Semantics in Web Engineering: Applying the Resource Description Framework

Reinhold Klapsing
University of Essen, Germany

Gustaf Neumann
Vienna University of Economics and Business Administration, Austria

Wolfram Conen
Xonar GmbH, Germany

We present an extensible Web modeling framework that applies the Resource Description Framework to Web engineering, providing an interoperable exchange format. Our framework uses the same (meta) data model to specify a Web application's structure and content, to make statements about a Web application's elements, and to reason about the data and metadata.

The initial purpose of the Web was to exchange documents in heterogeneous environments. Web application development was mainly an authoring task. As Web applications evolve into complex information systems, industry needs a rigorous and systematic development approach. Conventional software engineering approaches[1] could handle the complexity. However, the Web's low-level implementation model and document-centric, hypermedia architecture with its mix of structured and unstructured data requires us to recast conventional software engineering approaches or develop new ones.

Web engineering should help us manage the Web application's life cycle. To do so, we need a formal model for defining a Web application in an interoperable format that can be exchanged easily, so that a developer can use the suitable tool for each phase of the Web application's life cycle. The exchange of a Web application model (or parts of it) is also a prerequisite for the distributed development of a Web application.

To address these requirements, we present a formal Web application description model, the Extensible Web Modeling Framework (XWMF), which is an application of the Resource Description Framework.[2,3] More precisely, the XWMF consists of an extensible set of RDF schemata and descriptions. RDF is an open standard available from the World Wide Web Consortium (W3C) that provides a model for processing metadata, which is expressed in an interoperable and machine-understandable format that can be exchanged on the Web. (See the "Related Work" sidebar for background on other Web design approaches.) RDF is intended to serve as the foundation of the semantic Web—the recent vision of the W3C.

In XWMF, we use RDF metadata to describe Web resources' properties and their relationships. The distinction between data and metadata depends on the application domain. XWMF applies a data and metadata model to define a Web application's resources and to make statements about them. This model results in machine-understandable content and a metadata descriptions of a Web application. A single graph-based data model describes design concepts and the final product.

An RDF graph is a semantic network, which first-order logic (FOL) can describe formally. Different users can achieve interoperability only if they (their agents, tools, and systems) interpret an RDF data model in the same way. Important aspects of the RDF model are, however, expressed in prose, which might lead to misunderstandings. To avoid this, capturing the RDF's intended semantics in FOL might be a valuable contribution and provide the RDF with a formalization, allowing its full exploitation as a key ingredient in the evolving semantic Web.

In an earlier article,[4] we expressed the RDF model's concepts and constraints in FOL. The XWMF exploits this approach to describe a Web application and its content formally. As a benefit, a user or developer can query and validate the data and metadata's semantics of a Web application based on FOL. This enables not only syntactic

1070-986X/01/$10.00 © 2001 IEEE

interoperability (which we can already reach by applying the Extensible Markup Language, or XML) but also semantic interoperability.

The XWMF's basic model is extensible; thus, a Web developer can introduce new design artifacts corresponding to unforeseen requirements. The XWMF bridges the gap between high-level Web design concepts and the low-level file-based Web implementation model. We model a Web application in terms of classes and objects instead of files. The XWMF supports the separation of layout and content information and the reuse of artifacts.

We've developed XWMF tools that can automatically convert a Web application description into the corresponding Web implementation. Additionally, the tools implement a system that supports querying and validating the Web application's metadata as well as the application data.

Extensible Web modeling framework

The XWMF (see Figure 1, next page) is an extensible set of RDF schemata and descriptions defining a Web application's properties.

The generic Web engineering schemata and RDF provide the vocabulary for creating Web application schemata. A developer creates a Web application schema to specify high-level, application-specific concepts for a Web application's design. A developer uses the vocabulary RDF provides—the generic Web engineering schemata—and the Web application schemata to create Web application descriptions. A Web application description defines the objects, relationships of the objects, and properties attached to objects. The XWMF tools can automatically convert the Web application descriptions into the Web application.

