPD symbol numbers is about 2:1. As the complexity of the system grows, this ratio is at least 2:1 because in a connected bipartite graph each node of one type must be connected to at least two nodes of the other type. The number of symbols required to express some complex semantics in a CG is therefore at least double the number of symbols in an OPD. More formally, we state and prove the following CG to OPD symbol ratio theorem:

Theorem: *As the complexity of a system S grows, the ratio between the minimal number of symbols required to model S in a CG and the same number in a dynamic OPD approaches 2.*

Proof: Let O and P be the respective number of objects and processes in a dynamic OPD. Let C and R be the number of concepts and relations in a CG. The minimum number of links required for an OPD to be a connected bipartite graph is $L'_{\min} = O + P - 1$. The total number of OPD symbols is $Y_{\mathrm{OPD}} = L_{\min} + O + P - 1 = 2(O + P) - 1$. In a CG, both objects and processes are concepts, so $C = O + P$. Since each OPD link is a relation node in a CG, $R = L'_{\min}$. The total number of CG nodes is $C + R = O + P + L'_{\min}$, and the minimum number of CG links is $L_{\min} = C + R - 1 = O + P + L'_{\min} - 1$. The total number of CG symbols is $Y_{\mathrm{CG}} = C + R + L_{\min} = 2(C + R) - 1 = 2(O + P + L'_{\min}) - 1 = 2(O + P + O + P - 1) - 1 = 4(O + P) - 3$. The ratio $Y_{\mathrm{CG}}/Y_{\mathrm{OPD}}$ is:

$$\frac{Y_{\mathrm{CG}}}{Y_{\mathrm{OPD}}} = \frac{4(O+P)-3}{2(O+P)-1} .$$

The limit when either the number of objects or processes in the system approaches infinity is

$$\lim_{O,P\to\infty} \frac{Y_{\mathrm{CG}}}{Y_{\mathrm{OPD}}} = 2 , \quad \text{QED.}$$

The difference in favor of OPD is not just in the fact that the OPD is half the size of the CG of the same system, but more importantly that the appeal of the OPD graphic paradigm to human intuition is higher because relations are expressed as links between objects and processes (which are the edges between the nodes in the OPD graph) rather than another type of node in the CG graph, requiring edges that lack semantics or textual interpretation.

Summary of CG-OPM differences

Summarizing the main differences between CGs and OPM we note that:

(1) The symbol set of CGs is more compact than that of OPDs, but expressing the same complex semantics in CGs requires using at least twice the number of symbols required in OPD, yet the semantics is more explicit in OPDs.
(2) The CG formalism is probably better than OPM with respect to support for logic. While OPM does allow AND, OR, and XOR relations, as well as Boolean objects, it currently does not have the notion of quantifiers and cannot deduce new knowledge from existing knowledge. This is an issue being considered for inclusion in the next OPM versions.
(3) The text generated by OPM, the OPL paragraph, is a subset of English, enabling any English speaker to readily understand it, while the LF, the textual form of CGs, is still in symbolic form that is not legible to untrained humans.
(4) CGs are purely declarative and have no notion of system dynamics, which is a major feature of OPM.
(5) OPDs cater to complexity management by the ability of things to undergo refinement (in-zooming mainly in processes and unfolding mainly in objects) in new, descendant OPDs, such that no single OPD is too cluttered and there is no limit to the number of detail levels. These complexity management mechanisms provide for modeling systems of any size. No such mechanism exists in CGs.

In view of these fundamental differences, the choice of OPM as a basis for the Visual Semantic Web is quite obvious. We continue with a brief survey of RDF and the use of graphics in the Semantic Web.

4 The Semantic Web and the RDF syntax

RDF, the Resource Description Framework [8,20], aims at making the knowledge resources that are available on the Web amenable to machine interpretation, compilation, or other types of processing by imposing some structure on the pieces of knowledge. RDF provides a basis for a number of emerging initiatives, such as the Dublin Core Metadata Initiative [17], an open forum engaged in the development of interoperable online metadata standards. As noted, one must bear in mind that machines are never going to understand knowledge the way humans do. At best, they can exhibit treatment of and response to this knowledge, which would seem to humans as if they "understand" it.

The RDF Syntax document [20] introduces a model for representing RDF metadata as well as a syntax for encoding and transporting these metadata for interoperability of independently developed Web servers and clients. The syntax it presents uses the eXtensible Markup Language (XML) because one of the goals of RDF is to enable specifying semantics for data based on XML in a standardized manner. RDF and XML are complementary in that RDF is a model of metadata and only addresses encoding issues by reference. Such issues include internationalization and character sets, required by transportation and file storage. More importantly, the XML syntax of RDF is only one of the possibilities for encoding RDF and, as noted in [20], *alternate ways of representing the same RDF data model may emerge*. Indeed, this paper proposes an OPM-based alternative on top of the XML syntax that is human- and machine-oriented at the same time. This syntax enables bimodal, dual graphic-textual representation of the system model for human consumption while possessing a level of formality that makes it amenable to machine processing.

Use of graphics in the Semantic Web

The Semantic Web makes only limited use of graphical models. In RDF these are directed graphs, where subjects and objects are nodes and predicates are labels along the edges, directed from a subject to an object. However, since the Semantic Web in its current form and philosophy is geared primarily to cater to the needs of machines, and since machines do not need to read diagrams, the visual aspect of the information and knowledge modeling is not well developed. Semantic Web documents show few graphs early on but then abandon them and focus on the XML-based syntactical aspects of the machine-oriented language.

RDF basics and example

RDF is a model for representing named properties and property values [20]. RDF properties may be thought of as attributes of resources and, in this sense, correspond to traditional attribute-value pairs. RDF properties represent relationships between resources, and RDF Schemas, which are instances of RDF data models, are entity-relationship (ER) diagrams. The basic RDF data model consists of the following three object types:

Resource: Anything described by an RDF expression. A resource may be an entire Web site, a Web page, a part of a Web page, and any object that is not directly accessible via the Web, e.g., a printed book. Resources are always named by URIs and anything imaginable can be identified by a URI.
Property: A specific aspect, characteristic, attribute, or relation used to describe a resource. Each property has a specific meaning, which defines its permitted values, the types of resources it can describe, and its relationship with other properties.
Statement: A specific resource, the *subject*, together with a named property, the *predicate*, plus the value of that property for that resource, the *object*. The object of a statement (i.e., the property value) can be another resource, a literal, i.e., a resource (specified by a URI), a simple string, or another primitive datatype defined by XML.

To be able to talk concretely about RDF and its concepts, let us consider a few examples with increasing complexity. Following [8], consider first the sentence "*Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.*"

Translated to RDF format, this sentence can be interpreted as having the subject (resource) http://www.w3.org/Home/Lassila, the predicate (property) creator, and the object (literal) "Ora Lassila". As noted, RDF uses directed graphs to specify these notions graphically,

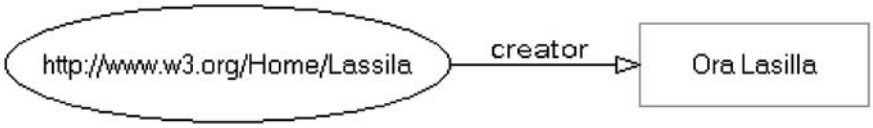


Fig. 10. A simple RDF graph example from [21]

Fig. 11. The RDF graph of the data model listed in Table 3 generated automatically by the RDF/XML Validation Service

where subjects and objects are nodes and predicates are labels along the edges, which are always directed from a subject to an object, as in Fig. 10. A resource node in the graph is drawn as an oval (ellipse), while a literal node is drawn as a rectangle.

As noted in [8], the graph in Fig. 10 is to be interpreted as "*http://www.w3.org/Home/Lassila has creator Ora Lassila*", and in general "*<subject> HAS <predicate><object>*". Interestingly, though, applying the RDF/XML Validation Service [37] using the RDF/XML script in Table 3 yields the graph in Fig. 11.

According to the W3C RDF Schema document [3], "*The RDF Schema class and property system is similar to the type systems of object-oriented programming languages such as Java. However, RDF differs from many such systems in that, instead of defining a class in terms of the properties its instances may have, an RDF schema will define properties in terms of the classes of resource to which they apply. This is the role of the rdfs:domain and rdfs:range constraints... For example, we could define the author property to have a domain of Book and a range of Literal, whereas a classical OO system might typically define a class Book with an attribute called author of type Literal*" [8]. In this regard, the Semantic Web is based on the same principle of OPM, where relations (called properties in the SW nomenclature) are edges of a graph rather than nodes, as in CGs.

Brickley and Guha [8] go on to say that one benefit of the RDF property-centric approach is that it is very easy for anyone to say anything they want about existing resources, indicating that this is one of the architectural principles of the Web expressed by Berners-Lee [3]. Not everything that is true about things that exist in the world needs to be specified a priori at the time of definition. A basic definition is sufficient, and it can be later augmented by other people linking it to other instances of `rdfs:domain`.

ViSWeb: an OPM-based Visual Semantic Web Spec alternative

The Visual Semantic Web [16] (ViSWeb) alternative to the RDF/XML knowledge representation takes advantage of the integrated graphic-text formal yet intuitive infrastructure that OPM provides. Figure 12 is a ViSWeb spec (Visual Semantic Web specification) that expresses the example in Fig. 10 in a bimodal fashion, both as an Object-Process Diagram (OPD) and an Object-Process Language (OPL) text. The OPD contains two object instances: **Ora Lasilla** and WWW.w3.org/Home/Lassila. To conform to OMG UML 1.4 [27], object names (i.e., instances of object classes) in OPDs are underlined, as in UML object diagrams.

