IEICE TRANS. INF. & SYST., VOL.E89–D, NO.4 APRIL 2006

PAPER *Special Section on Knowledge-Based Software Engineering*

A Graphical RDF-Based Meta-Model Management Tool

Takeshi MORITA[†a)], **Noriaki IZUMI**[††], ***Nonmembers***, **Naoki FUKUTA**[†††], ***and*** **Takahira YAMAGUCHI**[†], ***Members***

SUMMARY We propose a tool to manage several sorts of relationships among RDF (Resource Description Framework) and RDFS (RDF Schema). Our tool consists of three main functions: graphical editing of RDF descriptions, graphical editing of RDFS descriptions, and meta-model management facilities. In this paper, we focus on the meta-model management, a key concept which is defined as the appropriate management of the correspondence between a model and its meta-model: especially, the class and property in the meta-model, and the type of RDF resource and property in the model. The above facilities are implemented based on the plug-in system. We provide basic plug-in modules for checking the consistency of RDFS classes and properties. The prototyping tool, called MR^3 (Meta-Model Management based on RDFs Revision Reflection), is implemented by Java language. Through an experiment using MR^3, we show how MR^3 contributes to the Semantic Web paradigm from the standpoint of RDFs description management.

key words: *Semantic Web, RDF, RDFS, editor, meta-model management*

1. Introduction

The Semantic Web, which is one of the most promising candidates for tomorrow's Web, is based on RDF (Resource Description Framework) [1] and RDFS (RDF Schema) [2] recommended by the W3C (World Wide Web Consortium). The purpose of the Semantic Web is to make data on the Web available not only for human beings but also for automated processing, which would be specialized for the integration and reuse of data across various applications. For the management of RDF and RDFS descriptions, a number of graphical editors have been provided, and their main function is to display XML-based descriptions based on the RDF data model of "Resource-Property-Value" semantics. The graphical editors enable us to understand RDF descriptions graphically and to easily handle RDF files, which can be described as a complex XML resource. However, there still remains the difficulty of looking at the whole structure of both RDF and RDFS descriptions. In order to resolve this difficulty, a number of model-based frameworks for RDF descriptions have been proposed, such as N3 [3], TRIPLE [4], and so on. They provide the reasoning framework for the RDF model but still remain a model-based concept. A number of supporting environments have been developed as tools adopted from traditional knowledge engineering based on ontologies (e.g., [5], [6]). These products mainly concentrate on creating ontologies and managing ontology-based semantic markups.

As noted above, although several works on Semantic Web technologies have been proposed, there has been no focus on the practical issues of RDF and RDFS descriptions. Detailed semantics is required to capture the semantic correspondence between RDF and RDFS. Therefore, the semantics of higher-order models have been investigated in the field of computational logics. Furthermore, first-order formalization has been investigated to integrate meta-level concepts into layered first-order theory. In order to integrate the object-level and the meta-level concepts into the first-order theory, constructive mathematics [7] and meta logics [8], [9] provide the framework for the provability interpretation of logical formulas. Therefore, the unified semantics of the first-order framework and RDFs description management is required.

From the standpoint of the information lifecycle of the Semantic Web, an editing tool of RDF-based descriptions is developed in this work for the management of the relationship between RDF and RDFS descriptions. In order to maintain consistency among RDF descriptions, we adopt an approach based on the following framework of the layered first-order theory: reflection for meta-level reasoning [10].

The rest of this paper is structured as follows. Section 2 describes our concept of meta-model management and meta-model management scenarios. Section 3 describes the design of management facilities for RDF and RDFS. Section 4 presents an implementation of MR^3. Section 5 shows evaluations of MR^3. Then, we conclude and mention future work in Sect. 6.

2. Meta-Model Management

2.1 Concept of Meta-Model Management

This section discusses our concept of a meta-model and meta-model management. In this paper, meta-model is defined as a model expressing the components of models, especially the type of an RDF resource and RDF property. RDFS class is a model expressing the type of RDF resource. RDFS property is a model expressing an RDF property.

Manuscript received June 30, 2005.
Manuscript revised October 11, 2005.

[†]The authors are with Keio University, Yokohama-shi, 223–8522 Japan.
[††]The author is with the National Institute of Advanced Industrial Science and Technology, Tokyo, 135–0064 Japan.
[†††]The author is with Shizuoka University, Hamamatsu-shi, 432–8011 Japan.
a) E-mail: t_morita@ae.keio.ac.jp

DOI: 10.1093/ietisy/e89–d.4.1368

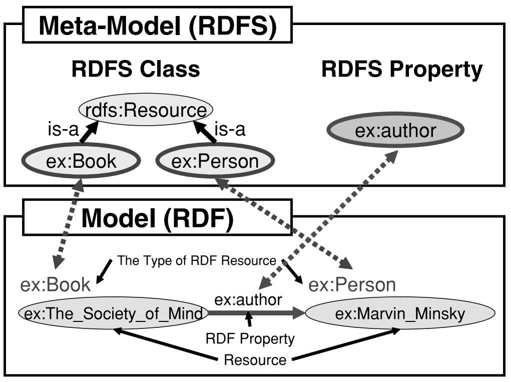


Fig. 1 Concept of meta-model management.

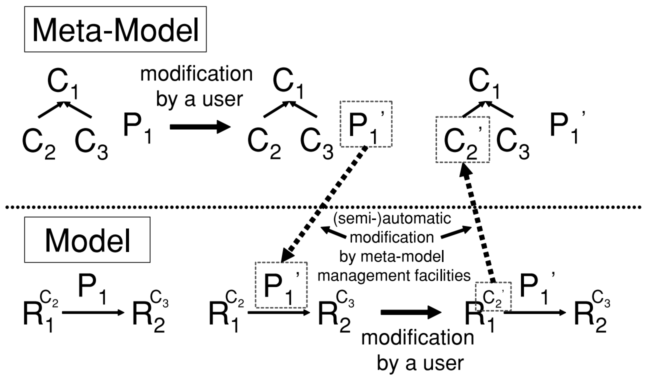


Fig. 2 An example of meta-model management process.