Web object composition model

The WOCM, the core of our framework, is an object-based formal metadata model for designing a Web application's structure and content. The WOCM gives an abstract view of the Web application not constrained by the file-based Web implementation model. The WOCM defines the constructs simplexon and complexon, which the developer arranges in a directed acyclic graph (DAG), with simplexons as the leafs. Complexons define a Web application's structure, while simplexons define the content and the content representation. Components are special complexons that represent a Web application's physical resources. Figure 2 (on page 65) shows an example of a DAG.

Related Work

XWMF avoids the constraints of a file-based approach to Web application modeling. Several other (Web) design approaches exist that also provide more sophisticated abstractions than the file-based Web implementation model. Examples include the object-oriented hypertext design model (OOHDM)[1] and relationship management methodology (RMM).[2]Developers mainly use these approaches for the design phase of a Web application's life cycle. In contrast, XWMF provides a standardized exchange format for a Web application design as a foundation to support the whole life cycle of Web applications.

The Web object-oriented model (WOOM)[3] provides a generative model for describing Web applications in terms of objects arranged in a directed acyclic graph (DAG). For each object, developers use a transformer method to convert the object into its Web implementation. In the XWMF's Web object composition model, objects are also arranged in a DAG. In contrast to WOOM, XWMF assigns a property to classes that provides information used to convert objects into their Web implementations to avoid hiding modeling information in a transformer method and, thus, not part of the Web application model.

Like the objects in WOOM, the objects in WebComposition[4] encapsulate their conversion in a dedicated method. A WebComposition application model is expressed in XML and thus in a standardized, programming-language-independent exchange format. In XWMF, we apply the standardized RDF. Thus, it's possible to reason about a Web application's data and metadata, which isn't directly supported if developers use pure XML.

Many commercial tools exist for Web authoring, development, and site management. However, many of these tools are rather self-contained and difficult to integrate with other tools used in the development process. XWMF applies a standardized exchange format, which can help developers integrate Web engineering tools.

References

1. D. Schwabe, G. Rossi, and S.D.J. Barbosa, "Systematic Hypermedia Application Design with OOHDM," *Proc. Hypertext*, ACM Press, New York, 1996.
2. T. Isakovitz, E.A. Stohr, and P. Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design," *Comm. ACM*, vol. 38, no. 8, Aug. 1995, pp. 34-44.
3. F. Coda et al., "Towards a Software Engineering Approach to Web Site Development," *Proc. Ninth Int'l Workshop Software Specification and Design* (IWSSD), IEEE Press, Piscataway, N.J., 1998.
4. H.-W. Gellersen, R. Wicke, and M. Gaedke, "WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle," *Computer Networks and ISDN Systems* (*Proc. Sixth Int'l World Wide Web Conf.*), vol. 29, Apr. 1997, pp. 1429-1437.

Complexons

Complexons are containers for simplexons and complexons. Circular containment is not allowed—a certain complexon instance can't contain itself. Complexons separate concerns by allowing definitions of a Web application's structure that remain independent of file organization.

