NESSTAR: A Semantic Web Application for Statistical Data and Metadata

Pasqualino 'Titto' ASSINI (titto@nesstar.com)
Nesstar Ltd - U.K.

Abstract

NESSTAR is a Semantic Web application for statistical data and metadata that aims to streamline the process of finding, accessing and analysing statistical information. The paper describes the rationale behind the design of the NESSTAR system and, more in general, the steps involved in the design and development of a typical Semantic Web application.

Keywords: Statistical Data and Metadata, Data Web, Semantic Web, Web Services, Integration of J2EE and RDF

1 Statistical Data Dissemination: The Reality and the Dream

The social sciences are big producers and consumers of statistical data. Surveys, censuses and opinion polls are basic sources of information for most quantitative research in sociology and economics. These data are collected and preserved by specialised data archives, national statistical offices or private research institutes (e.g. Gallup) and traditionally disseminated to researchers (social researchers, journalists, marketing experts, etc.) as datafiles stored on some form of magnetic media and accompanied by bulky printed documentation (metadata).

From the point of view of a researcher the current statistical data dissemination process is far from optimal. Given that there are many data publishers, each one with its own distinct access and dissemination procedures, it is often not easy to find and get hold of the right information. Also the artificial separation between statistical data and metadata complicates the assessment and processing of the information.

In 1998 the European Union funded a research and development project named Networked Social Science Tools and Resources (NESSTAR) [3] [14]. The aim of the project was to bring the advantages of the Web to the world

of statistical data dissemination. At the time the WWW had already made the publishing of textual and graphical information easier and cheaper than ever. Huge amount of information had been made available worldwide at a press of a button, at virtually no cost and in a highly integrated way. The question that NESSTAR was called to answer was if it was possible to create a "Data Web" that would make just as easy to publish, locate and access statistical data.

The NESSTAR project has been followed by FASTER [2], another European project that has further developed the Data Web concept and implementation. NESSTAR and FASTER have been sufficiently successful to convince two of their main contractors, the University of Essex in England and the University of Bergen in Norway, to spin off a company to exploit commercially the Data Web technology. Nesstar Ltd is currently busy developing Data Web solutions for a number of international clients.

2 The Semantic Web and the Data Web

The basic aim of NESSTAR is to make available, in an easily accessible way, a great quantity of statistical data and metadata that is currently locked in incompatible or human understandable only formats. If this objective could be achieved it would revolutionize the way people access statistical information just like the World Wide Web has already revolutionised access to other kinds of information.

The WWW has recently known a major evolution with the appearance of a set of new technologies that go under the name of the Semantic Web (SW):

> *The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. [12].*

The necessity of representing statistical metadata and data in a way that is machine understandable makes NESSTAR a perfect candidate for the application of Semantic Web technologies. The NESSTAR "Data Web" is therefore the application of Semantic Web techniques and principles to the problem of distributed data dissemination and processing.

The *modus operandi* of NESSTAR is very simple. Data publishers make their statistical information available as objects, as specified by the NESSTAR object model of statistical data and metadata, on the Net. The system is fully distributed: each publisher runs its own server.

Users use the system pretty much as they use the Web: if they know where some information is stored they can "point" their client application

to it (for example by typing the object URL in a location bar or by clicking on a hyperlink). The client will access the remote statistical object and display it to the user. The users can also perform searches to find objects with particular characteristics such as: "find all variables about political orientation". This is similar to using a search engine such as Google to find all HTML pages that contain a given keyword.

The NESSTAR system is built on top of a lightweight Web and object-oriented middleware, the NEsstar Object Oriented Middleware (NEOOM). NEOOM is closely based on Web and Semantic Web standards, in particular RDF [6], RDF Schema [11], HTML and HTTP. So closely, actually, that more than a distinct framework it can be considered as just a set of guidelines on how to use off the shelf (semantic) web technology to build distributed object-oriented systems.

In the following sections I will discuss the major steps in the design and development of a typical Semantic Web Application (SWA) using NESSTAR as a concrete full-scale example. More detailed technical information on NEOOM is available in [8] and in [7].

3 Choosing a Modelling Language

At the core of any SWA there is a formal model of the application domain. The application domain model must necessarily be expressed in a modelling language. The choice of this modelling language is the first significant step in the SWA design process.