Since RDF and RDFS descriptions are described with the same RDF syntax, there is no clear distinction between RDF and RDFS parts. This forces us to observe the type of resources in order to pick up only an RDF part or only an RDFS part. Although the recent trend of Semantic Web languages, including Web Ontology Language (OWL) [11], tries to capture the advanced concept of ontologies in a single framework, there is no clear support to manage the correspondence between RDF and RDFS.

Since RDF and RDFS can be regarded as the relationship between a model and a meta-model, in the concept of meta-model management, RDF and RDFS can be managed separately and maintain their relationship automatically. In some of the logical frameworks described above, the meta-model concept seems to capture the above RDF and RDFS relationship. Such a logical framework reminds us to distinguish RDF and RDFS clearly, and can be expected to bring (semi-)automated support of the consistency between RDF and RDFS.

As a key concept in this paper, we focus on the meta-model management, which is defined as the appropriate management of the correspondence between a model and its meta-model: especially, the management of the RDFS class and RDFS property in a meta-model, and the type of an RDF resource and RDF property in a model. Figure 1 sketches the concept of meta-model management.

2.2 Meta-Model Management Scenarios

From our experience, we consider that the actual construction of meta-models and models in RDFS/OWL are performed from two different views. Meta-models are constructed considering a conceptualization which is separated from a real object world. On the other hand, models are constructed considering a real object world. Some modifications in models will cause the modifications of the meta-models, and vice versa. Moreover, in order to construct appropriate meta-models and models, modelers need to modify models and meta-models repeatedly. In such situation, since the modifications are made so often in both models and meta-models, these refinement costs are quite expensive. The graphical editing functions of *MR*^{3} will help modelers manage the model and the meta-model from a panoramic view. The meta-model management facilities will help modelers concentrate on editing and evaluating models or meta-models without frequently switching the editing modes between a model and a meta-model. The aim of our research is to provide such functions and facilities that will reduce the refinement cost of models and meta-models by the reflective processes.

Figure 2 shows an example of meta-model management process. The upper part of Fig. 2 shows a meta-model modification process. The lower part of Fig. 2 shows a model modification process. C_1, C_2, and C_3 in the meta-model depict RDFS classes. P_1 in the meta-model depicts an RDFS property. R_1 and R_2 in the model depict RDF resources. C_2 and C_3 which are depicted the upper right of RDF resources in the model depict type of RDF resources. P_1 in the model depicts an RDF property. In Fig. 2, at the beginning, P_1 in the meta-model is modified to P_1' by a user. Along with that, corresponding the RDF property in the model is also modified by a meta-model management facility automatically. Next, C_2 which is the type of RDF resource R_1 in the model is modified to C_2' by a user. Along with that, corresponding the RDFS class in the meta-model is also modified by a meta-model management facility semi-automatically. The detail of meta-model management facilities are described in Sect. 3.3.

3. Design of Management Facilities for RDFs

Figure 3 shows a functional outline of *MR*^{3}. The goal of *MR*^{3} is to represent complicated models in a form that is as easy as possible for users to understand. *MR*^{3} provides three main functions: (1) graphical editing of RDF descriptions, (2) graphical editing of RDFS descriptions, and (3) meta-model management facilities, allowing several types of relationships in an RDF and RDFS description to be manipulated and managed. Here, the RDF elements considered are the RDF resource, RDF property and RDF literal, and the RDFS elements are the RDFS class and RDFS property. In the definition of these functions, a data graph refers to any visual expression of the data model.

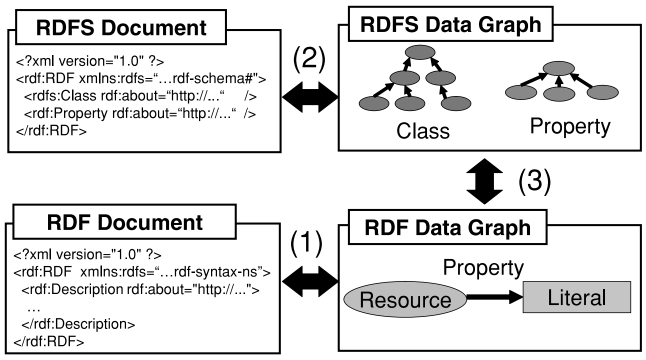

RDFS Document
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdfs="…rdf-schema#">
<rdfs:Class rdf:about="http://..." />
<rdf:Property rdf:about="http://..." />
</rdf:RDF>
(2)
RDFS Data Graph
Class
Property
(3)
RDF Document
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdfs="…rdf-syntax-ns">
<rdf:Description rdf:about="http://...">
...
</rdf:Description>
</rdf:RDF>
(1)
RDF Data Graph
Property
Resource
Literal

Fig. 3 *MR*^{3} functions.

3.1 Graphical Editing of RDF Descriptions

Function (1) in Fig. 3, which represents the graphical editing of RDF descriptions, supports the manipulation of the resource-property-value relation as defined by the semantics of the RDF data model. This function consists of the following sub-functions:

- Transformation of RDF resources into an RDF data graph
- Transformation of an RDF data graph into RDF resources

3.2 Graphical Editing of RDFS Descriptions

Function (2) in Fig. 3, which represents the graphical editing of RDFS descriptions, supports the manipulation of the attributes of classes such as the class-subclass relation and the attributes of properties such as `rdfs:domain`, `rdfs:range`, and the property-subproperty relation, as defined by the semantics of the RDF Schema model. This function consists of the following sub-functions:

- Transformation of RDFS resources into an RDFS data graph
- Transformation of an RDFS data graph into RDFS resources

3.3 Meta-Model Management Facilities

Function (3) in Fig. 3, which represents meta-model management facilities, is defined in this context as the checking of the consistency of classes and properties. The consistency checking mechanism consists of several facilities, as detailed below.