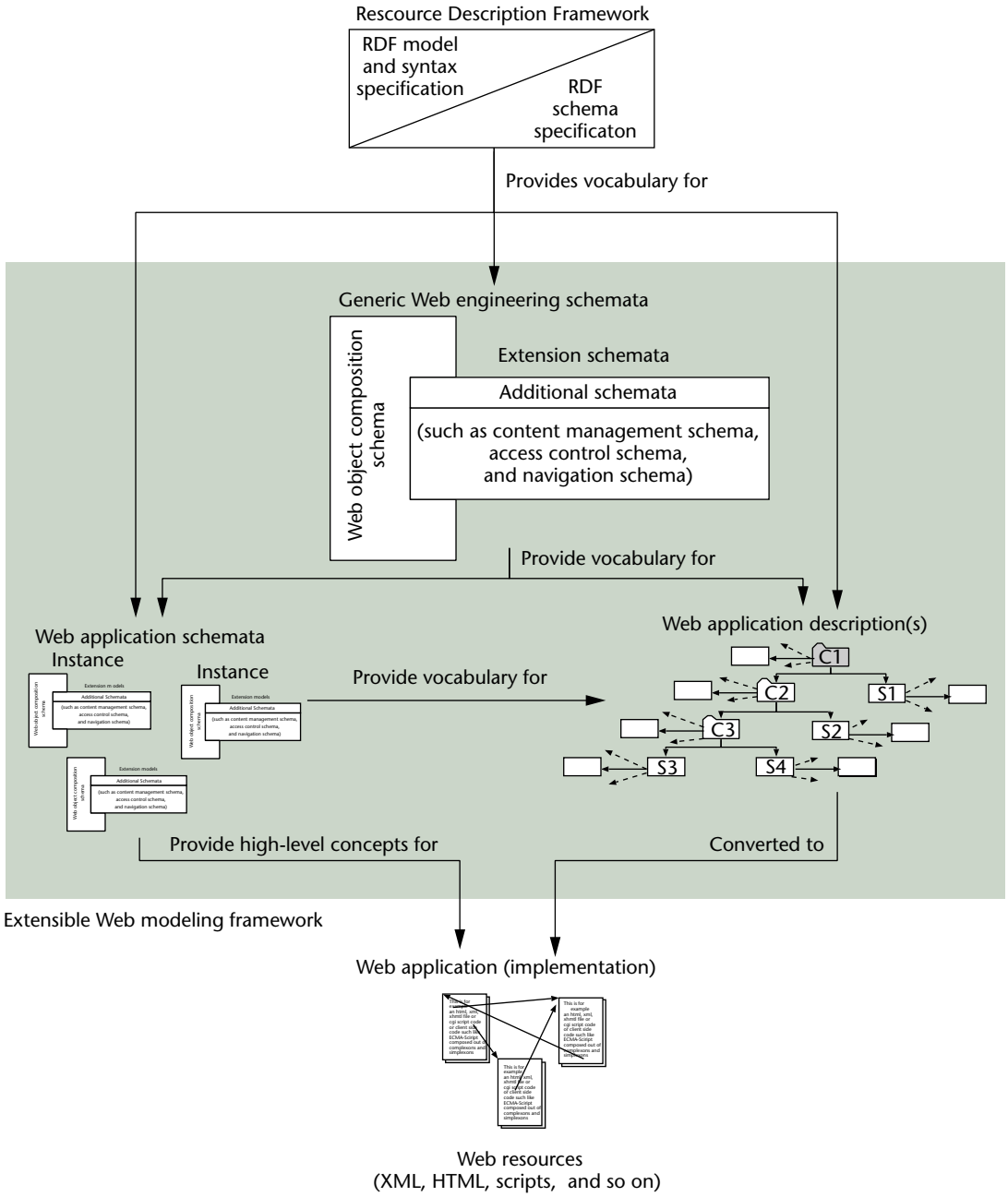


Figure 1. Extensible Web modeling framework. RDF provides the vocabulary for new RDF schemata and descriptions. This is used to express the generic Web engineering schemata in the center of the figure. These generic schemata define application-independent vocabularies to support Web engineering tasks. The generic Web object composition schema (WOCS) provides the vocabulary for defining a Web application's structure and content in terms of objects. The XWMF is extensible because the developer can include additional generic Web engineering schemata. Such extension schemata can provide vocabulary for various application areas such as advanced navigation modeling, access control modeling, or content management.

A developer can model the containment relationship with the property `hasPart`, which the Web object composition schema defines. Figure 3 gives an example.