Before discussing the ViSWeb extension of OPM, we address a few "syntactic sugar" issues. These syntactic sugar enhancements help the human reader to more easily digest OPL sentences, which are a subset of English anyway. As a subset of English, OPL allows spaces in the names of entities and also uses reserved phrases that contain more than one word. The distinction between bold and nonbold fonts is therefore helpful for human understanding. As the bottom window in Fig. 6 shows, OPCAT also provides the human reader with color cues such that the text matches the graphics: names of objects are green, processes blue, and reserved phrases (or structural link tags) black. OPCAT allows these colors as well as fonts to be configured by the user. In black and white text, OPL reserved phrases (collections of one or more words, such as requires or consists of) are denoted by nonbold Arial font, while names of entities (objects, processes, and states), which are nonreserved phrases, are written in **bold Arial font**. This is so because the interesting, domain-specific content lies in the nonreserved phrases rather than in the reserved ones. This is unlike most programming language editors, in which the reserved tokens are bold. Consider, for example, the following OPL sentence taken from Fig. 6: "**Testing** requires **Patient, Laboratory,** and **Test Prescription.**" The reserved phrases here are "requires" and "and", while the nonreserved phrases are "**Testing**", "**Patient**", "**Laboratory**", and "**Test Prescription**".

For standard, fast communication among nodes in a computer network, and especially over the Internet, it is convenient to have the files in a plain (rather than rich) text format (i.e., ASCII, as is the case with XML files). In this plain text version, which is geared toward quick and easy machine parsing, each entity name (nonreserved phrase) is enclosed within a pair of single quotes. For example, in this plain text, or 'quoted format', as we shall call it, the above OPL sentence would be:

```
'Testing' requires 'Patient',
'Laboratory', and 'Test Prescription'.
```

The quotes help a parser distinguish, for example, between the word `the` as a reserved word (when it is not within quotes) and the word `the` as part of the tag (relation name) '`is the creator of`', when it is within quotes. Quotes in a quoted format can be easily turned on (expressed) or off (suppressed) by a supporting CASE tool such as OPCAT. Since human reading is context sensitive, these quotes are quite annoying to people, who need them only very rarely to remove ambiguity. If in doubt, humans can get cues as to where the name

Table 3. The RDF/XML script that generates the graph in Fig. 11

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

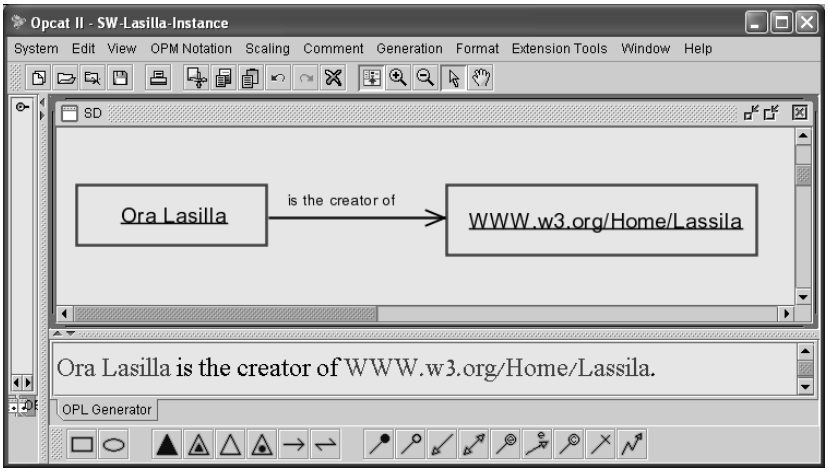


Fig. 12. The example in Fig. 10 expressed as a ViSWeb spec (Visual Semantic Web specification), consisting of an Object-Process Diagram (OPD) at the top window and its corresponding, automatically generated Object-Process Language (OPL) one-sentence paragraph at the bottom window

of a class, instance, or relation starts and ends based on the OPM convention that each word in the name of an (object or process) class and instance is capitalized, while names of relations (and states) are not. If this does not resolve the ambiguity, the human reader can consult the corresponding OPD, which will always disambiguate the sentence since there each name appears separately in its box or ellipse. Therefore, for human consumption quotes are suppressed and the quote-free format is used.

URLs and email addresses in OPL sentences are marked by underlined nonbold Arial font. This underlining conforms to the custom underlining in Web pages. The string WWW at the beginning of a URL is expanded to http://www, which current Web browsers do anyway.

Tagged structural links

A tagged structural link, depicted as an open arrow, such as the one pointing from **Person** to **URI** in Fig. 13, expresses the nature of the relation between these two objects. The tag is the text recorded along the structural link. The value of this tag is '**is the creator of**'. The value is a phrase, i.e., a collection of one or more words (separated by spaces, as in natural languages), such that when the name of the source object **Ora Lasilla** (an instance of the class **Person**) is concatenated with the tag value (i.e., the phrase) '**is the creator of**' followed by the name (value) of the **URL**, one automatically obtains the following OPL sentence, which is also generated automatically by OPCAT and recorded at the bottom of the OPD in Fig. 12:

Ora Lasilla is the creator of WWW.w3.org/Home/Lassila.

The automatic generation of the OPL sentence in this simple case was done by concatenating the name of the object at the source of the tagged structural link, **Ora Lasilla**, with the text string of the structural link's tag, **is the creator of**, with the name of the destination object, WWW.w3.org/Home/Lassila.

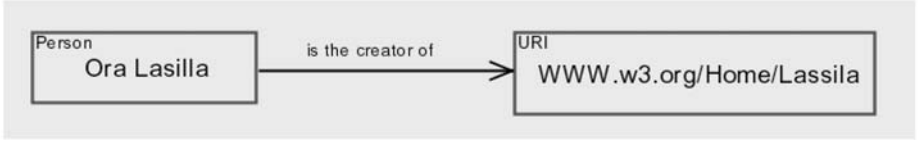
In RDF terminology, this OPL sentence is a *statement* that contains a specific resource – the *subject*, **Ora Lasilla** in our case – together with a named property – the *predicate*, '**is the creator of**' in our case – plus the value of that property for that resource – the *object*, WWW.w3.org/Home/Lassila in our case. Each word in an object (and process) name is capitalized, while in link names (tags) they are not. As Fig. 14 shows, names of objects and link names (tags) appear in different colors (which can be set by the user). Even though there are spaces between the words, using the capitalization rule above it is possible to mechanically parse the sentence even without the human-oriented color cues.

Table 4 compares RDF and OPM with respect to this example. For each method, the three elements and the respective parts of the example are written first, and below them are the graphical and textual representations of the RDF graph in Fig. 10 and the OPD in Fig. 14.

While the OPM model still does not account for namespaces, which are treated below, comparing this OPL/ViSWeb sentence to the RDF/XML script in Table 4 it is not difficult to see the benefit of using a more human-readable version, which, while still machine-readable, does not require the human reader to act like a mechanical XML parser.

Table 4. Comparison between RDF and OPM applied to the example in Figs. 10 and 14

<table>
<tr><td rowspan="2">RDF/XML</td><td>Object (domain):
Http://www.w3.org/Home/Lassila</td><td>Predicate (property):
Creator</td><td>Subject (range):
Ora Lassila</td></tr>
<tr><td colspan="3"><u>Graphics:</u>
 

<u>Text:</u>
<pre>
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
</pre></td></tr>
<tr><td rowspan="2">OPM/ViSWeb</td><td>Source object:
Ora Lasilla</td><td>Structural link tag:
is the creator of</td><td>Destination object:
WWW.w3.org/Home/Lassila</td></tr>
<tr><td colspan="3"><u>Graphics:</u>
 

<u>Text:</u>
Ora Lasilla is the creator of WWW.w3.org/Home/Lassila.</td></tr>
</table>

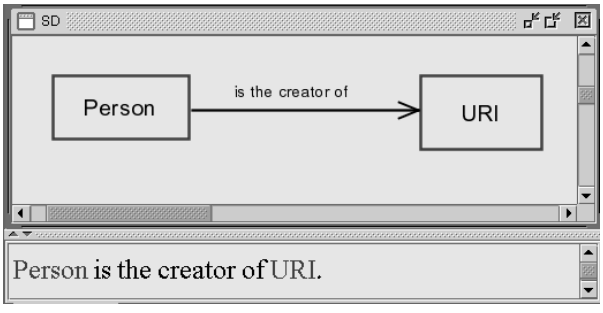


Fig. 13. A ViSWeb schema showing the class OPD along with its corresponding OPL sentence to which the ViSWeb spec in Fig. 12 conforms

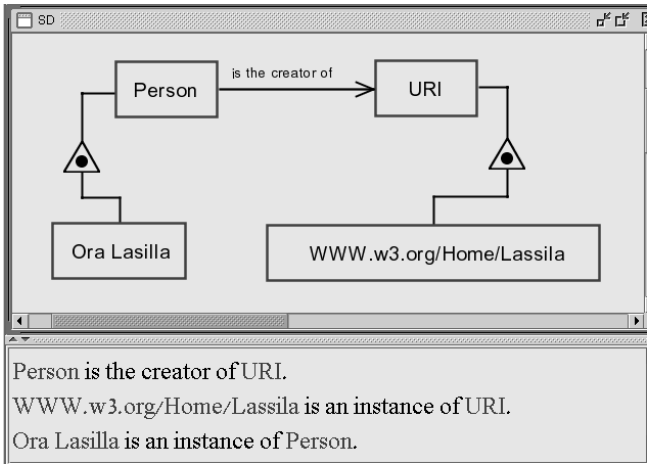


Fig. 14. The instantiated ViSWeb schema generated by adding the instance specification of Fig. 12 to the class information in the ViSWeb schema in Fig. 13

5 The ViSWeb schema: a template for a ViSWeb Spec

The lines under the two objects in Fig. 12 denote the fact that these are object instances, not object classes. The class information is still missing in this OPD.