Meta-model management facilities are categorized as **O→M** or **M→O**. **O→M** is the facility to reflect the change in an ontology (RDFS class and property) in a model (the type of an RDF resource and RDF property). **M→O** is the facility to reflect the change in a model in an ontology. `Manipulation of RDFS Class` and `Manipulation of RDFS Property` are **O→M** facilities. `Replace the Type of an RDF Resource` and `Replace RDF Property` are **M→O** facilities.

3.3.1 Manipulation of an RDFS Class

The manipulation function of an RDFS class is operated by the meta-model management facility, and consists of replacing and removing an RDFS class.

- Replace RDFS Class
  When an RDFS class name is replaced, the type name of the RDF resource, which refers to the replaced RDFS class, is also replaced at the same time.
- Removal of RDFS Class
  When an RDFS class is removed, *MR*^{3} shows the list of RDF resources, which includes the removed RDFS class as a type. The user can choose (or empty) other RDFS classes as a type of RDF resource.

3.3.2 Manipulation of an RDFS Property

The manipulation function of an RDFS property is operated by the meta-model management facility, and consists of replacing and removing an RDFS property.

- Replace RDFS Property
  When an RDFS property name is replaced, the RDF property, which refers to the replaced RDFS property, is also replaced at the same time.
- Removal of RDFS Property
  When an RDFS property is removed, *MR*^{3} shows the list of the RDF properties which refer to the removed RDFS property. A user can choose other RDFS properties (or the default property - `mr3:nil`) as a property of the RDF resources.

3.3.3 Replacing the Type of an RDF Resource

When it is not clear which RDFS class corresponds to the type of an RDF resource replaced by the user, the meta-model management facility is applied. When the type of an RDF resource replaced by the user is defined by the RDFS class, *MR*^{3} matches the type of the RDF resource and the RDFS class corresponding to the type of RDF resource. In addition, if the class is not defined, the user can choose one of the following:

- Replace the RDFS class name with that referred to before the user replaced the type of the RDF resource.
- Create a new RDFS class that has yet to be defined.

3.3.4 Replacing the RDF Property

When it is not clear which RDF property corresponds to the

RDFS property replaced by a user, the meta-model management facility is applied. When the RDF property which the user replaced is defined by the RDFS property, *MR*^{3} matches the RDF property and the RDFS property corresponding to the RDF property. If the property is not defined, the user can choose one of the following:

- Replace the RDFS property name with that referred to before the user replaced the RDF property.
- Create a new RDFS property that has yet to be defined.

3.3.5 Importing an RDF Document

When importing an RDF document, the type of RDF resource or an RDF property may not be defined as an RDFS class or an RDFS property. In this case, in order to maintain consistency, a type of RDF resource which is not defined as an RDFS class is created as a sub class of the `rdfs:Resource` class. In the same way, an RDF property which is not defined as an RDFS property is created.

Figure 4 shows an example of importing an RDF document. The left side of Fig. 4 depicts the state before importing the RDF document. The right side of Fig. 4 depicts the state after importing the RDF document. `ex:Book`, with the type of `ex:The_Emotion_Machine` and `ex:The_Society_of_Mind`, is not defined as an RDFS class. Also `ex:author` in the RDF model is not defined as an RDFS property. In order to maintain consistency, *MR*^{3} creates a `ex:Book` class and `ex:author` property in the RDFS data graph automatically.

3.4 Other Functions

`Keeping Element Names Unique`
This function prevents RDF and RDFS from overlapping other element names when a user renames and creates an RDF or RDFS element. If duplication of an RDFS element name is allowed, consistency cannot be maintained.

`Setting a Meta Class and Property`
A user can set meta classes and properties in *MR*^{3}. This function controls whether to consider a resource of a certain type as a class or a property. For example, if a user sets `owl:Class` as a meta class and `owl:ObjectProperty` and `owl:DatatypeProperty` as meta properties, *MR*^{3} can import the class and property hierarchy in OWL. The default meta class is `rdfs:Class` and the default meta property is `rdf:Property`.

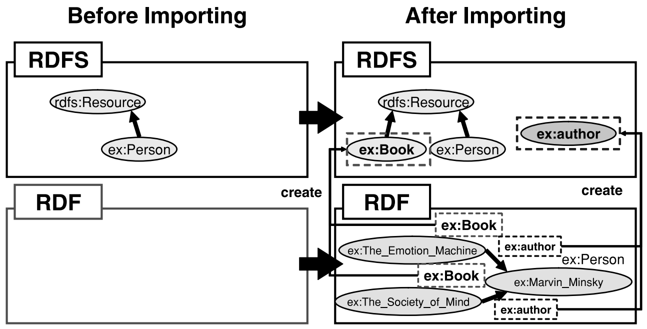


Fig. 4 An example of importing an RDF document.

`Validation`
When constructing an RDF model, *MR*^{3} doesn't check `rdfs:domain` and `rdfs:range` in the RDFS properties. However, *MR*^{3} can perform validation of an RDF model using vOWLidator [12]. This function indicates the resources that don't match the `rdfs:domain` and `rdfs:range` in the RDFS properties.

4. Implementation

4.1 Implementation Architecture of *MR*^{3}

Figure 5 shows the system architecture of *MR*^{3} from the aspect of system implementation. *MR*^{3} is implemented in Java language, using the Java Swing user interface. *MR*^{3} uses JGraph [13] for RDF(S) graph visualization, and Jena - A Semantic Web Framework [14] for enabling the use of Semantic Web standards such as RDF, N3, N-triple, RDFS, and OWL. The Parser and Generator in *MR*^{3} are implemented using Jena APIs. By using these libraries, *MR*^{3} is implemented as an environment for graphical representation of Semantic Web descriptions. Additionally, *MR*^{3} also has a plug-in facility to extend its functionality. At present, *MR*^{3} offers two kinds of APIs for plug-in development: one for changing the *MR*^{3} data graph into a model object of Jena, and one for changing the model object of Jena into a *MR*^{3} data graph. In the future, *MR*^{3} will also offer APIs for creating a greater variety of plug-ins, such as an API for managing the consistency between RDF and RDFS more strictly.

