# APPLICATION OF A METASYSTEM IN UNIVERSITY INFORMATION SYSTEM DEVELOPMENT

**Petr Smolík, Tomáš Hruška**

Department of Computer Science and Engineering, Faculty of Computer Science and Engineering, Brno University of Technology, Božetěchova 1, CZ-61266 Brno, {smolik@dcse.fee.vutbr.cz, hruska@dcse.fee.vutbr.cz}

## Abstract

As all other higher education institutions, Brno University of Technology has been working on its university information system in order to provide its students and faculty with better tools for communication and reporting. Facing the distributed nature of its quite independent schools and departments, the university had to employ new technologies to enable co-operation of various information systems, already in place or being developed, with the central databases held at the top level. The solution seems to be usage of a metasystem to generate a system based on Web services complemented with an object-oriented data access. This paper gives a short overview of the technologies used and discusses the advantages they brought.

## 1. Background

University information systems are specific in the scale of various data produced and processed by different groups of users. E-commerce systems used by companies to communicate their business data with their partners usually exchange just a few types of business documents, such as orders, stock quotes, or invoices. Even though these documents change a little over time, there is no dramatic need to keep adding new types of them to cover other business needs. Nevertheless, in the university environment with many diverse and changing activities, there is high demand for information system flexibility. New services for new groups of people need to be designed and redesigned continuously. Therefore there is need for effective means to build user-to-system or system-to-system interfaces. An on-going growth of the information system has to be taken care of. The engineers at the Brno University of Technology have taken advantage of technologies provided by a couple of leading local IT companies and designed a core part of the Web-based information system that is now ready to grow.

In order to manage a growing system one needs to have a metasystem that defines and manipulates with the underlying system. Information systems usually contain one or several databases where data are stored. For each piece of data in any of the databases, the metasystem needs to have a description (metadata). We will introduce an object-oriented mechanism used to define the metadata and we will also turn our attention to description of the services provided by the system.

The information system that is being developed at the Brno University of Technology uses the concept and concept view definitions presented in this paper. Concepts are used to describe data in object-oriented fashion and concept views are used to describe more complex views on interrelated data. Since the definitions are always accompanied with human readable navigation and support texts, it is possible to use them as metadata for generation of Web enabled services for human users or for system-to-system communication. So far it is our experience that using a metasystem to define, redefine, refine, and generate underlying system speeds implementation time and improves quality of the resulting expanding system. This is because it enables the project team to focus on the services that need to be offered to users instead of continuously having to program the services at low level with many chances of mistake.

## 2. The Problem

Before our project began, there were three parts of a university information system in operation: economical agenda, personal agenda, and student agenda. All these systems are usable only by the administrative staff and none has Web accessible interface. The student agenda has never been completed enough to produce any results. Students register to classes based on filled-up application forms submitted to the registration office. Further, there are many requirements of the staff to report their research activities, publications, or projects. The reports are gathered at departments in various types of electronic documents and sent to the central computing center. The computing center than was able to provide overall report for the Ministry of Education.

The university therefore recognized the need to advance their university information system and enable both students and faculty to work with their data on-line. The matter has been complicated by the fact that the university is composed of several quite independent faculties (schools). Each faculty has its specific needs of their information system and the question was whether the central part of the system should satisfy all the requirements of all the faculties. Decision was made to centrally offer central services that are equivalent for all the faculties, and let the faculties invest into the specifics they require over the scope of the central system.

In order to start development of such a large-scale and distributed system, a pilot project was needed. Because there was no science and research reporting system in place, building a research reporting system in a way that would satisfy all requirements became the pilot project. Requirements were the following:

- Web access to central storage of science and research activities
- user authentication and authorization for access to specific resources
- interfaces for existing systems to submit and request data on-line
- extensibility on the side of different faculties

The central computing center took responsibility for implementing the pilot project. Up to date there was a good experience with Progress database and its Web-enabled facilities. All the earlier school agendas were built as applications using Progress as its application development environment. Nevertheless, future transfer to another environment was expected. For that reason, there was need to build the pilot as open as possible to accommodate to whatever databases that might be later available.

## 3. The Solution

The computing center decided to base the solution on open technologies. The main one of them is XML, Extensible Markup Language used for platform-neutral representation of complex data [2]. Since there was just a little experience with XML, the computing center requested expertise from Brain Systems, since Brain Systems had already two years of using XML in commerce applications. Brain Systems became technology provider for the pilot project and provided its development tools, their database independent XML application server (eBrain), and an early version of a metasystem able to generate eBrain applications (MetaBrain). The technologies are based on open standards such as Java, XML, XSL, and HTTP. Since the system developed is always defined in metadata on the metasystem level, it is possible to generate application not only for eBrain used for the pilot, but for other available application servers. That ensures independence of the resulting solution on a specific platform.