Figure 13 shows a *ViSWeb schema*, an OPD-OPL template that contains class information. Note that the ViSWeb schema follows the OPM text-graphic equivalence principle: the OPD and the OPL paragraph are completely reconstructible from each other. Each ViSWeb spec conforms to a ViSWeb schema. Thus, the ViSWeb schema in Fig. 12 conforms to the ViSWeb spec in Fig. 13. This ViSWeb schema can be thought of as a

template that expresses a rule. In our example, the rule stipulates that the source (which in RDF schema terminology is termed the *domain*) of the relation (the RDF *predicate*) '**is the creator of**' is an object that belongs to the class **Person** and that the destination (*range*) of that relation is an object that belongs to the class **URI**.

Having established the **Person-URI** ViSWeb schema, we can now use it to add the object instance for each of the two classes. This is done in the instantiated schema shown in Fig. 14, where the ViSWeb schema of Fig. 13 and the ViSWeb spec of Fig. 12 are combined. The combination uses the OPM classification-instantiation relation, which is denoted as a bulleted triangle whose tip is linked to the class and whose base is linked to the instance. Note that the instances **Ora Lasilla** and WWW.w3.org/Home/Lasilla need not be underlined here to denote that they are instances. The underlining of the instance names is mandatory only if the class information is not present in the OPD, but here this is indicated by the classification-instantiation links from the classes to the respective instances.

The OPL paragraph of the OPD in Fig. 14 is shown in Fig. 14 as well:

Person is the creator of URI.
WWW.w3.org/Home/Lassila is an instance of **URI**.
Ora Lasilla is an instance of **Person**.

Note that predicates (such as **is the creator of**) do not have explicit instance names that are distinct from their class names. Thus, for example, we use the same predicate in the OPL sentence "**Mark Twain is the creator of Huckleberry Finn**" as in "**Ora Lasilla is the creator of** WWW.w3.org/Home/Lassila." The tagged structural relation '**is the creator of**' from the class **Person** to the class **URI** is inherited by their respective instances, so there is an implicit tagged structural relation with the same tag, '**is the creator of**', from **Ora Lasilla**, an instance of the class **Person**, to WWW.w3.org/Home/Lassila, an instance of the class **URI**. Applying template information and using chaining rules one can establish that **Ora Lasilla is the creator of** WWW.w3.org/Home/Lassila, although this is not explicit in the OPM model in Fig. 14. However, the instantiated ViSWeb schema in Fig. 14 has a couple of drawbacks. First, it is space consuming, and second, it requires the reader to realize the existence of the implicit tagged structural relation. These two problems are solved in the compact version of the instantiated ViSWeb schema of Fig. 14, shown in Fig. 15.

The OPL paragraph that corresponds to the OPD in Fig. 15 is also more compact than the three-sentence OPL paragraph of Fig. 14, as it consists of just one sentence:

The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila.

This sentence combines the OPL schema sentence from Fig. 12, which is "**Person is the creator of URI**" with the OPL instance sentence in Fig. 13, which is "**Ora Lasilla is the creator of** WWW.w3.org/Home/Lassila." In the new OPL sentence, which reflects both the classes and the instances, we added the class information of both **Ora Lasilla**, which is **Person**, and of WWW.w3.org/Home/Lassila, which is **URI**. **Ora Lasilla** is classified in the OPL sentence as belonging to the class **Person** by preceding the name of the instance by the reserved word "The" followed by the class name **Person**. Likewise, the string WWW.w3.org/Home/Lassila was classified as belonging to the class **URI** by preceding the value of the string by the reserved word "the" followed by the class name **URI**. The corresponding quoted sentence is:
`The 'Person' 'Ora Lasilla' 'is the creator of' the 'URI' 'WWW.w3.org/Home/Lassila'`. Theoretically, lacking quotes, the above OPL sentence contains a potential ambiguity for the human reader, who might think that the class name is '**Person Ora**', while the instance of that class is '**Lasilla**'. However, even in this case, a quick look at the corresponding OPD clarifies that '**Person**' and not '**Person Ora**' is the class name.

Recall that the OPM text-graphics equivalence principle mandates that any piece of information contained in the OPL paragraph be represented in the corresponding OPD, and vice versa. This principle, which makes the OPD and its OPL paragraph fully equivalent in terms of information content, is followed in the OPM specifications of both Fig. 14 and Fig. 15.

6 A taste of OPL parsing

In this section we demonstrate how, by using BNF production rule formulation of a tagged structural sentence and several nonterminals (OPL phrases), one can parse a given tagged structural sentence. The OPL sentence "The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila" makes use of a *class-specified instance phrase*, which is used twice. The first class-specified instance phrase is "The **Person Ora Lasilla**", and the second is "the **URI** WWW.w3.org/Home/Lassila". The BNF production rule of a class-specified instance phrase (where phrase is a *nonterminal* in formal language terminology) is

```
Csi ::= The | the
<class_name><instance_name>.
```

(Class-specified instance phrase).

We define the nonterminal `Class-or-Instance` as follows:

```
Class-or-Instance ::= Csi | class_name |
instance_name.
```

Now we can specify a tagged structural sentence as

```
Tagged_structural_sentence::=
Class-or-Instance tag
Class-or-Instance.
```

Using the above production rule, we can parse the sentence "The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila." as follows:

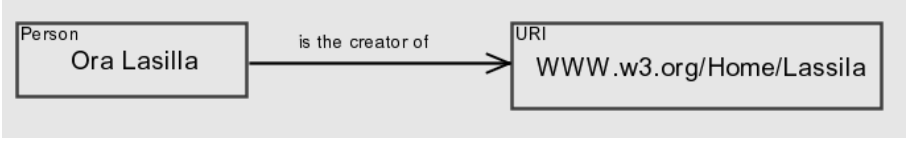


Fig. 15. A compact version of the instantiated ViSWeb schema in Fig. 14

```
Tagged_structural_sentence::=
Class-or-Instance tag
Class-or-Instance.
Tagged_structural_sentence::= Csi is the
creator of Csi.
Tagged_structural_sentence::= The
<class_name> <instance_name> is the
creator of the <class_name>
<instance_name>.
Tagged_structural_sentence::= The Person
Ora Lasilla is the creator of the URI
WWW.w3.org/Home/Lassila.
```

Note that this is a formulation of a context-free grammar. The implication of this is that we cannot enforce rules like "If the first `Class-or-Instance` in a `Tagged_structural_sentence` is `Csi`, then the second one must also be `Csi`." Therefore, sentences like

Ora Lasilla is the creator of the **URI** WWW.w3.org/Home/Lassila

and

The **Person Ora Lasilla is the creator of** WWW.w3.org/Home/Lassila

are also legal tagged structural sentences. But this is fine because in a larger system the fact that **Ora Lasilla** is a **Person** or WWW.w3.org/Home/Lassila is a **URI** may be specified elsewhere and we do not wish to repeat it each time the instance appears in an OPL sentence. This example demonstrates that although OPL is a subset of natural English, with a context-free grammar OPL sentences are amenable to parsing as a formal language, which is a most desirable attribute of a "machine-readable" language.

7 OPM namespace specification

Namespaces [7] are definitions of terms and relations of some domain ontology. The OPL sentence "The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila" does not specify the namespaces that contain the definitions of **Person**, **URI**, and the structural link tag (predicate) '**is the creator of**'. In contrast, the XML script in Table 4 does mention two namespaces, `rdf` and `docs`. The namespaces, which are part of the XML tags, enable us to know that we are looking at a `Description` in the sense defined in the `rdf` namespace definition, that the value of its `about` attribute is "`http://www.w3.org/Home/Lassila`", and that the value of the `Creator` entity as defined in the `docs` namespace is `Ora Lassila`. This information is clearly richer than what can be extracted from the RDF graph in Fig. 10 since that graph does not specify any namespace information.

As noted, the RDF graph is only an auxiliary means to make it easier for humans to "get the picture". It is not required to contain all the information expressed by the corresponding XML script and therefore cannot replace it (although the RDF Validator does so, albeit in a manner that is not very user friendly). The OPM text-graphics equivalence principle mandates that any piece of information contained in the OPL paragraph that corresponds to an OPD be represented in the OPD, and vice versa, making the OPD and its OPL paragraph fully equivalent in terms of information content. To keep up with the text-graphics equivalence principle, we introduce the concept of namespace to both the OPD and the OPL.

Let us assume that the subject **Person** and the object **URI** are both defined in the namespace whose name is **Semantic Web** and whose URI is WWW.SemanticWeb.org/definitions. We further assume that the predicate '**is the creator of**' is defined in the namespace whose name is **Documents** and whose URI is WWW.Documents.org/definitions. The OPD in Fig. 16 elaborates on that of Fig. 15, as it provides the complete namespace information. The ViSWeb convention is to stack all the namespaces used in the OPD at its top left corner. Each namespace is recorded in an object box, with the string "**Namespace:** `<blank><namespace_name>`" appearing at the top left corner of the box and the corresponding URI recorded at the bottom of the box. Here, `<namespace_name>` is the name of the namespace. Two namespace names appear in Fig. 16 – **Semantic Web** and **Documents**. Using these two namespace specifications, instances in an OPD can be annotated not just with the class specification, as in Fig. 15, but also with the namespace within which the class is specified, preceding the class name, as in Fig. 16. Thus, **Semantic Web: Person** is the complete namespace and class specification of the **Person** instance **Ora Lasilla**, and **Semantic Web: URI** is the complete namespace and class specification of the **URI** instance WWW.w3.org/Home/Lassila. Finally, **Documents:** is the complete namespace specification of the predicate (tagged structural link in OPM terminology) '**is the creator of**'.

The following two OPL namespace declaration sentences are the textual equivalents of the two namespace boxes stacked in the top left of Fig. 7.

The namespace **Semantic Web** is at URL WWW.SemanticWeb.org/definitions.
The namespace **Documents** is at URL WWW.Documents.org/definitions.