4.2 System Overview of *MR*^{3}

Figure 6 shows a system overview of *MR*^{3}. *MR*^{3} consists of the Parser module, Generator module, Meta-Model Management module, Plug-ins, and User Interface. The user edits the RDFs description visually via the User Interface, which also includes the Graphical Modeler and Plug-in interfaces. The Graphical Modeler provides access to the basic functions of *MR*^{3}, while the Plug-in Interface provides access to the functions of the plug-ins. The input and output of *MR*^{3} are RDFs documents. The Parser analyzes input

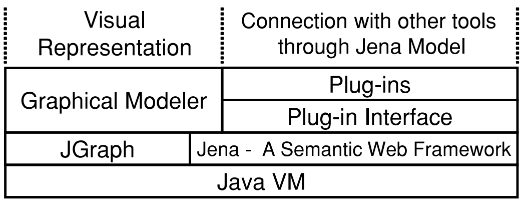

Visual Representation
Connection with other tools through Jena Model
Graphical Modeler
Plug-ins
Plug-in Interface
JGraph
Jena - A Semantic Web Framework
Java VM

Fig. 5 Implementation architecture of *MR*^{3}.

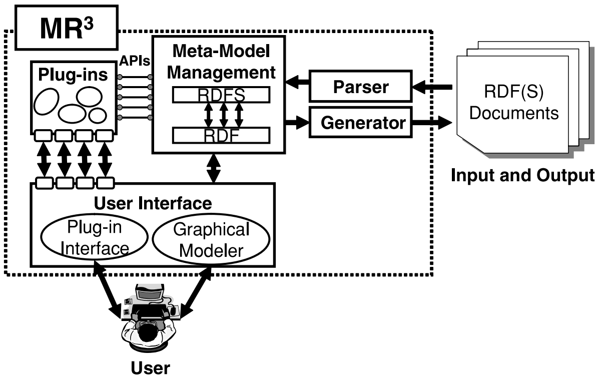


Fig. 6 System overview of MR^3.

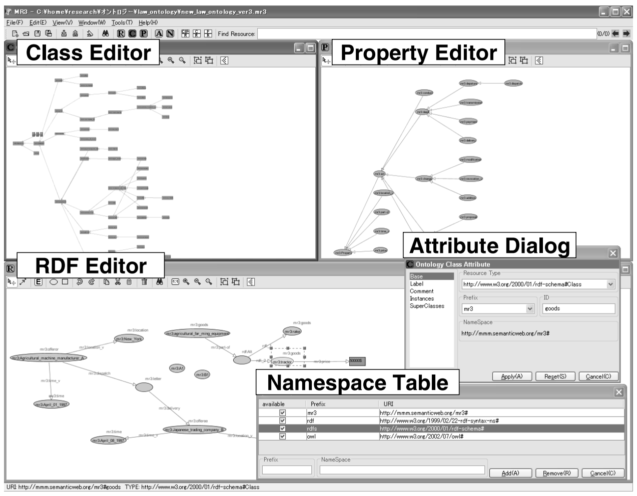


Fig. 7 Typical screen with Graphical Modeler interface of MR^3.

RDFs documents and makes further operations possible by transforming the RDFs document into a Jena model. Then, the Parser changes the Jena model into an internal data expression, and meta-model management is performed. Plug-ins are built using APIs provided by MR^3. The Generator changes the internal data expression into a Jena model. Finally, the Jena model is changed into an RDFs document.

4.3 Detailed Implementation of MR^3

Figure 7 shows a typical screen showing the Graphical Modeler interface of MR^3. The Graphical Modeler consists of five main windows; RDF Editor, Class Editor, Property Editor, Attribute Dialog, and Namespace Table. The RDF Editor allows the user to express the relationship between an RDF resource, RDF property, and RDF literal using a directed graph, and also allows the attributes of each element to be edited. The attributes of an RDF resource consist of a URI, the URI type, and the RDF resource type. The RDF resource type can be chosen using the Class Editor. The URI type can be chosen from either a URI or can be set as anonymous. The Class Editor allows the user to express the relationship between RDFS classes, if one exists, and edit the attributes of an RDFS class. The attributes of an RDFS class consist of a URI, `rdfs:label`, and `rdfs:comment`. The user can refer to a super class of the selected class and the list of RDF resources that consider the class. The attributes of an RDFS property consist of a URI, `rdfs:label`, `rdfs:comment`, `rdfs:domain`, and `rdfs:range`. Through the Property Editor also, the user can express the relationship between RDFS properties, if one exists, and edit the attribute of an RDFS property. The parameters `rdfs:domain` and `rdfs:range` can be chosen from the Class Editor. The user can refer to a super property of the selected property and the list of RDF resources that have the property. The attribute of each element can be displayed and edited via the Attribute Dialog, and the user can replace a namespace with a prefix via the Namespace Table. In order to support the graphical display of the RDF model, the JGraph [13] library is linked into the User Interface module.

5. Evaluation

5.1 Comparison with Other Related Tools

In this section, we clarify our study and the differences in other related studies. Recently, numerous commercial and open-source ontology development tools have become available along with the standardization of Semantic Web technologies (e.g., RDF, RDFS, OWL). In [15], the survey results of 94 ontology editors currently available to the ontology building community are presented. An early version of the MR^3 is also introduced in the survey results. We selected 5 major related tools from the survey results and compared them with our tool.