Simplexons

A simplexon class defines an abstract data type. In addition, a simplexon class defines the Web implementation for objects of that class. Thus, it

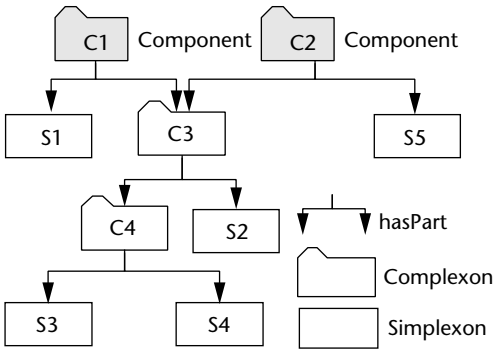


Figure 2. Example directed acyclic graph of a component. We chose to use a directed acylic graph (DAG) because it can express a set of trees in one graph. A tree can express the structure of a markup, which represents a Web implementation's content.

```
<rdfs:Class rdf:ID="Employee"/>

<lr:Employee rdf:ID="i1EmployeeHtml">
 <xwmf:hasPart>
  <rdf:Seq>
   <rdf:li rdf:resource="#htmlDocumentBegin"/>
   <rdf:li rdf:resource="#i1Contact"/>
   <rdf:li rdf:resource="#htmlDocumentEnd"/>
  </rdf:Seq>
 </xwmf:hasPart>
</lr:Employee>
```

Figure 3. Definition of the complexon class Employee (in abbreviated RDF syntax). The complexon instance i1EmployeeHtml of type Employee has the property hasPart referring to a sequence of simplexon instances. Note that the type concept lets us query the model to show all instances of a certain type (for example, type Employee).

```
<rdfs:Class rdf:ID="Contact"/>

<rdf:Property rdf:ID="name">
 <rdfs:domain rdf:resource="#Contact"/>
 <rdfs:range rdf:resource=
  "http://.../PR-rdf-schema-19990303#Literal"/>
</rdf:Property>
```

Figure 4. Definition of the simplexon class Contact and the property name assigned to it.

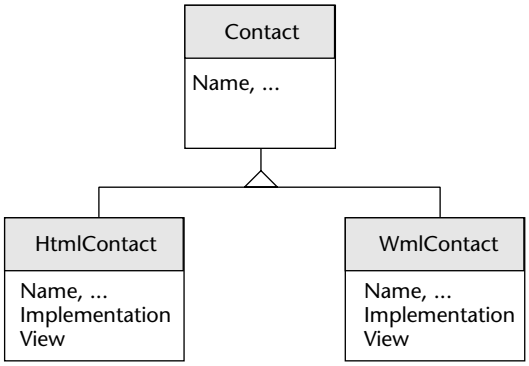


Figure 5. An example of inheritance. The simplexons HtmlContact and WmlContact inherit the property name.

supports the separation of content from its Web implementation because we can define an object's content independently of its Web implementation. The developer of a Web application defines which properties a simplexon should have. Figure 4 gives an example. In contrast to other object-oriented systems, RDF treats properties as first class objects. This property-centric approach enables multiple uses of properties in different contexts.

For simplexons, the XWMF supports subclassing. The subclassing relationship lets the developer model property inheritance. Figures 5 and 6 give examples.

A developer defines implementation templates as the value of the property `implementation`. The XWMF tools use the implementation template to convert a simplexon into the corresponding Web implementation. The property `implementation` can be respecified by a subclass to refine the Web implementation model. The WOCS allows any markup or string as a value for the property `implementation`. It allows, for example, HTML, XML, and Wireless Markup

```
<rdfs:Class rdf:ID="HtmlContact">
 <rdfs:subClassOf rdf:resource="#Contact"/>
 <xwmf:view>HTML</xwmf:view>
 <xwmf:implementation rdf:parseType="Literal">
  <![CDATA[<CENTER><TABLE bgcolor="#e8e8e8" >
   <TR><TD>
    <var>lr:name</var><br>
    <var>lr:phone</var><br>
    <var>lr:fax</var><br>
    <A HREF="mailto:<var>lr:email</var>">
    <var>lr:email</var></A><BR>
  </TD></TABLE></CENTER>]]>
 </xwmf:implementation>
</rdfs:Class>
```

Figure 6. The RDF syntax that defines the subclass relationship of class Contact and HtmlContact with the property subClassOf. The construct <var>lr:name</var> is a placeholder to be substituted for a value of the property name of an instance of class HtmlContact.