(Namespace declaration sentences)

Just as namespace graphical specifications in an OPD are part of the graphical syntax of ViSWeb, OPL namespace declaration sentences are part of the textual syntax of ViSWeb. The BNF production rule of a namespace declaration sentence is:

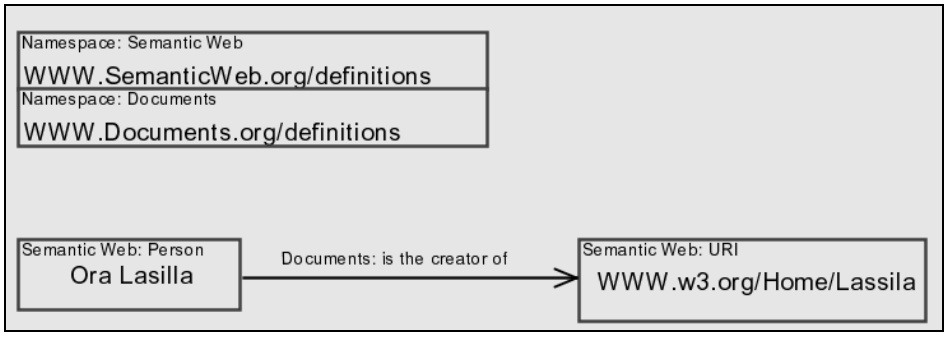


Fig. 16. The OPD of Fig. 14 with the namespace object boxes in the top left corner of the OPD

`Namespace_spec::=` The namespace `<namespace_name>` is at URL `<URL>`.

Based on the above two namespace declarations, the class and relation definition sentences that follow are:

The namespace **Semantic Web** defines the class **Person**.
The namespace **Semantic Web** defines the class **URL**.
(Class definition sentences)
The namespace **Documents** defines the relation '**is the creator of**'.
(Relation definition sentence)

The syntax of a class definition sentence is:

`Class_definition::=` The namespace `<namespaceName>` defines the class `<className>`.

Similarly, the syntax of a relation definition sentence is:

`Relation_definition::=` The namespace `<namespaceName>` defines the relation `<relationName>`.

Following OPL (and English) conventions, two or more class definition sentences with the same namespaceName can be joined. In our case, since both classes **Person** and **URL** are defined in the **Semantic Web** namespace, the two class definition sentences above are merged into the following sentence:

The namespace **Semantic Web** defines the classes **Person** and **URL**.

If there were more classes defined by the same namespace, they could be added to the same sentence as well, using a comma-separated list. The OPL syntax takes care of such comma-separated lists. For example, if **File** and **Image** were two additional classes defined by the **Semantic Web** namespace, then the above sentence would become

The namespace **Semantic Web** defines the classes **Person, URL, File,** and **Image.**

The default namespace convention

Usually, most if not all the names in a single OPD are defined in the same namespace. Thus, it is redundant and cumbersome to specify separately for each name that it is defined within that namespace. A simplifying OPM *default namespace convention* is that the namespace at the top of the namespace stack in the OPD is the default namespace, so any class in the OPD that is defined within this default namespace does not require that the namespace name precede it. In our example, the default namespace declaration sentence is

The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
(Default namespace declaration sentence)

The syntax of a default namespace declaration sentence is

`Default_Namespace_spec::=` The default namespace `<namespace_name>` is at `<URL>`.

Applying this default namespace convention, the OPL paragraph (collection of OPL sentences) that corresponds to the OPD in Fig. 16 is

The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
The namespace **Documents** is at WWW.Documents.org/definitions.
The namespace **Documents** defines the relation '**is the creator of**'.
The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila.

The XML script that specifies the analogous semantics, where the **Semantic Web** namespace is replaced by `rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"` and the **Documents** namespace is replaced by `s="http://description.org/schema/"`, is the one listed in Table 3.

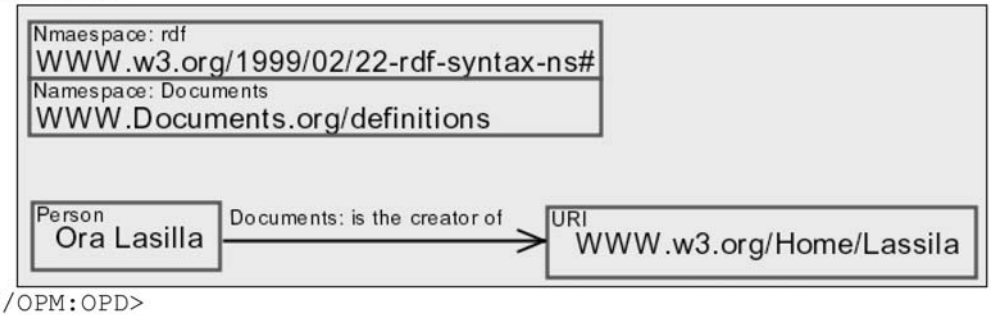
Wrapping the ViSWeb specification with XML and the DRF equivalent

Having finalized the ViSWeb specification, we now describe how it is all wrapped with XML to make it amenable to Web transfers and manipulations. We call this form XML/ViSWeb. We also describe how this XML/ViSWeb is translated into the XML/RDF standard. The following namespace declaration sentence, which is a constant part of any XML/ViSWeb text, specifies the OPM namespace:

Table 5. Comparison between the complete human-oriented XML/ViSWeb specification of our example and its corresponding machine-oriented XML/RDF translation

Human-oriented XML/ViSWeb

```
<?xml version="1.0"?>
<OPM:OPD>
```




```
</OPM:OPD>
<OPM:OPL>
```

The namespace **OPM** is at WWW.ObjectProcess.org/definitions.

The default namespace **rdf** is at WWW.w3.org/1999/02/22-rdf-syntax-ns#.

The namespace **Documents** is at WWW.Documents.org/definitions.

The namespace **Documents** defines the relation **'is the creator of'**.

The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila.

```
</OPM:OPL>
```

Machine-Oriented XML/RDF

```
<?xml version="1.0"?>
<OPM:OPD>
-- Here comes the XMI [11] specification of the OPD, which
enables its rendering shown above for human consumption. --
</OPM:OPD>
<OPM:OPL>
 xmlns:OPM="http://www.ObjectProcess.org/Definitions"
 <rdf:RDF>
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:Documents="http://documents.org/defintions">
   <rdf:Description about="http://www.w3.org/Home/Lassila">
   <Documents:'is the creator of'>
     Ora Lassila
   </Documents:'is the creator of'>
   </rdf:Description>
  </rdf:RDF>
</OPM:OPL>
```

The namespace **OPM** is at WWW.ObjectProcess.org/definitions.

The entire ViSWeb OPL specification is incorporated into the XML syntax by simply enclosing it within the <OPM:OPL> and </OPM:OPL> tags, as shown in Table 5. The OPD is likewise enclosed within the <OPM:OPD> and </OPM:OPD> tags. For human consumption, the actual graphic display of the OPD is presented in the XML/ViSWeb specification, as shown in Table 5. For machines, the actual OPD is replaced in the corresponding XML/RDF script by its XMI [28] representation.

The definition of 'is the creator of' as it would appear in the Documents namespace in its machine-oriented XML/RDF syntax might look something like this:

```
<Documents:'is the creator of'
  rdf:ID="creator"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
  syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-
  schema#">
rdf:about="http://www.documents.org/-
  defintions">
   <rdfs:isDefinedBy
   rdf:resource=
   "http://www.documents.org/defintions"/>
   <rdfs:label xml:lang="en"> 'is the
   creator of' </rdfs:label>
   <OPM:tag xml:lang="en">is the
   creator of</OPM:tag>
   <rdfs:domain rdf:resource="#Person"/>
   <rdfs:range rdf:resource="#URI"/>
```

```
   <rdfs:comment>
    Person and URI are not really defined in
    xmlns:rdf
   </rdfs:comment>
</Documents:'is the creator of'>
```

If we wish, we can ask for a translation of this XML/RDF script to its XML/ViSWeb human-oriented counterpart, using the RDF-to-ViSWeb compiler outlined in the next section.

Comparing the human-oriented XML/ViSWeb specification to its corresponding machine-oriented XML/RDF translation in Table 5, the advantages for humans of the former over the latter are evident:

- Graphically, the machine-oriented XMI specification of the ViSWeb OPD is rendered and displayed for humans as an intelligible diagram that contains the same information as the corresponding ViSWeb OPL script below it.
- Textually, the ViSWeb OPL script contains only sentences in a subset of natural English, which humans can read and understand with significantly less effort than is required for performing "mental compilation". Such mental compilation is what humans are effectively required to execute when they encounter any XML/RDF script that they wish to interpret.

The idea, then, is to show humans a human-oriented XML/ViSWeb specification, exemplified by the top part of Table 5, while the machine will still be able to process its "pure" XML/RDF translation, shown at the bottom of Table 5. To do this, we must have a utility for bidirectional translation between ViSWeb and RDF. In the next section, we outline how the XML/RDF translation can be obtained from the XML/ViSWeb specification and, conversely, how the XML/ViSWeb translation can be obtained from the XML/RDF specification.

8 Bidirectional translation between ViSWeb and RDF

In this section we outline how the XML/ViSWeb specification can be compiled into XML/RDF and vice versa. This bidirectionality is important since for machine processing and communication between nodes in the Web the XML/RDF is best, while for human understanding we have seen that the XML/ViSWeb specification is far more intelligible and user friendly in its bimodal graphic-text representation.