Table 1 shows a comparison of our study (MR^3) with other related studies. The functions which are shown in Table 1 are necessary to achieve our aim to reduce the refinement cost of models and meta-models by the reflective processes. The necessity of all these functions depends on the complexity of problem structure and the conceptualization by human experts in advance. Each item in Table 1 is evaluated subjectively from our experience. Since there are many methods to realize each function in Table 1, only the tools which have the functions to achieve our aim are checked.

The benefit of MR^3 is having the graphical facilities with RDF models and meta-model management facilities. The tools in Table 1 are categorized as RDF-based tools or ontology-based tools. IsaViz [16], RDFAuthor [17], and MR^3 are categorized as RDF-based tools, and they concentrate on constructing RDF models. Therefore, they have the enhanced graphical facilities with RDF models. KAON OI-modeler [18], OntoEdit [5], and Protégé OWL Plugin [6] are categorized as ontology-based tools, and they concentrate on constructing and managing ontologies. Therefore, they either have partial graphical facilities with RDF models or they don't have any. IsaViz and KAON OI-modeler mix and display RDF and RDFS models. Since it is difficult for the

Table 1 A comparison of MR^3 with other related studies.

	Name	**Source**	**Graphical Facilities with RDF Models**	**Meta-Model Management Facilities**		**Easy to Define RDF Properties**	**Import/Export/Edit**	
				O→M	**M→O**		**RDF**	**RDFS**
RDF based	IsaViz	W3C	[1]			✓	✓	✓
	RDFAuthor	Demian Steer	✓			✓	✓	[4]
	MR^3	Keio University, Shizuoka University, and AIST	✓	✓	✓	✓	✓	✓
Ontology based	KAON OI-modeler	FZI Research Center and AIFB Institute, University of Karlsruhe	[1]	✓			[3]	✓
	OntoEdit	Ontoprise GmbH	[2]	✓			[3]	✓
	Protégé OWL Plugin	Stanford Medical Informatics, Stanford University		✓			[3]	✓

[1] These tools do not support displaying an ontology and a model separately. [2] OntoEdit does not support editing RDF elements while displaying the whole RDF model. [3] These tools do not support treating anonymous resources. [4] RDFAuthor does not support editing and exporting RDFS models.

user to distinguish RDF elements and RDFS elements, the burden of the user in RDF model construction is substantial. RDFAuthor, MR^3, and OntoEdit can display an ontology and a model separately. The graphical facilities with RDF models are demonstrated in Sect. 5.3.

O→M and **M→O** in Table 1 are the sub facilities for meta-model management. **O→M** is the facility to reflect the change in an ontology (RDFS class and property) in a model (the type of RDF resource and RDF property). **M→O** is the facility to reflect the change in a model in an ontology. IsaViz and RDFAuthor don't have the meta-model management facility. Ontology-based tools have the facility of **O→M**. However, they don't have the facility of **M→O**. The benefit of MR^3 is having an RDF-based tool and the meta-model management facilities. In particular, **M→O** is an original facility of MR^3.

Easy to Define RDF Properties in Table 1 means whether the tool can define RDF properties (a relationship between RDF resources) without defining the values of `rdfs:domain` and `rdfs:range`. In ontology-based tools, it is necessary to define the values of `rdfs:domain` and `rdfs:range` beforehand in order to define the RDF properties. RDF-based tools can define RDF properties without defining the values of `rdfs:domain` and `rdfs:range`. For RDF model construction, it is considered that the user should be able to define RDF properties freely.

There are two methods of constructing RDF models. One is the method to construct RDF models using predefined ontologies. The other is the method to construct RDF models without referring to ontologies. A user can construct RDF models by the latter method using the facility of **M→O** and **Easy to Define RDF Properties**. Thanks to the **M→O** facilities, when a user chooses classes or properties which are not defined in ontologies in order to construct RDF models, the classes or properties are defined in the ontologies automatically.

Import/Export/Edit in Table 1 shows the functions of import, export, and edit for RDF and RDFS descriptions. In these functions, the handling of an RDF is the main difference between RDF-based tools and ontology-based tools. Ontology-based tools do not support treating RDF models, including anonymous resources. Anonymous resources are often used to describe groups of things. Since most of the existing RDF descriptions (e.g., RSS [19]) include anonymous resources, treating anonymous resources is important for constructing RDF models. RDF-based tools can treat anonymous resources. The tools in Table 1 can treat RDFS models. RDFAuthor can import RDFS descriptions. However, RDFAuthor does not support the editing and exporting of RDFS models.

5.2 The Effectiveness of Meta-Model Management Facilities

In order to evaluate the effectiveness of the meta-model management facilities, two kinds of experiments were conducted. In the experiments, the user is knowledgeable about RDF, RDFS, and the usage of MR^3; however, the user does not have enough domain knowledge about ontologies and models to construct them in these experiments (The user is not a domain expert). In experiment1, the user constructs an ontology and a model for a car sales domain using MR^3, which has meta-model management facilities. The user refers to the workflow of car sales, which includes 862 words in Japanese. In experiment2, the user constructs an ontology and a model for a mail-order fruits domain using a modified version of MR^3 which has none of the facilities of meta-model management. The user refers to the workflow of mail-order fruits, which includes 1245 words in Japanese. To observe the user's actual operations during the experiments, we added an operation logging function to both versions of MR^3.

Tables 2 through 6 show the results of these experiments. Table 2 shows a comparison of the number of con-

Table 2 Comparison of the number of constructed statements in experiment1 and experiment2.

	Experiment1	**Experiment2**
# RDFS Classes	18	21
# RDFS Properties	16	16
# Statements (RDFS Part)	65	79
# Statements (RDF Part)	31	26
# Statements (All)	96	105

Table 3 Comparison of the elapsed time for a user to complete a task in each experiment.

Experiment1	**Experiment2**
1660 sec	2520 sec

Table 4 Comparison of the number of executed operations in experiment1 with the number of executed operations in experiment2.