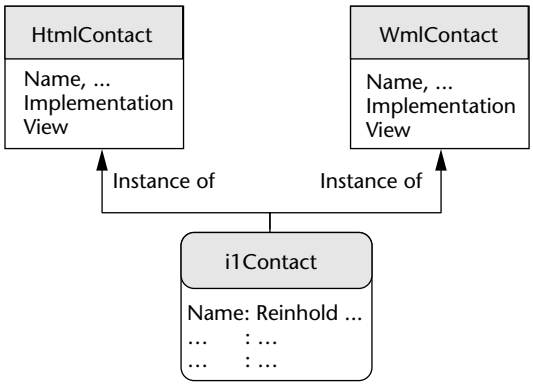
```
<lr:HtmlContact rdf:ID="i1Contact">
 <lr:name>Reinhold Klapsing</lr:name>
 <lr:phone>+49 (201) 183 4078</lr:phone>
 <lr:fax>+49 (201) 183 4073</lr:fax>
 <lr:email>
   Reinhold.Klapsing@uni-essen.de
 </lr:email>
</lr:HtmlContact>
```

Figure 7. The object i1Contact, an instance of the class HtmlContact. Note that i1Contact is part of the complexon itEmployeeHtml, as Figure 3 defines.

```
<CENTER><TABLE bgcolor="#e8e8e8" ><TR>
 <TD>Reinhold Klapsing<br>
  +49 (201) 183 4078<br>
  +49 (201) 183 4073<br>
 <A HREF=
  "mailto:Reinhold.Klapsing@uni-essen.de">
  Reinhold.Klapsing@uni-essen.de</A><BR>
</TD></TABLE></CENTER>
```

Figure 8. The Web implementation (in this case HTML) of the object i1Contact. The occurrences of the var construct in the value of the property implementation of the class HtmlContact have been replaced by the values of the corresponding properties of the object i1Contact.

Figure 9. Multiple inheritance of values lets us capture different views.



Language (WML). In addition to modeling Web pages (or parts of them), the developer can use simplexons to model all or part of the program code for the client or server side. Markup or program code at any granularity are allowed by WOCS as the value of the property **implementation**. For example, defining a whole (parameterized) Web page with a single simplexon supports ad-hoc design. By defining one simplexon per concept embodied in a Web page and composing these simplexons using complexons, the developer can achieve a more sophisticated design.

The XML element **var** can be used as a placeholder inside the value of the property **implementation**. Figure 6 shows an example with several placeholders. Merging an object's values via the **var** construct with the value of the property **implementation** defined by the class an object is related to generates the Web implementation. This supports reuse because objects belonging to a simplexon class use the same Web implementation—the value of the property **implementation**. Figures 7 and 8 give an example.

An object can be an instance of more than one simplexon class. This indicates that the object has all the characteristics that developers expect from an object of each class. This, in conjunction with the **view** property, which the WOCS defines, lets us define different views for an object according to the context in which we use it. Additionally, this supports object reuse because the developer has to define only one object for possibly many views (see Figures 9 and 10).

Components

A complexon is augmented with an **isComponent** property is a component. A component represents one of a Web application's physical parts (for example, a Web page stored in a file). A Web application consists of a set of components. For each component, the corresponding complexons and simplexons are arranged by the **hasPart** property in a tree (see Figure 2). The developer gives a Uniform Resource Identifier as a value of an **isComponent** property. The URI references the place from which a component can be feteched. The value of the (optional) property **isView** defines in which context a component's simplexons must be instantiated. A component must have a property **isView** if it contains simplexons that are instances of more than one class (see Figure 11).