Outline of a ViSWeb-to-RDF compiler

The ViSWeb-to-RDF compiler translates XML/ViSWeb specification to its equivalent standard XML/RDF. Continuing our example, the ViSWeb script enclosed between the <OPM:OPL> and the </OPM:OPL> tags shown in Table 5 are in their quote-suppressed form for human reading, but, as noted, for machine consumption it is available in the quote-expressed version, as follows:

```
<OPM:OPL>
The namespace 'OPM' is at
'WWW.ObjectProcess.org/definitions'.
The default namespace 'rdf' is at
'WWW.w3.org/1999/02/22-rdf-syntax-ns#'.
The namespace 'Documents' is at
'WWW.Documents.org/definitions'.
The namespace 'Documents' defines the
relation 'is the creator of'.
The 'Person' 'Ora Lasilla' 'is the
creator of' the 'URI'
'WWW.w3.org/Home/Lassila'.
</OPM:OPL>
```

The XML/RDF equivalent of the XML/ViSWeb system specification above can be obtained by a ViSWeb-to-RDF compiler whose operation is outlined as follows:

1. The ViSWeb namespace definition sentences are translated to their XML counterparts. For example, the sentence "`The namespace 'OPM' is at 'WWW.ObjectProcess.org/definitions'`" is translated into `xmlns:OPM=http://www.ObjectProcess.org/definitions`.
2. The ViSWeb class and relation definition sentences are translated into their XML formats. For example, the OPL sentence "`The namespace 'Documents' is at 'WWW.Documents.org/definitions'`" is translated into `Documents: 'is the creator of'`.
3. The namespaces are accessed to obtain the required definitions and use the details they contain, such as domain and range of predicates (tagged structural relations). For example, looking at the above `Documents` namespace definition of the predicate '`is the creator of`', the compiler finds that the domain and range of this predicate are `rdf:resource="#Person"` and `rdf:resource="#URI"`, respectively.
4. The compiler now examines the sentence "`The 'Person' 'Ora Lasilla' 'is the creator of' the URI 'WWW.w3.org/Home/Lassila'`."
   a. Having found out (in step 3) that `Person` is the domain of the predicate '`is the creator of`', the compiler looks for the quoted element following the OPL string '`Person`' and finds that it is '`Ora Lasilla`'.
   b. Similarly, knowing that `URI` is the range of '`is the creator of`', the compiler looks for the quoted element following '`is the creator of`' and finds that it is '`WWW.w3.org/Home/Lassila`'. `WWW` is expanded to `http://www`.
5. The compiler is now ready to construct the core of the XML script, which is equivalent to the OPL sentence "The **Person 'Ora Lasilla' is the creator of** the **URI** WWW.w3.org/Home/Lassila." The XML script is:

```
<rdf:Description about=
"http://www.w3.org/Home/Lassila">
<Documents:'is the creator of'>
  Ora Lassila
</Documents:'is the creator of'>
</rdf:Description>
```

This XML script is enclosed within the <rdf:Description> opening and closing tags,

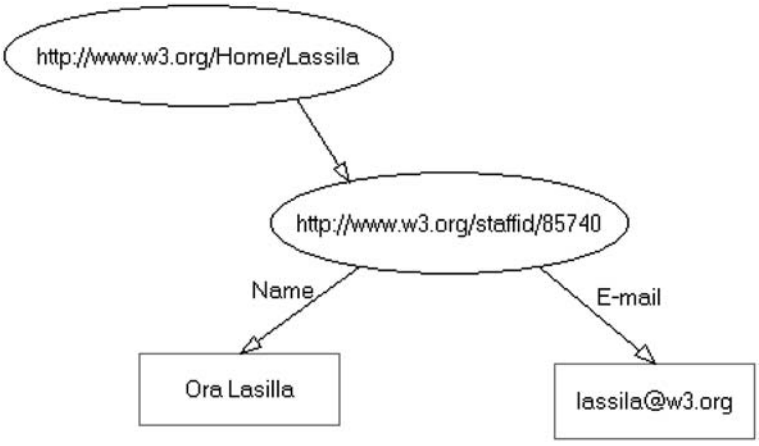


Fig. 17. An identified property with structured value [20]

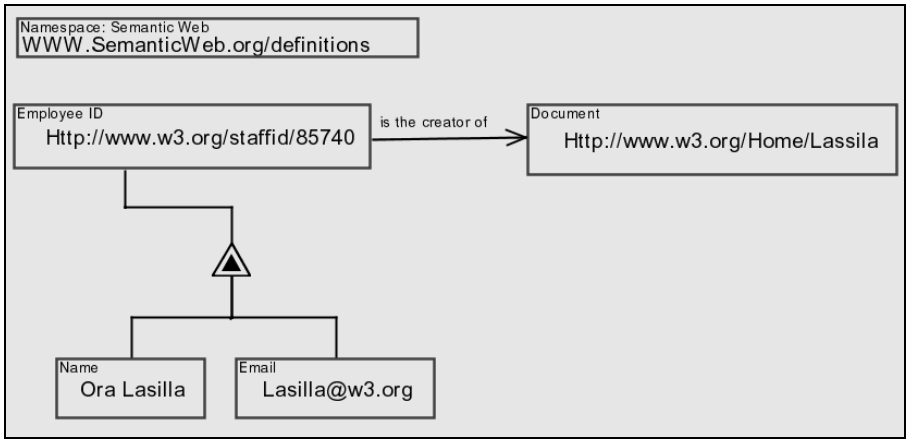


Fig. 18. The OPD that corresponds to the graph in Fig. 17

with the `about=` followed by the destination URI in double quotes. The domain, `Ora Lassila`, is then enclosed within the `<Documents:'is the creator of'>` and `</Documents:'is the creator of'>` tags.

Outline of an RDF-to-ViSWeb compiler

The reverse direction, that of obtaining the XML/ViSWeb script from the XML/RDF one, follows similar principles, as outlined next.

1. Namespace declarations are translated into their corresponding OPL namespace definition sentences. For example, the XML namespace declaration `xmlns:OPM=http://www.ObjectProcess.org/definitions` is translated into the OPL sentence "The namespace **OPM** is at WWW.ObjectProcess.org/definitions."
2. Class and relation definitions in XML are translated into their OPL sentence forms. For example, the XML script `Documents: 'is the creator of'` is converted to the OPL sentence "The namespace **Documents** defines the relation '**is the creator of**'."
3. Looking at the definition above of `'is the creator of'` in the `Documents` namespace, the compiler finds that the domain and range of this relation are `rdf:resource="#Person"` and `rdf:resource="#URI"`, respectively. The same information can be extracted from the XMI specification of the OPD.
4. Inspecting the XML script between the `<rdf:Description ... >` and the `</rdf:Description>` tags, the compiler extracts `Ora Lassila` as an instance of the class `Person`, which is already known to be the domain, and "`http://www.w3.org/Home/Lassila`" as an instance of the class `URI`, which is already known to be the range. This URI value is the value of the `about=` attribute of the `<rdf:Description>` tag. With this information, the compiler constructs the OPL sentence: "The **Person Ora Lasilla is the creator of** the **URI** WWW.w3.org/Home/Lassila."

9 Adding attributes

Continuing with the example from [20], for specifications that are more complex, a compound resource can be created, as the following sentence and the corresponding graph in Fig. 17 demonstrate:

"The individual referred to by employee ID 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource http://www.w3.org/Home/Lassila was created by this individual."

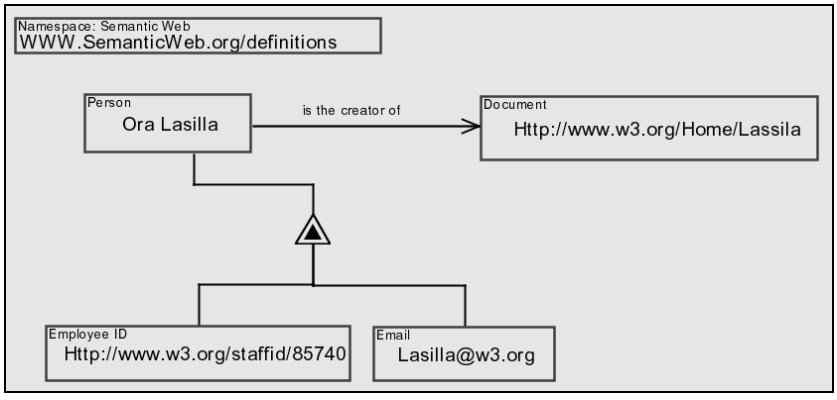


Fig. 19. A better representation of the information presented in the OPD in Fig. 18

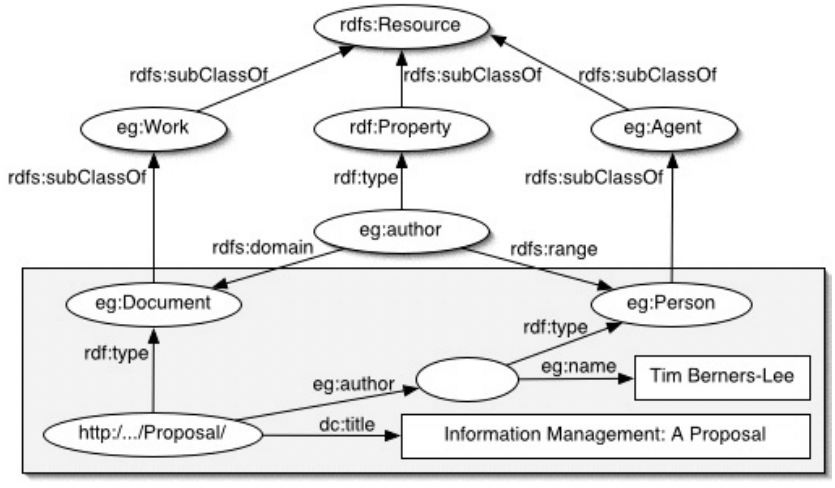


Fig. 20. Example of an RDF graph [8]

The OPL paragraph that corresponds to the OPD in Fig. 18 is:

> The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
> The **Employee ID** WWW.w3.org/staffid/85740 **is the creator of** the **Document** WWW.w3.org/Home/Lassila.
> The **Employee ID** WWW.w3.org/staffid/85740 exhibits the **Name Ora Lasilla** and the **Email** Lasilla@w3.org.