The whole business model is built on modern trend of information system development where there are two fundamental partners. One is an experienced technology provider but knows a little about the problem domain of the resulting application. And the other has very good know-how in the problem domain, but doesn't have much experience with the technologies. Either of the partners may for specific projects outsource the knowledge and work of the other. In the case of our pilot project, the central computing center contracted Brain Systems to outsource the technologies, but itself it is taking care of defining the university information system know-how. In the following subsections we will provide closer look on the technologies used to satisfy all the project requirements.

### 3.1 Conceptual Model

In this section we will present the fundamental mechanism we have selected for object-oriented description of data in a system. Our metasystem uses descriptions, called concepts, to provide static views of systems it controls. The concept definitions were first employed in the G2 object-oriented database system designed at VEMA Computers and Projections. The G2 model of concepts provides the most complete picture of what could be implemented at the metasystem level to describe data processed in systems. For the purposes of this paper it will be sufficient to introduce the concept of "concept definition", but an interested reader may find full G2 model description in [1].

CDL (**C**oncept **D**efinition **L**anguage) is the specification language for the conceptual object modelling of G2 system applications. It exists in two syntax forms - the textual form and UML. The object model consists of *objects*. An object is an ordered tuple of property values. Each property has *name* and *data type*. We call *concept* the Cartesian product of the named property data types. For the time being, concept is equivalent to the well-known notion of *class*, but we have expanded it in its semantics. Each property can be parameterised, i.e. each property can have an arbitrary number of parameters. Each parameter must also have some *name* and *data type*.

We start our examples with a Person object. Each person can have a *list of names*, a *family name*, and a *birth date*. Concepts are used to define objects. The following concept defines the names Name, Family, and DateOfBirth as properties with their corresponding data types **String** and **Date**. The Name property is parameterised. The Order parameter is used to define the order of person names.

```
Concept Person
Properties
     Name(Order: Integer):   String
     Family:             String
     DateOfBirth:        Date
End Concept
```

We do not state whether the property is a date or an algorithm. The concept defines the interface of the object for a user communication. The implementation is defined elsewhere.
The basic set of elementary data types includes **`Byte`**, **`Integer`**, **`Long`**, **`Single`**, **`Double`**, **`Currency`**, **`Date`**, **`String`**, and **`Boolean`**.

## 3.2 Object-Oriented Data Access

In the previous section we have shown how data in a system may be described on the metasystem level using the concept descriptions. In this section we will explore way how the data may be expressed as instances of concepts (objects) and how collections of interrelated objects may be expressed. Further, we will present the XML Data Access (XDA) query language that was developed at Brain Systems and has been used on several e-commerce and e-business projects [2]. The system realized at the Brno University of Technology completely relies on the XDA technology to enable flexible data access to various types of databases. Moreover, XDA is complemented with Web Services Access Control (WSAC) that allows limiting access to objects based on user and group rights to services and objects within those services. WSAC was also developed at Brain Systems, but its description is out of the scope of this paper.

### *3.1.1 Expressing Objects in XML*

There are two ways how data in an object-oriented database might be accessed. Either it could be accessed directly by application objects via an object interface, or it could be accessed externally through an XML interface.

Each object or a collection of objects of a certain concept may be expressed in XML exactly as defined in the concept definition. The property values of the objects are enclosed in elements tagged with the corresponding property names. All the property value elements are then enclosed in an element tagged with the name of the corresponding concept. For example, a sample *object* of the Person concept may look like the following:

```
<Person oid="25456787">
  <Name>Drundun</Name><Family>Hallway</Family>
  <DateOfBirth>1955-05-16</DateOfBirth>
</Person>
```

Similarly we may wrap individual objects into *collections of objects* using the concept collection name also defined in the concept definition. Then the individual person objects are enclosed in the corresponding collection element:

```
<People>
  <Person oid="25456787">
    <Name>Drundun</Name><Family>Hallway</Family>
    <DateOfBirth>1955-05-16</DateOfBirth>
  </Person>
  <Person oid="25456788">
    <Name>Miranda</Name><Family>Hallway</Family>
    <DateOfBirth>1959-11-07</DateOfBirth>
  </Person>
</People>
```

### 3.1.2 Concept Views

What we have shown so far could be also expressed as simple rows of a table as we know them from result-sets in relational databases. Nevertheless, in the world of objects we need much more than this. It should be possible to complex data structures corresponding to *views of related objects*. Concept definition allows properties to define relationships with other concepts. According to the Person concept each person may have a father and several children. That we may express in the following way:

```
<Person oid="25456787">
  <Father oid="12456477">
    <Name>Eduard</Name><Family>Hallway</Family>
    <DateOfBirth>1926-01-21</DateOfBirth>
  </Father>
  <Children>
    <Child oid="84544587">
      <Name>Monica</Name><Family>Hallway</Family>
      <DateOfBirth>1978-10-29</DateOfBirth>
    </Child>
  </Children>
</Person>
```

Since objects are often interrelated in complex way, we need mechanism that would allow us to access only the data corresponding to few relationships that we are interested in. The view of related objects that we call a concept view is defined in a *concept view definition*. Definition for the above view might be the following:

```
<ConceptView name="PeopleWithFathersAndChildren">
  <Include concept="Person">
     <Include relationship="Father"/>
     <Include relationship="Children"/>
  </Include>
</ConceptView>
```