The standard SW modelling language is RDF Schema [11] (in the future this might be superseded by a more sophisticated schema language: the Web Ontology Language (WOL) [4]). RDF Schema is an useful tool but as a modelling language has some significant shortcomings:

- limited expressive power (e.g., no relationships, no operations)
- lack of modelling tools (such as graphical editors for RDF Schema models)

For many SWAs a better choice would be the Unified Modelling Language (UML). UML is a standard and expressive modelling language based on the object-oriented paradigm. It is well documented and good design tools (both open source and commercial) are available. UML is very extensive and complex but this complexity should not be a deterrent for its adoption as in most cases only a modest subset of the language will be needed to model the application model of a SWA. For example the NESSTAR domain model has been defined using just a simple UML Class Diagram.

4 Modelling the Application Domain

In the case of NESSTAR the application domain model is an object-oriented model of statistical data and metadata[1]. Though not very extensive, it currently contains about 15 classes, it captures many of the key domain concepts: statistical studies, datafiles, statistical variables, indicators, tables, etc. The classes are linked together by a set of relationships: a Study for example contains Cubes each having one or more Dimensions, etc. Some objects have also methods. Statistical Studies for example have methods such as Tabulate or Frequency and other common statistical operations.

In addition to the domain specific concepts the model includes another 10 or so domain independent "support" classes. An example is the Server class. It represents the server where the objects are hosted and provides basic administrative functions such as file transfer, server reboot and shutdown, etc. It also plays, in the Data Web, a role similar to that of the home page in the normal Web. From a home page one can, by following hyperlinks, reach all the contents of a web site. Similarly, starting from a server object an application can, by recursively traversing the object relationships, reach all the other objects hosted by the server. Another generic class is Catalog. Catalogs are used to group objects, just like folders in a filesystem. Catalogs can be browsed, an application can get the complete list of all the objects contained in a catalog, or searched to select only the objects that satisfy a particular condition.

Many statistical studies contain sensitive information that cannot be made available without restrictions. For this reason the model includes a set of objects to represent Users, the Roles they play in the system (example: Administrator, EndUser, DataPublisher), the agreements that they have accepted (such as: "I agree to use these data only for non commercial research purposes"), etc. On top of this small security model is possible to define a variety of access control policies[2].

The domain independent classes of the model are particularly interesting as they can be reused by different SWAs. Hopefully in the near future we will see the emergence of reusable 'libraries' ("ontologies" in SW-speak) of domain independent concepts. That will greatly speed up the process of modelling and implementing new SWAs.

[1] For reasons of space the object model class diagram is not included in this paper. It can be found at http://www.nesstar.org/sdk/nesstar_object_model.pdf

[2] NESSTAR servers support a wide range of Access Control mechanisms.

5 Publishing the Model

Once the model is defined it is useful to make it available on the Net. By doing so it becomes possible to build highly generic applications that discover object characteristics (properties, relationships, methods, etc. as specified by the object class definition) dynamically at run-time. For NESSTAR we have developed one such tool: the Object Browser, a web-based generic client used for object testing and administration. The Object Browser can be used to display and operate on any NESSTAR object.

To publish the model on the SW it has to be converted to an equivalent model expressed in the SW standard modelling language: RDF Schema. Mapping a UML Class Diagram, or other simple object-oriented formalisms, to RDF Schema is relatively straightforward. RDF Schema provides all the basic object-oriented primitives: classes, properties and inheritance. There is only one major omission: RDF Schema does not provide a way of describing the behaviour of an object, that is to say the operations that it can perform.

5.1 Specifying Behaviour

The absence in the SW of a standard way of representing behaviour is a major problem. Another technology, known as Web Services, has recently stepped in to fill the gap. The downside of this otherwise very positive development is that, as Web Services and the Semantic Web have been defined by different organisations, they are not well integrated.

Luckily the standardization process of the Web Services Description Language (WSDL) [10] (the part of the Web Services proposal that deals with the specification of services) has now moved to the W3C. As part of this activity the W3C should soon produce a standard model to describe object behaviour in RDF Schema.