	Experiment1	**Experiment2**
# Operations for Model	83	69
# Operations for Ontology	49	96
# Total Operations	132	165

Table 5 Number of operations using meta-model management facilities in experiment1.

Meta-Model Management Facilities		**#**
Type	**Name**	
O→M	Replace RDFS class	0
	Replace RDFS property	2
M→O	Replace type of RDF resource (create)	4
	Replace type of RDF resource (rename)	2
	Replace RDF property (create)	15
	Replace RDF property (rename)	0

Table 6 Comparison of the number of switching the editing modes in experiment1 and experiment2.

	Experiment1	**Experiment2**
# switching from the model editing mode to the meta-model editing mode	13	34
# switching from the meta-model editing mode to the model editing mode	12	34
# total switching editing modes	25	68

structed statements in experiment1 and experiment2. Table 3 shows a comparison of the elapsed time for the user to complete the task in each experiment. Table 4 shows a comparison of the number of executed operations in experiment1 with the number of executed operations in experiment2. The operations in Table 4 include insertion, editing, and deletion of RDF resources, RDFS classes, and RDFS properties. Since the operations for **O→M** are performed automatically, we do not count the operations for **O→M** in Table 4. But, the operations for **M→O** are performed semi-automatically; therefore, we count the operations for **M→O** in Table 4. Table 5 shows the number of operations using the meta-model management facilities in experiment1. Table 6 shows a comparison of the number of switching the editing

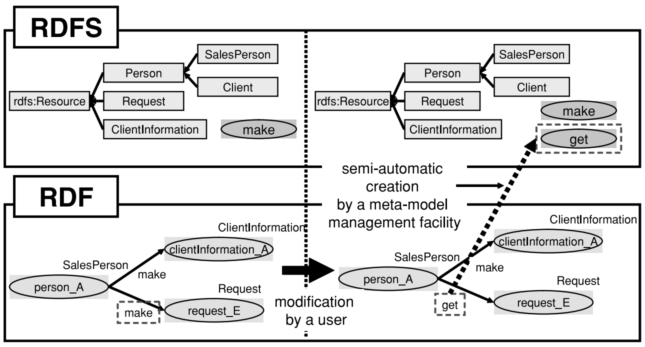


Fig. 8 An example of using a meta-model management in experiment1.

modes in experiment1 and experiment2.

According to Table 2, the size of the ontologies and models constructed in experiment1 was approximately the same size in experiment2. According to Table 3, the user could construct models and ontologies in a shorter time using the meta-model management facilities. According to Table 4, the meta-model management facilities reduced the number of operations to construct the models and ontologies. According to Table 6, the meta-model management facilities reduced the number of switching the editing modes between the model and the meta-model.

Figure 8 shows an example of using a meta-model management facility in experiment1. The upper part of Fig. 8 shows an RDFS model and the lower part of Fig. 8 shows an RDF model. The RDF and RDFS models in Fig. 8 are extracted a part related to use situation of a meta-model management facility from the whole constructed RDF and RDFS models in experiment1. Rectangles in the RDFS model depict RDFS classes. Ellipses in the RDFS model depict RDFS properties. Ellipses in the RDF model depict RDF resources. Labeled arcs in the RDF model depict RDF properties. The upper right of RDF resources depicts the type of RDF resources. In this example, the RDF property `make` which was the relation between `person_A` resource and `request_E` resource was modified to `get` by the user. At that time, since RDFS property `get` was not defined in RDFS model, MR^3 asked the user whether renaming RDFS property `make` or creating `get`. Here, since the user would like to keep the relation `make` between `person_A` resource and `clientInformation_A` resource, the user select creating `get` property. As a result, `get` property was created in the RDFS model by a meta-model management facility semi-automatically.

Since ontology design is a creative process and no two ontologies designed by different people would be the same [20], evaluation of the ontology and model development tools is difficult. The experimental results depend on the user's skills, such as domain knowledge and proficiency using the tools. There is little statistical data on constructing existing ontologies and models at the operation level. Moreover, it is difficult to evaluate the quality of a constructed model and ontology. However, with these experiments, the effectiveness of meta-model management facilities can be seen from the viewpoint of reducing the time for construction, the number of operations, and the number of switching the editing modes between the model and the meta-model.

5.3 Graphical Facilities with RDF Models

Here, we show the differences of the graphical facilities of each tool in Table 1 with respect to RDF models. As experimental data, we use the schema of the RDF Site Summary [†], RDFS description of the RDF vocabulary [††], and RSS data of the Web content in our laboratory. Table 7 shows the number of resources, literals, and statements in the experimental data. Table 8 shows the number of classes and properties in the experimental data. The above data with intermingled RDF and RDFS descriptions were imported into the tools in Table 1. Protégé OWL Plugin didn't have the graphical facilities with RDF models; therefore, we didn't test Protégé OWL Plugin in this experiment. Since anonymous resources were included in the experimental data, KAON OI-modeler and OntoEdit could not import the data. In the following, we compare the graphical facilities with RDF models in IsaViz, RDFAuthor, and *MR*^{3}.

Figure 9 shows a screenshot of IsaViz while importing the experimental data. It is difficult to understand the hierarchical relationships between RDFS elements from the RDF graph produced by IsaViz. Since RDF and RDFS elements are displayed in the same color of ellipse or as rectangle nodes in IsaViz, the user needs to search for resources that have the property of `rdf:type` with values of `rdfs:Class` or `rdf:Property` in order to differentiate the RDFS elements. Figure 10 and Figure 11 show screenshots of RDFAuthor and *MR*^{3}, respectively, while importing the experimental data. RDFAuthor and *MR*^{3} display RDF elements and RDFS elements separately, making it easier for the user to distinguish RDF elements from RDFS elements. RDFAuthor shows the list of RDFS classes and properties in each namespace. However, RDFAuthor does not support showing hierarchical relationships in the RDFS classes and properties. In addition, RDFAuthor does not support editing the RDFS classes and properties. In contrast, *MR*^{3} can show hierarchical relationships in the RDFS classes and properties and edit them.