Integrating extension models

Using XML namespaces[5] supports integrating extension models into the XWMF. An XML namespace refers to the corresponding RDF schema that defines the vocabulary used to apply the extension model. To extend the WOCM, a developer can assign properties to complexons, simplexons, and components. Figure 12 shows the RDF syntax of the simplexon object **i1Contact**'s extension. The property **expires** defines the expiration date of the information represented by that object. A content management software can query the model for the property **expires** to determine which information became obsolete. The RDF schema http://.../schema/cm.rdf defines the property

```
<rdf:Description rdf:ID="i1Contact">
 <rdf:type rdf:resource="#HtmlContact"/>
 <rdf:type rdf:resource="#WmlContact"/>
 <lr:name>Reinhold Klapsing</lr:name>
 <lr:phone>+49 (201) 183 4078</lr:phone>
 <lr:fax>+49 (201) 183 4073</lr:fax>
 <lr:email>
    Reinhold.Klapsing@uni-essen.de
 </lr:email>
</rdf:Description>
```

Figure 10. The RDF syntax corresponding to Figure 9. Note that it isn't possible to use abbreviated RDF syntax as in the other examples because of the multiple use of the type property. In Figure 6, the simplexon class HtmlContact defines the value HTML for the property view. Thus, we can use the instance i1Contact in an HTML context. In the same way, the class WmlContact defines the view WML, so we can also use the instance i1Contact in a WML context.

```
<lr:Employee rdf:ID="i1EmployeeHtml">
 <xwmf:isComponent rdf:resource =
  "http://.../Klapsing.html"/>
 <xwmf:isView>HTML</xwmf:isView>
 <xwmf:hasPart>
  <rdf:Seq>
   <rdf:li rdf:resource="#htmlDocumentBegin"/>
   <rdf:li rdf:resource="#i1Contact"/>
   <rdf:li rdf:resource="#htmlDocumentEnd"/>
  </rdf:Seq>
 </xwmf:hasPart>
</lr:Employee>
```

Figure 11. The component instance i1EmployeeHtml, an instance of the complexon class Employee. It is accessible via the URI http://.../Klapsing.html. The value HTML of the property isView defines the context in which the component's simplexons must be instantiated. The context HTML defines that the implementation of the simplexon class HtmlContact (and not of the simplexon class WmlContact) must be used during the generation process.

expires. The content management schema is bound to the XML namespace prefix **cm**.

Note the possibility of storing an extension description separate from the WOCM description. Thus, a content management tool needs to operate only on the content management description, possibly stored on a remote system.

Implementation

We developed a set of tools to create (Gramtor), to process (Web application generation tools), and to query and analyze (WebObjectBrowser, RDFHandle, RDF Schema Explorer) a Web application's RDF description. The tools are written in the object-oriented scripting language Extended Object Tcl (XOTcl)[6] and in Prolog. For parsing RDF models, we developed an RDF parser that uses the TclXML parser (see http://www.zveno.com/zm.cgi/in-tclxml). The RDF parser produces the triples of an RDF model. The RDF Handle operates on triples and provides an interface to modify an RDF model and store the model in triple notation and RDF syntax. Additionally, the RDF Handle provides an interface for querying an RDF model. The WebObjectComposer uses the query interface to analyze the Web application description. The WebObjectComposer generates XOTcl classes and objects representing a Web application's simplexons, complexons, and components and can generate the corresponding Web implementation.

Gramtor is an RDF editor for the graphical development of RDF models. It can save RDF models in different formats including the triple notation and the RDF syntax.

```
<rdf:RDF xmlns="http://.../22-rdf-syntax-ns#"
 xmlns:rdf="http://.../22-rdf-syntax-ns#"
 xmlns:rdfs="http://.../CR-rdf-schema-20000327#"
 xmlns:xwmf="http://.../schema/xwmf.rdf#"
 xmlns:cm="http://.../schema/cm.rdf#"
 xmlns:lr="this#">
...
<lr:HtmlContact rdf:ID="i1Contact">
 <lr:name>Reinhold Klapsing</lr:name>
 <lr:phone>+49 (201) 183 4078</lr:phone>
 <lr:fax>+49 (201) 183 4073</lr:fax>
 <lr:email>
  Reinhold.Klapsing@uni-essen.de
 </lr:email>
 <cm:expires>2001-12-31T12:00:00Z</cm:expires>
</lr:HtmlContact>
</rdf:RDF>
```

Figure 12. Integrating extension schemata.