The OPL reserved word "exhibits" expresses the exhibition-characterization relation (the relation between a class and its attributes, symbolized by a black-in-white triangle) from The **Employee ID** Http://www.w3.org/staffid/85740 to the **Name Ora Lasilla** and to the **Email** Lasilla@w3.org.

A better representation of the information presented in the OPD in Fig. 18 is shown in the OPD of Fig. 19. The **Employee ID** is now an attribute of the **Person** rather than the other way around. That this is a better way of modeling is clearly seen when we compare the OPL paragraph below, which corresponds to the OPD in Fig. 19, to the previous OPL paragraph, which corresponds to the OPD in Fig. 18.

> The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
> The **Person Ora Lasilla is the creator of** the **Document** WWW.w3.org/Home/Lassila.
> The **Person Ora Lasilla** exhibits the **Employee ID** WWW.w3.org/staffid/85740 and the **Email** Lasilla@w3.org.

10 A final RDF schema example and its OPM counterpart

We conclude our examples with an example of an RDF schema and its OPM counterpart.

The example in Fig. 20, taken from [8], illustrates the way in which RDF can be used to describe real-world things (people, documents), the classes they fall into (such as `eg:Document, eg:Person`), and the properties that are used to relate members of these classes, such as the property `eg:author`. Through the `rdfs:domain` and `rdfs:range` predicates, the RDF Schema in Fig. 20 specifies that the `eg:author` property relates documents (the domain) to people (the range). The example also shows that all documents are considered to be works and that all people are agents.

Figure 21 is a ViSWeb OPD that expresses the semantics of the RDF graph in Fig. 20. Here, `rdfs` and `eg` are the two namespaces. Comparing Fig. 20 with Fig. 21, one can see the benefit of using OPD. In addition to the fact that the OPD

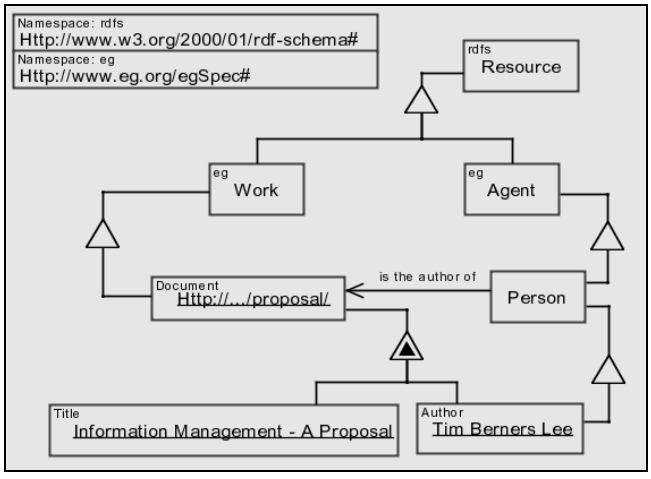


Fig. 21. An OPD expressing the semantics of the RDF graph in Fig. 20

translates completely to the OPL script below, it is also more compact and at the same time more expressive. For example, the node `eg:author` along with the links `rdfs:domain` and `rdfs:range` in the RDF graph are replaced in the OPD by the single directed structural link whose tag is '**is the author of**'. The ViSWeb OPL paragraph below corresponds to the OPD in Fig. 21.

The default namespace **rdfs** is at WWW.w3.org/2000/01/rdf-schema#.
The namespace **eg** is at WWW.eg.org/egSpecs#.
The namespace **eg** defines **Work** and **Agent**.
Work and **Agent** are **Resources**.
Document is a **Work**.
Person is an **Agent**.
Author is a **Person**.
Document exhibits **Author** and **Title**.
The **Document** Http://.../Proposal exhibits the **Author Tim Berners Lee** and the **Title Information Management – A Proposal.**

11 Navigation, querying, and reasoning

A critical issue of a modeling methodology is its ability to support knowledge management operations such as navigation, querying, and reasoning. To demonstrate these potential capabilities, consider the OPD in Fig. 22, which is part of an OPM model of a vehicle ABS (Antilock Break System), an instance of a real-time embedded system. Figure 22 shows a possible response of a future enhanced version of OPCAT, answering the query:
"Show immediate links between **Converted Signal Set** and **Vehicle Velocity**." Submitting this query generated the dashed lines in the OPD along the paths linking the two objects as well as the region of interest in the OPD that contains all the entities and links that are relevant to the query marked by the enclosing dotted line. In parallel, the text window would show the following OPL sentences:

Converted Signal Set consists of **Vehicle Acceleration Data**, **Yaw Acceleration Data**, and other parts.
Filtering consumes **Vehicle Acceleration Data** and **Yaw Acceleration Data**.
Filtering yields **Filtered Data**.
Integrating consumes **Filtered Data**.
Integrating yields **Vehicle Velocity**.

The two objects appearing in the query are underlined in the text to show that a path (consisting of the edges marked with the dashed lines) has been established between these two objects of interest. The entities and links contained in the region of interest can be drawn as a separate OPD to which facts expressed in other related OPDs can be added. This is a basis for more sophisticated reasoning queries about cause and effect, e.g., through forward or backward chaining. For example, since **Integrating** is a process along the path, a defective **Integrating** algorithm (process class) can compute an incorrect **Vehicle Velocity**, but **Vehicle Velocity** is not affected by **Differentiating** since that process is only responsible for yielding **Steering Velocity** and **Steering Angle**.

While these querying capabilities are not yet present in the current OPCAT version, the new XML-based knowledge base of OPCAT enables their development in a straightforward manner, and we are currently working toward achieving this goal.

12 OWL set operators in OPM

Since OWL class extensions are sets, OWL provides the means to manipulate class extensions using basic set operators, which include `intersectionOf`, `unionOf`, and `complementOf` [32]. These concepts are built into OPM both graphically in OPDs and in natural language via OPL. We demonstrate this using OWL's `intersectionOf` construct through the example in [32], listed below.

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType=
  "Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
    <owl:onProperty rdf:resource=
    "#hasColor" />
    <owl:hasValue rdf:resource="#White"
/>
   </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

As noted in [32], classes constructed using the set operations are *closed*. The members of the class are completely specified by the set operation. This is an important capability that permits us to state that `WhiteWine` is exactly the intersection of the class `Wine` and the set of things that are white in color. OPM's analog of the OWL's `intersectionOf` construct is qualification. Qualification restricts the set membership of instances of a class to just those with a specified value of a certain attributes. The OPL of our example is:

Wine exhibits **Color.**
Color can be **red** or **white.**
White Wine is a **Wine** the **Color** of which is **White.**

Since this OPL paragraph is a set of three concise English sentences, it does not require any more explanation or inter-

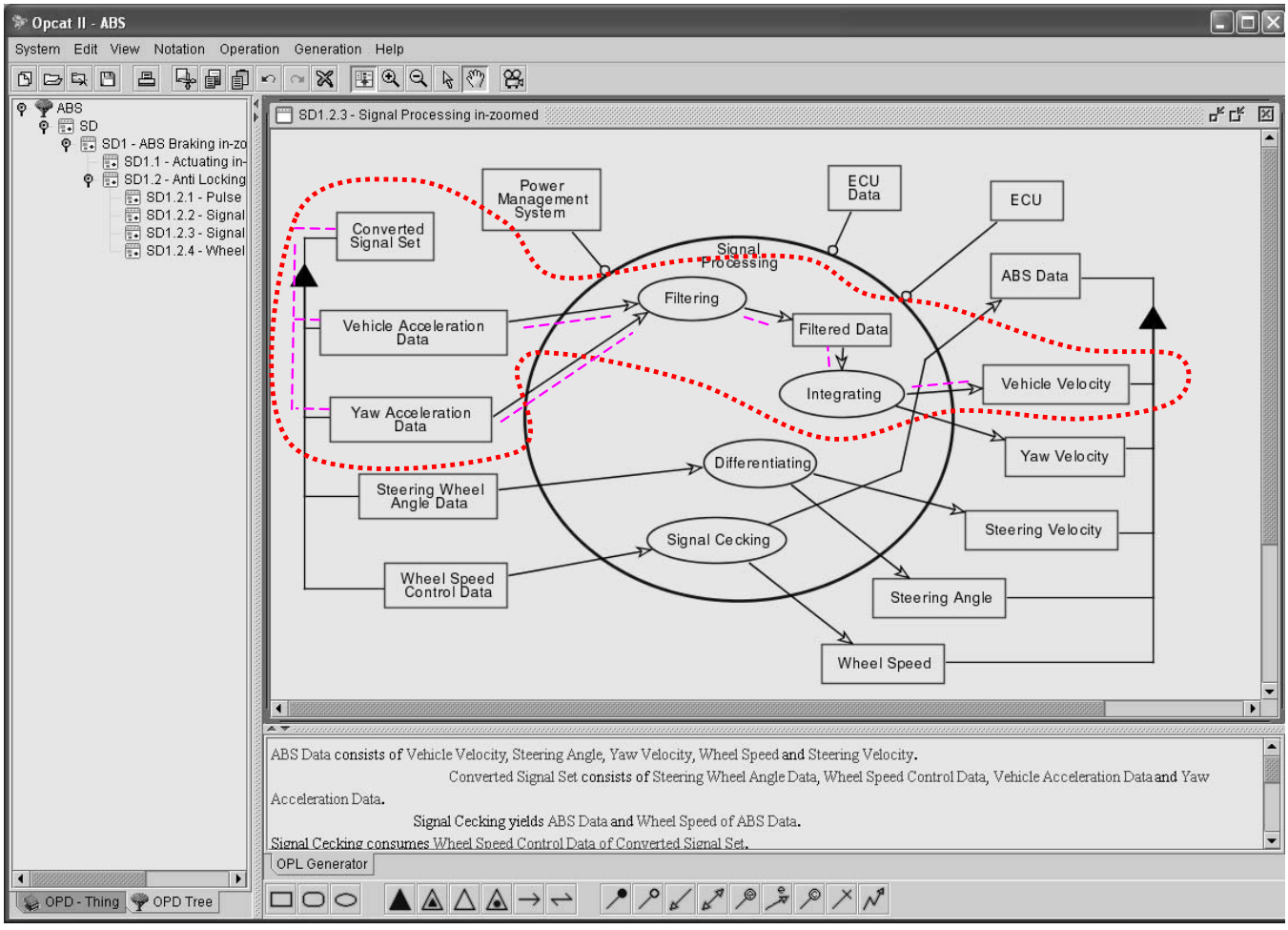


Fig. 22. An OPM model of a vehicle ABS (Antilock Break System) showing navigation and querying capabilities

pretation. The third sentence is a qualification sentence, which specifies in plain language that "**White Wine** is a **Wine** the **Color** of which is **White**." In other words, **Wine** whose **Color** is anything other than **white** is not **White Wine**. The corresponding OPD, depicted in Fig. 23, shows the qualification relation between the value **white** of **Color** of **Wine** and **White Wine**. The qualification relation is a specialization of the generalization-specialization relation in that, rather than linking the generalizing object to the specialized one, it links a specific value of an attribute of the generalizing object to the specialized object.