For NESSTAR we could not wait for this standardization process to complete. The NESSTAR objects can perform a wide range of operations: queries, statistical operations, file transfers, etc. and we needed a way of specifying them formally. In order to do so we have defined the NEOOM Object Model, a small 'RDF ontology' to describe methods (see [7] for details). With this extension RDF Schema becomes a fully-fledged Interface Definition Language (IDL).

In the NEOOM Object Model (see Fig.1), methods (e.g. *Login*) are defined as subclasses of the *Method* class. Method invocations are represented by instances of the method classes. Defining a method as a class is a bit unusual [3] but it has the advantage of making a method invocation an object

[3] In Java for example, a method is represented by an *instance* of

in its own right. Being an object a method invocation can be represented in RDF (and therefore stored, transmitted or logged easily) and can have methods and properties.

Method parameters are defined by instances of the *Parameter* class. A *Parameter* is conceptually very similar to an *rdf:property* and it inherits from it. As methods are classes then a parameter is just one of its properties.

6 Managing Model Changes

No matter how good a SWA's domain model it will never be complete and final. Information publishers will always need to extend it to represent more specific information or to add new functionality. These extensions will often be applied by different organizations and in an uncoordinated way.

The long-term viability of a SWA depends on its ability of accommodating these extensions gracefully so that they do not lead to a "balkanisation" of the system.

RDF and RDF Schema have been designed with distributed extensibility in mind and provide a variety of mechanisms at this effect: class and property inheritance, open-ended set of class properties, reification.

Using inheritance is possible to extend and specialize an existing concept without breaking all the applications that depends on it. We have had a good demonstration of the power of this mechanism recently when after having massively extended the NESSTAR object model our old clients have (mostly) kept on working by treating the new more specialised objects of the new model as the more general concepts they were originally developed to process.

Incidentally this is one major advantage of RDF Schema with respect to WSDL [10] as an IDL. WSDL not being object-oriented, basically is just an RPC specification language, does not support the same kind of interface extensibility. This will make it difficult to upgrade and evolve Web Services applications (an issue that might soon disappear if WSDL will be redefined as an RDF model).

The fact that the modelling language is extensible unfortunately does not automatically guarantee that any software application built on top of it will be able to gracefully handle future extensions. Applications developers can easily commit the mistake of relying on aspects of the model that will change and be invalidated at a later time.

---

nesstar.lang.reflect.Method, not by a separate class.