The comparisons among IsaViz, RDFAuthor, and *MR*^{3} show that the graph of RDFAuthor and *MR*^{3} is much easier to understand than that of IsaViz. One reason for this improvement is the separation of the RDF and RDFS elements. Especially, *MR*^{3} hides unnecessary elements from the user, which also assists in understanding the graph. For example, in *MR*^{3}, the attributes of the RDF resources, such

Table 7 Number of RDF and RDFS elements in the experimental data.

	# Resource	# Literal	# Statement
RDF Part	15	35	59
RDFS Part	34	46	131
Total	49	81	190

Table 8 Number of classes and properties in the experimental data.

# Class	# Property
13	19

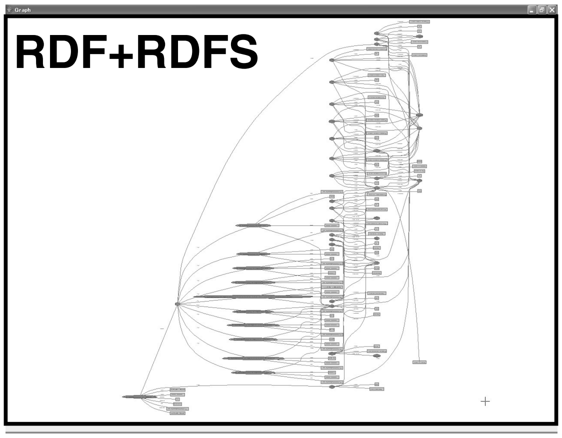
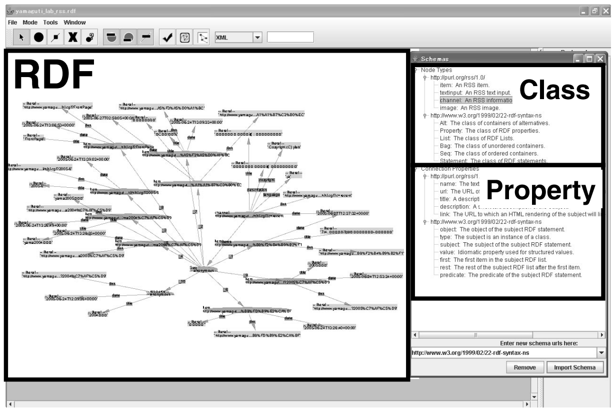


Fig. 9 RDF representation by IsaViz.



Node Types
http://purl.org/rss/1.0/
item: An RSS item.
textinput: An RSS text input.
channel: An RSS informatio
image: An RSS image.
http://www.w3.org/1999/02/22-rdf-syntax-ns
Alt: The class of containers of alternatives.
Property: The class of RDF properties.
List: The class of RDF Lists.
Bag: The class of unordered containers.
Seq: The class of ordered containers.
Statement: The class of RDF statements.
Connection Properties
http://purl.org/rss/1.
name: The text
url: The URL o
title: A descript
description: A
link: The URL to which an HTML rendering of the subject will li
http://www.w3.org/1999/02/22-rdf-syntax-ns
object: The object of the subject RDF statement.
type: The subject is an instance of a class.
subject: The subject of the subject RDF statement.
value: Idiomatic property used for structured values.
first: The first item in the subject RDF list.
rest: The rest of the subject RDF list after the first item.
predicate: The predicate of the subject RDF statement.
http://www.w3.org/1999/02/22-rdf-syntax-ns

Fig. 10 RDF representation by RDFAuthor.

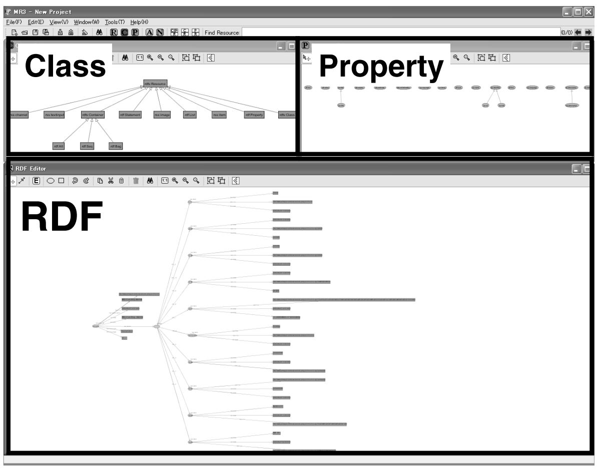


Fig. 11 RDF representation by *MR*^{3}.

[†]http://web.resource.org/rss/1.0/schema.rdf
[††]http://www.w3.org/TR/rdf-schema/rdfs-namespace

as `rdfs:label`, `rdfs:comment` which are mainly provided for human beings, are hidden but can be edited via the `Attribute Dialog`. In a similar manner, the attributes of the RDFS classes and properties, such as `rdfs:label`, `rdfs:comment`, `rdfs:domain`, and `rdfs:range`, are hidden. By hiding unnecessary elements, the burden of information on the user when many RDF resources refer to the RDFS class as a type is reduced, allowing the user to concentrate on the most relevant resources.

5.4 MR^3 as an Open Source

In order to open our technology to the Semantic Web community, the MR^3 system and its source code are provided via our Web site [21]. At present, there are about 300 user registrations. Some users have provided comments about MR^3. Some users selected MR^3 since MR^3 is compliant with RDFS-guided annotation and is not strictly frame-oriented. Some users requested the capability to treat reification and several thousand classes. We would like to reflect these comments in our future work. In addition, some users developed a plug-in to handle a container model (e.g. `rdf:Bag`) and to connect Sesame [22].

6. Conclusions

From the standpoint of meta-model management, we have developed an RDFs description management tool: MR^3 (Meta-Model Management based on RDFs Revision Reflection). Since RDF and RDFS can be regarded as the relationship between a model and a meta-model, MR^3 enables us to manage RDF and RDFS separately and to maintain their relationship automatically.