The example goes on to define `Burgundy` to include exactly those wines that have at least one `locatedIn` relation to the `Bourgogne Region`, as expressed in the RDF/OWL script below.

```
<owl:Class rdf:about="#Burgundy">
  <owl:intersectionOf rdf:parseType=
  "Collection">
   <owl:Class rdf:about="#Wine" />
   <owl:Restriction>
    <owl:onProperty rdf:resource=
    "#locatedIn" />
    <owl:hasValue rdf:resource=
    "#BourgogneRegion" />
   </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Finally, the example ends with the OWL script below, which specifies that the class `WhiteBurgundy` is exactly the intersection of white wines and `Burgundy`.

```
<owl:Class rdf:ID="WhiteBurgundy">
  <owl:intersectionOf rdf:parseType=
  "Collection">
   <owl:Class rdf:about="#Burgundy" />
   <owl:Class rdf:about="#WhiteWine" />
  </owl:intersectionOf>
</owl:Class>
```

To do the same in OPM, we add to **Wine** of Fig. 23 the attribute **Growing Region** with values **Burgundy**, **Anjou**, and **Chardonnay**. We then link **Burgundy** with a qualification relation to the more restricted class **White Burgundy**. The complex qualification sentence "**White Burgundy** is a **Wine,** the **Color** of which is **White** and the **Growing Region** of which is **Burgundy**" then expresses exactly in plain English what wines are considered to be **White Burgundy**.

The OPL paragraph that is equivalent to the OPD in Fig. 24 follows.

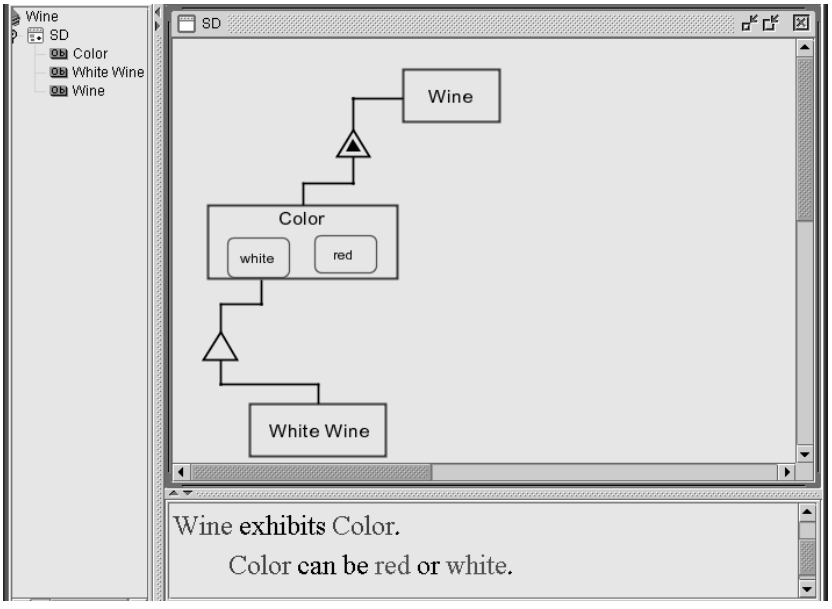


Fig. 23. An OPM model that defines **White Wine** as a qualification of **Wine** the **Color** of which is **white**

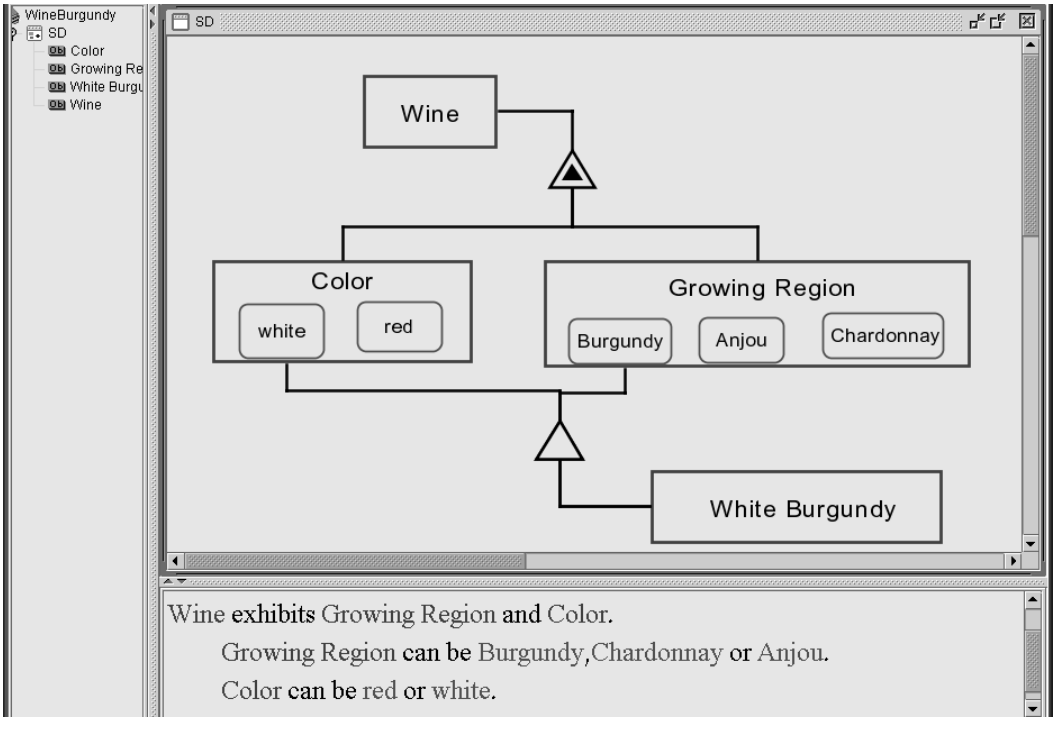


Fig. 24. The OPM model of Fig. 23 augmented with the attribute **Growing Location**, which defines **White Burgundy** as a qualification of **Wine** the **Color** of which is **white** and the **Growing location** of which is **Burgundy**

Wine exhibits **Color** and **Growing Region.**
Color can be **red** or **white.**
Growing Region can be **Burgundy, Anjou,** and **Chardonnay.**
White Burgundy is a **Wine,** the **Color** of which is **White** and the **Growing Region** of which is **Burgundy.**

Comparing the (last) qualification sentence with the six-line XML/RDF/OWL script above tells the whole story about the human orientation of OPL and how it is contrasted with XML machine orientation. Moreover, the OPL paragraph is also specified graphically, with no parallel in OWL.

13 Advantages of the Visual Semantic Web paradigm

The ViSWeb paradigm has a number of important advantages over present OWL/RDF/XML approaches, which are summarized in this section.

1. **Graphic-text knowledge representation:** The powerful graphic-text bimodal representation of OPM is extended to

the ViSWeb paradigm. Rather than having to mentally parse cryptic XML scripts, knowledge is presented to the user in a subset of natural language as well as diagrammatically. The two modalities complement each other, so if something is unclear in one representation, the other can be consulted for clarification. Using ViSWeb, one can ask for a translation from XML/RDF to XML/ViSWeb to get both visualization and a human-readable version of the XML syntax. The graphic representation can then be manipulated, changed, or augmented. Any such change would be reflected in the ViSWeb OPL script and through it transparently back to the XML/RDF machine-oriented syntax. This way, working in a round-trip engineering mode, the human gets to think and develop ideas in a user-friendly environment without compromising the technical soundness of her/his work.

2. **Visual navigability:** In addition to the advantages of putting to work the "two sides of the human brain", the visual and the lingual, there are benefits that are unique to the Semantic Web. The formal robust, yet intuitive, diagrammatic display enables users to surf and navigate the Web in a visual way in search of knowledge. For example, the bottom-left box in Fig. 21 can be a hyperlink to the actual document titled "Information Management – A Proposal." Objects, processes, classes, and links can be hyperlinked to pertinent Web sites, which themselves may contain ViSWeb or any other multimedia knowledge representations.

3. **Semantic sentence interpretation:** In spite of the aspirations of the Semantic Web, the basis of the RDF framework is syntactic rather than semantic: it draws on the concepts of *subject*, *predicate*, and *object*, which are parts of speech used to analyze natural language sentences from a syntactic viewpoint. The same semantics can be expressed by inverse syntactic expressions. For example, without changing the semantics, we could easily switch the roles of subject and object in the example of Fig. 10 by writing the sentence as "*The resource http://www.w3.org/Home/Lassila was created by Ora Lassila.*" Now the (syntactic) subject is *http://www.w3.org/Home/Lassila*, the object is *Ora Lassila*, and the predicate is "*was created by*". While the parts of speech are turned upside down and the predicate was changed from active to passive, the meaning of the sentence is still the same. The proposed OPM-based ViSWeb paradigm is based on a sound ontology of *objects* with *states* and *processes*: objects are things that (at least potentially, and possibly at some state) exist, while processes are things that happen to objects and transform them (i.e., create or destroy them, or change their state). Based on this ontology, sentences can be interpreted semantically rather than syntactically. In OPM, each structural relation pair has a forward direction and a backward direction [14], so, for example, the forward relation "*is the creator of*" is paired with "*was created by*". This helps overcome syntactic differences and establish semantic equivalence.

4. **Specification of system dynamics:** Current work on the Semantic Web places emphasis on declaratively specifying structural knowledge, which relates to the static aspect of systems. Structural knowledge pertains to relations among objects that are not related to the objective of the system or the way it operates. According to Berners-Lee [3], "*the RDF model is basically an opening of the ER model to work on the Web.*" A typical ER model involves entity types, each with its set of relationships. "*The RDF model is the same, except that relationships are first-class objects: they are identified by a URI, and so anyone can make one.*" This is a purely static world view, where everything can be expressed in terms of structural, time-independent relations. However, a major part of the knowledge about a system is functional (what is its purpose) and dynamic (how it operates). The current SW offers very little in this regard. Since OPM combines function, structure, and behavior in the same bimodal model, it provides a sound infrastructure for representing system dynamics and function in the ViSWeb model, as demonstrated in Fig. 6. While the details are beyond the scope of this work, suffice it to mention that knowledge about reactive and real-time systems requires treatment of events, conditions, actions, state transitions, and time exceptions, to name but a few major issues. All those and more can be modeled in OPM.

5. **Complexity management:** A major problem in real-life systems is their complexity due to the sheer amount of knowledge details. In addition to the OWL set operators that can be translated into OPM as demonstrated above, OPM has built in abstraction-refinement mechanisms, including in-zooming and out-zooming, unfolding and folding, and state expression and suppression. These allow for building hierarchies of knowledge representation in general and over the Web in particular, enabling navigation up and down abstraction-refinement hierarchies.