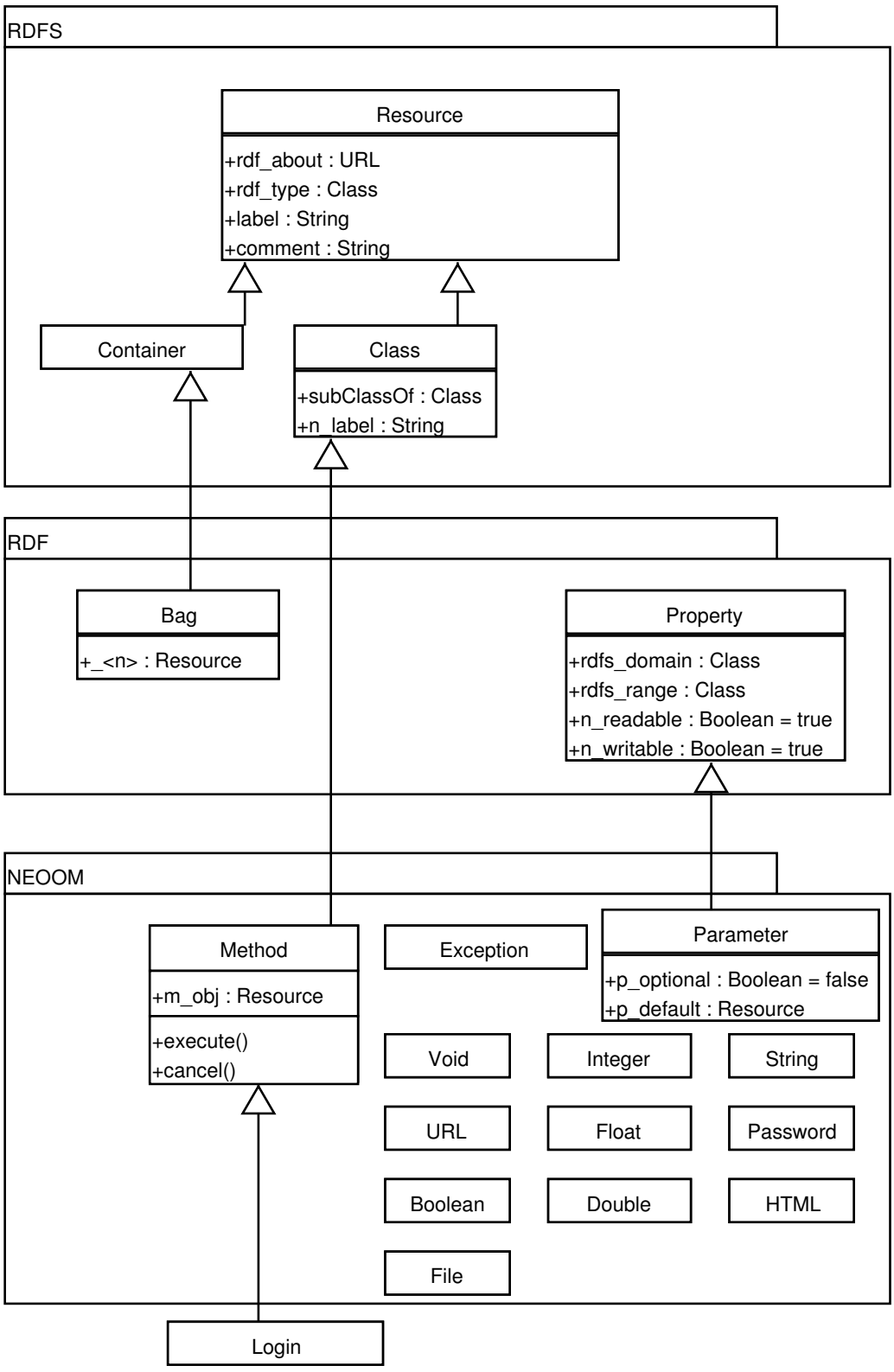

Fig. 1: NEOOM Object Model

7 Choosing a Representation Format

Once the model has been defined it is possible to describe and publish the actual instances of the objects (e.g. a Dataset, a Variable) on the SW.

To do so we need to choose a suitable representation format, a syntax that we will use to code our objects in a machine-parsable form.

The SW provides a standard solution to this problem. The objects are described in RDF [13] and coded in the standard RDF's XML representation.

A possible alternative would be to define a specialised, *ad hoc* XML syntax. A possible advantage of this solution is that, if properly done, it could lead to a more compact or simpler syntax. The disadvantages are far more significant though: a good syntax is difficult to design properly, a specialised syntax requires a specialised parser and being non-standard will make it harder the integration with other SWAs. For these reasons in NESSTAR we have decided to adopt the standard RDF's XML syntax.

8 Publishing Information on the Semantic Web

The XML-coded objects can be distributed in a number of different ways: stored in a file, sent by email, published on a Web site, etc. Naturally, for a SWA is particularly important to specify how the objects are going to be made available on the Web.

There is a very simple solution to this problem. Being so simple it deserves a pompous name, we might call it the Self-Description principle:

> *Semantic Web Objects and Classes self describe themselves by making accessible, via HTTP or other protocols, their RDF description at their URL.*

This is nothing new. The principle simple states that SW objects are accessed exactly as any other WWW resource. This is also the solution that we have adopted in NESSTAR. Each NESSTAR object "lives" at some HTTP URL. When a user client accesses the object URL the object returns a description of its current state in RDF. The same applies to the class definitions. We assign an HTTP URL to each class and we make the RDF Schema definition of the class available at that URL.

8.1 Performing Remote Method Calls

Finally we need to specify how to perform remote object-oriented calls so that remote clients can access objects behaviour.

SOAP [9] is an increasingly popular solution to this problem. In NESSTAR, even if we plan to support SOAP in the future, we currently rely on an older and much simpler alternative: the protocol used to submit HTML Forms (as specified in [15, sect17]). It is very easy to define a set of conventions to map method calls on top of this simple protocol (again see [7] for details). This solution has a couple of significant advantages with respect to sending a SOAP XML message: method calls can be performed using a normal web browser (and in general using programs written in any language that comes with an HTTP library) and it is easy to determine an URL that corresponds to the operation. The URL can be used by an application to represent and replay the operation.