We have focused on meta-model management as the key concept in this paper. Meta-model management is defined as the appropriate management of the correspondence between a model and its meta-model: especially, the management of the class and property in the meta-model, and the type of an RDF resource and a property in the model. At the same time, we have developed a prototyping system of MR^3 using Java language and evaluated it in an experimental study by a comparison of other related tools. Through the case study of the use of MR^3, we have confirmed that our proposed tool contributes to the Semantic Web paradigm, which is focused on the technological basis of RDFs.

Finally, we are developing an additional plug-in module for MR^3 corresponding to our other work [23]. In [23], we propose a domain ontology development environment called DODDLE-OWL (Domain Ontology rapiD DeveLopment Environment - OWL extension). DODDLE-OWL generates a domain ontology semi-automatically in OWL format. At present, MR^3 supports OWL partially (only the class and property hierarchies). In the future, we'd like to support OWL Lite level ontologies and its model by integrating MR^3 and DODDLE-OWL.

References

[1] O. Lassila and R.R. Swick, "Resource Description Framework(RDF) Model and Syntax Specification," 1999, http://www.w3.org/RDF/
[2] D. Brickley and R. Guha, "RDF Vocabulary Description Language 1.0: Rdf Schema," http://www.w3.org/TR/rdf-schema/
[3] T. Berners-Lee, "Notation 3: Ideas aboutweb architecture - yet another notation," 2001, http://www.w3.org/DesignIssues/Notation3
[4] D. Beckett and A. Barstow, "N-triples," 2001, http://www.w3.org/2001/sw/RDFCore/ntriples/
[5] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke, "Ontoedit: Collaborative ontology engineering for the semantic web," The First International Semantic Web Conference, vol.2342 of LNCS, pp.221–235, 2002.
[6] H. Knublauch, R.W. Fergerson, N.F. Noy, and M.A. Musen, "The protégé owl plugin: An open development environment for semantic web applications," Third International Semantic Web Conference, pp.229–243, 2004.
[7] M. Beeson, Foundations of constructive mathematics, Springer-Verlag, 1985.
[8] N. Davies, "A first order logic of truth, knowledge and belief," LNAI-478, pp.170–179, 1980.
[9] R. Turner, Truth and modality for knowledge representation, MIT Press, 1991.
[10] G. Attardi and M. Simi, "Reflections about reflection," in Principles of Knowledge Representation and Reasoning (KR-91), pp.22–31, Morgan Kaufmann, 1991.
[11] M.K. Smith, C. Welty, and D.L. McGuinness, "Owl Web Ontology Language Guide," http://www.w3.org/TR/owl-guide/
[12] BBN, "vowlidator," http://owl.bbn.com/validator/
[13] G. Alder, "Jgraph," http://www.jgraph.com
[14] HP Labs, "Jena Semantic Web Framework," http://jena.sourceforge.net/downloads.html
[15] M. Denny, "Ontology tools survey, revisited," http://www.xml.com/pub/a/2004/07/14/onto.html
[16] E. Pietriga, "Isaviz, a visual environment for browsing and authoring rdf models," Eleventh International World Wide Web Conference Developers Day, 2002, http://www.w3.org/2001/11/IsaViz/
[17] D. Steer, L. Miller, and D. Brickley, "Rdfauthor: Enabling everyone to author rdf," Eleventh International World Wide Web Conference Developers Day, 2002, http://rdfweb.org/people/damian/RDFAuthor/
[18] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, "An infrastructure for searching, reusing and evolving distributed ontologies," Proc. 12th Int. World Wide Web Conference (WWW'03), pp.439–448, ACM Press, 2003.
[19] R.S.S..S.W. Group, "Rdf site summary (rss) 1.0," 2001, http://web.resource.org/rss/1.0/spec
[20] N.F. Noy and D.L. McGuinness, "Ontology development 101: A guide to creating your first ontology," Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.
[21] "Web site *mr*^{3}," http://mmm.semanticweb.org/mr3/
[22] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A generic architecture for storing and querying rdf and rdf schema," The First International Semantic Web Conference, http://www.openrdf.org/
[23] T. Morita, Y. Shigeta, N. Sugiura, N. Fukuta, N. Izumi, and T. Yamaguchi, "Doddle-owl: Owl-based semi-automatic ontology development environment," Evaluation of Ontology-based Tools, 2004, http://mmm.semanticweb.org/doddle/

Takeshi Morita is a graduate student at the school of Science and Technology at Keio University. He received B.E. and M.E. degrees in Computer Science from Shizuoka University in 2003 and 2005. His research interests include Ontology Engineering and Semantic Web.

Noriaki Izumi is a researcher at the National Institute of Advanced Industrial Science and Technology. He received B.E. and M.E. degrees from Osaka Prefecture University in 1992 and 1994. He received a Ph.D. degree from Keio University. His research interests include Intelligent Systems, Knowledge Modeling, Semantic Web, and Web Services. He is a member of JSSST, IPSJ, and JSAI.

Naoki Fukuta is a research associate at the Faculty of Informatics at Shizuoka University. He received B.E., M.E., and Ph.D. degrees from Nagoya Institute of Technology in 1997, 1999, and 2002, respectively. His research interests include Software Engineering, Semantic Web, and WWW-based Intelligent Systems. He is a member of IEEE-CS, ACM, JSAI, IPSJ, and JSSST.

Takahira Yamaguchi is a professor at the Faculty of Science and Technology at Keio University. He received his B.E., M.E., and Ph.D. degrees in telecommunication engineering from Osaka University in 1979, 1981, and 1984, respectively. His research interests include Ontology Engineering, KBSE, Advanced Knowledge Systems, and Machine Learning. He is a member of IPSJ, JSAI, JSFTS, JCSS, AAAI, IEEE-CS, and ACM.