14 Summary and future work

The Visual Semantic Web (ViSWeb) paradigm proposes to unify human and machine representations of knowledge. The foundation for this unification is Object-Process Methodology (OPM), which advocates the integration of a system's structure and behavior in a single graphic and textual model. The paper has presented the principles and outlined an implementation of the ViSWeb. Like OPM, the ViSWeb model enables the representation of static and dynamic knowledge using a combination of Object-Process Language (OPL), a subset of English, and Object-Process Diagrams (OPDs), an equivalent visual formalism.

The advantages of this approach include graphic-text knowledge representation, visual navigability, semantic sentence interpretation, specification of system dynamics, and complexity management.

As noted in [20], "*It is also important to understand that this XML syntax is only one possible syntax for RDF and that alternate ways to represent the same RDF data model may emerge.*" Indeed, this work presents an OPM-based approach to representing the Semantic Web on top of the RDF data model, which is expressed graphically, using OPDs, and textually in OPL, a subset of natural English that is also "machine understandable", i.e., amenable to parsing and converting back to the XML-based RDF syntax.

Future work will proceed in both theoretical and practical directions. The theory will focus on extending the idea behind the ViSWeb paradigm and its initial specification, presented in this work, to cover other important knowledge and system representation aspects. Based on OPM, ViSWeb will be able to handle not only the declarative static structural aspects of

knowledge, which is the focus of the current Semantic Web initiative, but also procedural, dynamic behavioral aspects, and functional ones. The practical work will augment the current capabilities of OPCAT to make it suitable for modeling the various ViSWeb requirements presented here and provide the services of bidirectional RDF-ViSWeb compilation. An even more ambitious goal is to design and build a Web crawler that will automatically generate ViSWeb representations of knowledge stored in Web pages. Accomplishing even some of these goals will greatly benefit the huge World Wide Web user community by providing them with a friendly semantic surfing tool and relieving them of the need to mentally compile XML scripts.

References

1. Anderson JR, Bower GH (1973) Human associative memory. Winston, Washington, DC
2. Arnheim R (1969) Visual thinking. University of California Press, Berkeley, CA
3. Berners-Lee T (1998) What the Semantic Web can represent. http://www.w3.org/DesignIssues/RDFnot.html
4. Berners-Lee T, Hendler J (2001) Scientific publishing on the Semantic Web. Nature. http://www.nature.com/nature/debates/e-access/Articles/bernerslee.htm
5. Berners-Lee T, Hendler J, Lassila O (2001) The Semantic Web. Sci Am 284(5):34–43
6. Brachman R (1979) On the epistemological status of semantic networks. In: Findlee NV (ed) Associative networks: representation and use of knowledge by computer. Academic, New York, pp 3–50
7. Bray T, Hollander D, Layman A (1999) Namespaces in XML, World Wide Web Consortium Recommendation, 14 January 1999. http://www.w3.org/TR/REC-xml-names
8. Brickley D, Guha RV (2002) RDF Vocabulary Description Language 1.0: RDF Schema, W3C work in progress draft, 30 April 2002. http://www.w3.org/TR/rdf-schema
9. Chein M, Mugnier ML (1992) Conceptual graphs: fundamental notions. Rev Intell Artif 6(4):365–406
10. Corby O, Dieng R, Hebert C (2000) A conceptual graph model for W3C RDF. In: Proceedings of the international conference on conceptual structures (ICCS), Darmstadt, Germany, 14–18 August 2000. http://citeseer.nj.nec.com/cache/papers/cs/14625/http:zSzzSzwww.int.gu.edu.auzSzkvozSzreadingzSzoliviericcs2000.pdf/dieng00conceptual.pdf
11. Cyc. OpenCyc.org (2002) http://www.opencyc.org/
12. Delteil A, Faron C (2002) A graph-based knowledge representation language. In: Proceedings of the 15th European conference on artificial intelligence (ECAI), Lyon, France, 21–26 July 2002
13. Delugach H (2003) Conceptual Graphs homepage. http://www.cs.uah.edu/∼delugach/CG/
14. Dori D (2002) Object-Process Methodology – a holistic systems paradigm. Springer, Berlin Heidelberg New York. www.ObjectProcess.org
15. Dori D (2002) Why significant change in UML is unlikely. Commun ACM 45(11):82–85
16. Dori D, Reinhartz-Berger I, Sturm A (2003) OPCAT – a bimodal CASE tool for object-process based system development. In: Proceedings of the IEEE/ACM 5th international conference on enterprise information systems (ICEIS 2003), Angers, France, 22–26 April 2003, pp 286–291. www.ObjectProcess.org
17. Dublin Core Metadata Initiative (2002) http://www.dublincore.org/
18. Gaines BR, Shaw MLG (1995) Concept maps as hypermedia components. Int J Hum Comput Stud 43(3):323–361
19. Genesereth MR (1998) Knowledge Interchange Format. Draft proposal American National Standard (dpANS), NCITS.T2/98-004. http://logic.stanford.edu/kif/dpans.html
20. Lassila O, Swick R (1999) Resource Description Framework (RDF) model and syntax specification. W3C Recommendation, 22 February 1999. http://www.w3.org/TR/REC-rdf-syntax
21. Lehman F (ed) (1999) Semantic networks in artificial intelligence. Pergamon, Oxford, UK
22. Martin P, Eklund P (1999) Embedding knowledge in web documents: CGs versus XML metadata languages. In: Proceedings of the 7th international conference on conceptual structures (ICCS'99), Blacksburg, VA, 12–15 July 1999. Springer, Berlin Heidelberg New York
23. Mayer RE (2002) Multimedia learning. Cambridge University Press, New York
24. McTear MF (ed) (1998) Understanding cognitive science. Ellis Horwood, Chichester, UK
25. Novak JD (1977) A theory of education. Cornell University Press, Ithaca, NY
26. Novak JD, Gowin DB (1984) Learning how to learn. Cambridge University Press, New York
27. OMG UML1.4. Object Management Group (2001) Unified Modeling Language v.1.4, September 2001. http://www.omg.org/technology/documents/formal/uml.htm
28. OMG XMI. Object Management Group (2002) XML Metadata Interchange (XMI) specification v.1.2, January 2002. http://www.omg.org/technology/documents/formal/xmi.htm
29. Peirce CS (1932) Collected papers of Charles Sanders Pierce. In: Hartshorne C, Weiss P (eds) Harvard University Press, Cambridge, MA
30. Peleg M, Dori D (2000) The model multiplicity problem: experimenting with real-time specification methods. IEEE Trans Softw Eng 26(8):742–759
31. Pepper S, Moore G (2001) XML Topic Maps (XTM) 1.0. TopicMaps. Org Specification, 2001. http://www.topicmaps.org/xtm/1.0/
32. Smith MK, McGuinness D, Volz R, Welty C (2002) Web Ontology Language (OWL) Guide Version 1.0. W3C Working Draft, 4 November 2002. http://www.w3.org/TR/2002/WD-owl-guide-20021104/
33. Sowa JF (1984) Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading, MA
34. Sowa JF (2000) Conceptual Graph Standard, 2000. http://users.bestweb.net/∼sowa/cg/cgstandw.htm#Header_44
35. Sowa JF (2002) The Common Logic Standard initiative. http://suo.ieee.org/email/msg08241.html
36. Sowa JF (1999) Knowledge representation: logical, philosophical, and computational foundations. Brooks Cole, Pacific Grove, CA
37. W3C RDF Validation Service (2003) http://www.w3.org/RDF/Validator/