9 NESSTAR System Components

The basic architecture of the NESSTAR system is identical to the WWW architecture. Resources are hosted on NESSTAR Servers. The Servers serve both normal WWW resources such as HTML pages, images, etc. and statistical objects. Just as the WWW, NESSTAR is at the same time fully distributed, each server is totally independent and there is no single point of failure, and integrated, as users experience it as an interconnected whole.

Users can access the system using the NESSTAR Explorer (a Java application for power users), the NESSTAR Light Explorer (a WWW interface based on Servlet technology), the NESSTAR Publisher (a metadata editing, validating and publishing tool) or the Object Browser (a web based generic client used for object testing and administration).[4]

The NESSTAR Explorer is particularly interesting for its similarity to a common Web Browser. Users can enter the URL of any WWW or NESSTAR resource in a Location bar. Normal WWW resources will be displayed just as they would in a normal Web Browser (or an external Web Browser is invoked to handle them), NESSTAR statistical objects are handled specially and displayed and manipulated in an efficient and flexible user interface.

The current version of NESSTAR is mainly implemented in Java (some modules are in C++). The deployment diagram 2 shows the main components of the system. On the server side we have three main components: an Enterprise Java Bean Container [1] (JBoss 2.4.x in the current implementation), a Web Server/Container (currently Tomcat Catalina) and a relational database. The NESSTAR objects are hosted in the EJB Container as Beans. They have only local interfaces so they are not directly accessible from the

[4] The clients can be downloaded from the NESSTAR Web Site [3]. Sample Web clients (the Object Browser and the NESSTAR Light) can be accessed at http://nesstar.data-archive.ac.uk/