The WebObjectBrowser provides a graphical interface that shows the class hierarchy and the objects of a Web application as a graph. A developer can browse the simplexons, complexons, and components to analyze the classes and objects. The main parts of the Gramtor and WebObjectBrowser are in the XOTcl scripting language. We developed the graphical user interfaces with the Motif version of Wafe,[7] which provides a Tcl interface to the XToolkit.

The RDF schema explorer we developed is a Prolog-based tool built on top of the SWI-Prolog RDF parser (see http://www.swi.psy.uva.nl/projects/SWI-Prolog). It lets us query an RDF

model on the statement level and with respect to the facts and rules that capture the RDF's semantic concepts and constraints.[4] For this purpose, a number of predefined predicates is available, letting the user validate models against the RDF rule set. In addition, the user can define the semantics of newly introduced predicates from within the RDF and can query, check, and validate these extended models as well.

Future work

In the future, we plan to support more sophisticated navigational user guidance by defining vocabulary for guided tours. This, together with the XWMF query system, will allow such tasks as generating a guided tour of all resources of a certain type. To support a wider reuse of concepts and Web implementation templates, we plan to develop a repository mechanism and extend the XWMF tools accordingly. Readers can obtain current and future versions of the tools from the XWMF homepage (http://nestroy.wi-inf.uni-essen.de/xwmf). **MM**

Acknowledgments

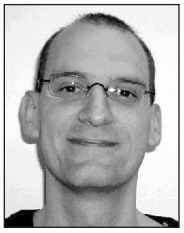
We thank Eckhart Köppen for many useful discussions regarding the XWMF. Alexander Block developed significant parts of the Gramtor and the RDF-Handle as part of his thesis. Fredj Dridi is author of the tool Inspector; the WebObjectBrowser is a readjustment of this tool.

References

1. *IEEE Software Engineering Standards Collection*, IEEE Press, Piscataway, N.J., 1999.
2. D. Brickley and R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0. Candidate Recommendation*, World Wide Web Consortium, 2000, http://www.w3.org/TR/2000/CR-rdf-schema-20000327.
3. O. Lassila and R.R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification Recommendation*, World Wide Web Consortium, 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.
4. W. Conen and R. Klapsing, "A Logical Interpretation of RDF," *Linköping Electronic Articles in Computer and Information Science*, vol. 5, no. 13, Dec. 2000, http://www.ep.liu.se/ea/cis/2000/013.
5. T. Bray, D. Hollander, and A. Layman, *Namespaces in XML Recommendation*, World Wide Web Consortium, 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114.
6. G. Neumann and U. Zdun, "XOTcl, An Object-Oriented Scripting Language," *Proc. Seventh Usenix Tcl/Tk Conf.* (Tcl2k), Usenix, Berkeley, Calif., 2000.
7. G. Neumann and S. Nusser, "Wafe—An X Toolkit Based Frontend for Application Programs in Various Programming Languages," *Proc. Usenix Winter Technical Conf.*, Usenix, Berkeley, Calif., 1993.

Reinhold Klapsing is a PhD student in the Department of Information Systems and Software Techniques at the University of Essen, Germany. His technical interests include (mobile) Internet and intranet technologies, the semantic Web (including RDF), and software engineering. He has a BS and MS in information systems from the University of Essen.

Gustaf Neumann is the chair of the Information Systems and New Media Department at the Vienna University of Economics and Business Administration, Austria. He has published books and papers in information systems technologies, data modeling, security management, program transformation, and logic programming. He is the author of several widely used open-source software packages, including the object-oriented scripting language XOTcl.



Wolfram Conen is a cofounder of Xonar GmbH, a company developing tools and applications supporting Web enginnering, mobile content sharing, and combinatorial auctions. He holds an MS in information systems from the University of Mannheim.

Contact Klapsing at Information Systems and Software Techniques, University of Essen, Universitätsstrasse 9, D-45141 Essen, Germany, email Reinhold.Klapsing@uni-essen.de, http://nestroy.wi-inf.uni-essen.de.