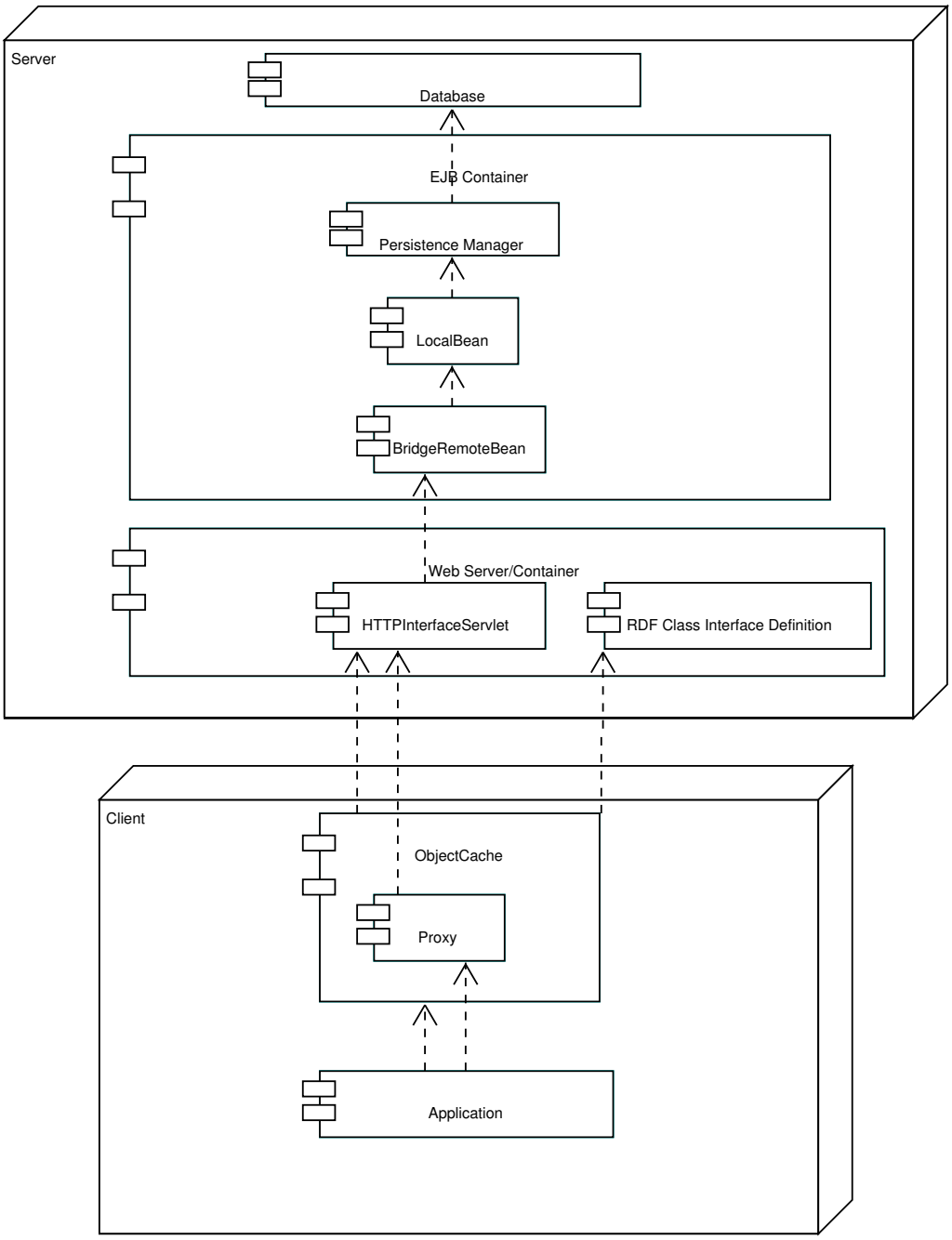

Fig. 2: System Components

outside of the container. Access takes place through a Bridge Bean. The interface with the Web is implemented by a Servlet that converts HTTP requests in RMI calls to the Bridge Bean. The RDF class interface definitions are stored in the Web Container. They are downloaded on demand by clients, such as the Object Browser, that need to discover at run-time the interface of an object.

Java client side applications access the remote NESSTAR objects through an object oriented API that supports:

- execution of remote object methods
- access of remote object properties
- traversing of object relationships
- operation bookmarks
- caching of remote objects
- handling of (multiple) authentication challenges
- HTTP and HTTPS wire protocols

For every class of NESSTAR objects the API includes a corresponding 'proxy' class. All accesses to remote objects take place through the proxy classes. The proxies are hold in an object cache. When a client application asks for an object with a given URL the API checks if the object is already in the cache. If this is not the case it performs an HTTP GET operation to the object URL. If an RDF description is found at the URL the corresponding proxy instance is automatically created, cached and returned to the calling application.

Once an application has got the proxy corresponding to the desired object it can access its properties via normal accessor methods (get/set) and perform operations on it. For each operation the API provides a corresponding URL. The application can store the operation URL so that the operation can be replayed at a later time.

10 Design and Development in NESSTAR

The design and development of NESSTAR objects is rather straightforward (see the use case diagram Fig.3). A Designer uses an UML modelling tool (currently we use Argo/UML or Poseidon) to create a Class Diagram specifying the classes to be created. The class diagram is saved as an XMI [5] file, the standard format for storage of UML diagrams. The XMI file is processed, using XSLT scripts, to generate for each class:

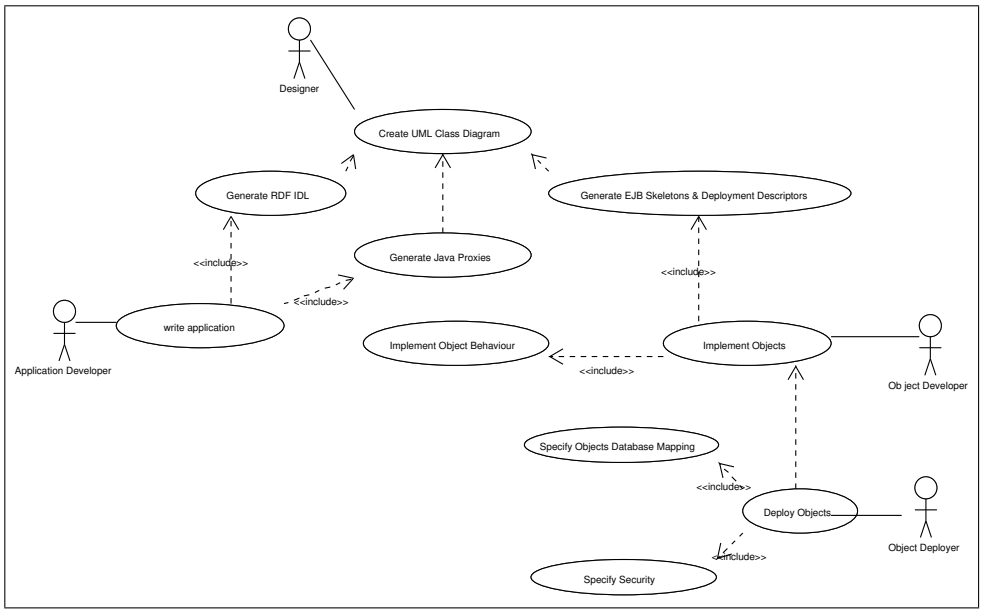

Fig. 3: Design and Development Use Cases

- the interface definition in RDF
- the Java proxy for the API
- the skeletons of the Enterprise Java Bean plus the corresponding EJB deployment descriptor
- the skeleton of a JUnit compliant testing class

The Object Developer can now proceed to implement the object by adding the methods implementations to the object skeletons. A Deployer completes the process and deploys the new classes in the EJB server by specifying their database mapping and security information.

The Application Developer adds the proxy classes for the objects he is interested in processing to its classpath and use them to write his application. Applications can ask the API to read in the RDF Interface Definitions to discover dynamically the properties and operations of any NESSTAR object.

11 Conclusions

The advent of the Semantic Web has made possible the creation of a new class of distributed applications that combine the simplicity and scalability of the

Web with the formal interfaces and modelling of traditional object-oriented middleware.

SW technology offers many benefits:

- Support for sophisticated domain models
- Distributed extensibility
- Integration with the current Web
- Programming language independence

NESSTAR uses the strengths of both the current WWW infrastructure and the new Semantic Web technologies to provide at least some of the basic element of the Data Web "dream".

Compared with other Semantic Web applications NESSTAR is rather simple. It doesn't define sophisticated ontologies or make use of advanced SW features such as reification or logical inference. The fact that even simple SWAs as NESSTAR can deliver real value to information publishers and users is certainly a good omen for the future of the Semantic Web vision.

References

[1] Enterprise JavaBeans. http://java.sun.com/products/ejb/.

[2] EU Project FASTER. http://www.faster-data.org.

[3] EU Project NESSTAR. http://www.nesstar.org.

[4] Web Ontology Language (WOL) Working Group. http://www.w3.org/2001/sw/WebOnt.

[5] XML Metadata Interchange (XMI). http://www.oasis-open.org/cover/xmi.html.

[6] Resource Description Framework (RDF) Model and Syntax Specification. http://web4.w3.org/TR/REC-rdf-syntax/, February 1999.

[7] Pasqualino Assini. NEOOM: A Web and Object Oriented Middleware System. http://www.nesstar.org/sdk/neoom.pdf, 2001.

[8] Pasqualino Assini. Objectifying the Web the 'light' way: an RDF-based framework for the description of Web objects. In *WWW10 Poster Proceedings*, Hong Kong, May 2001. World Wide Web Consortium.

[9] Erik Christensen et al. *Simple Object Access Protocol (SOAP) 1.1*. World Wide Web Consortium, 2000.

[10] Erikothers Christensen, editor. *Web Services Description Language (WSDL) 1.1*. World Wide Web Consortium, March 2001.

[11] R.V. Guha and Dan Brickley, editors. *Resource Description Framework (RDF) Schema Specification 1.0*. World Wide Web Consortium, March 2000.

[12] James Hendler, Tim Berners-Lee, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[13] Ora Lassila and Ralph R. Swick, editors. *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium, 1999.

[14] Simon Musgrave and Jostein Ryssevik. The Social Science Dream Machine: Resource Discovery, Analysis, and Delivery on the Web. *Social Science Computer Review*, 19(2 Summer), 2001.

[15] Dave Raggett et al., editors. *HTML 4.01 Specification*. World Wide Web Consortium, December 1999.