### 3.1.3 XML Data Access (XDA)

In order to communicate with an object-oriented database a query language is needed. There could be three types of queries. Queries that get data, update data, or delete data. Each query specifies which concept or concept view defines the data of interest. The get data queries further include constraints limiting the scope of the selected data. The update data queries include the data to being updated or inserted, and the delete data queries include the necessary object ids that identify the objects to be deleted. For example to obtain collection of all Person objects available we may use the following query:

```
<GetData concept="Person"/>
```

To obtain collection of all the people, whose family name is "Hallway", the following query might be used:

```
<GetData concept="Person">
  <Where><Eq property="Family" value="Hallway"/></Where>
</GetData>
```

Concept view queries are built similarly. To obtain all people with their fathers and children the following might be an appropriate query:

```
<GetData conceptview="PeopleWithFathersAndChildren"/>
```

The concept view queries might also be constrain on any level of the included subtree as in the following example, which would provide all people whose child's name is Monica.

```
<GetData conceptview="PeopleWithFathersAndChildren">
  <Where>
    <Eq property="/Person/Child/Name" value="Monica"/>
  </Where>
</GetData>
```

Finally, here are examples of both an update data and delete data queries:

```
<UpdateData concept="Person" insert="yes">
  <Data>
    <Person oid="25456787">
      <DateOfBirth>1955-05-17</DateOfBirth>
    </Person>
  </Data>
</UpdateData>
<DeleteData concept="Person">
  <Data><Person oid="25456787"/><Data>
</DeleteData>
```

We have shown here only simple examples of concept and concept view queries in order to demonstrate the object access mechanism usable to access data in object-oriented database. We have already used exactly the same mechanism also to provide XML based access to data in various relational databases and the query language specification covers many features needed for real-life application. The complete presentation of the query language is out of the scope of this paper.

## 3.3 Web Services

We understand Web services as either human or machine interfaces to information systems. Web services are accessible over the Internet where human users use browsers to view HTML pages or other systems communicate XML documents. XDA mentioned in the previous section serves as a Web services provider for other systems to access data in our system and it also enables the business logic tier to access data in an object-oriented fashion.

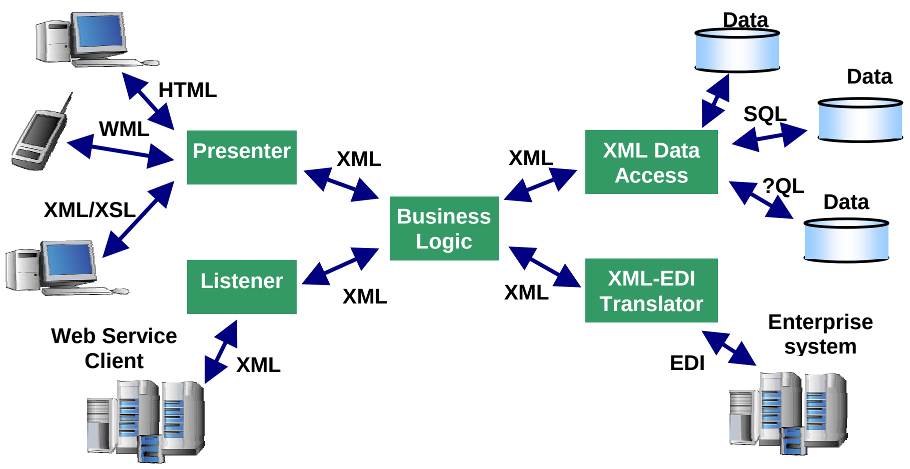

Figure 1: Web Services Provider Architecture

Figure 1 shows general Web services provider architecture. On the right side, each Web service accesses data in various data sources, such as databases or directly accessible enterprise information systems. On the left side, each Web service communicates with the outside world. It could present its services directly to users via the Presenter, which transforms the pure data into appropriate Web pages, and to other systems it provides its services via the Listener, which listens for service requests and provides service replies. All communication with the outside world is based on the HTTP protocol and the data is represented either in some visualization format for users (i.e. HTML) or in XML for other systems. Any Web service may use other Web services to obtain necessary information or to have computations done. Web services are mostly seen as services provided by a machine to another machine, nevertheless, we see them also as services directly offered to human users. From our experience the logic of the service is equivalent in both cases, only the visualization part is being added for the comfort of the user. And further, if there is a good service description then the graphical user interface could be easily generated. Therefore we will consider each Web service as both machine and human accessible.

Web services enable communication of data among companies or independent parts of an organization (i.e. business entities). Each business entity may offer their Web services to other entities. Because the use of Web services has to be easy and connecting to new services needs to be simple, there will be standard ways to describe, offer, and find services. The most promising language for description of Web services is the XML-based Web Services Description Language (WSDL) jointly developed by Microsoft and IBM. Further, the Universal Description, Discovery and Integration (UDDI) specification describes Web services and programmatic interface that define a simple framework for describing any kind of service.

As we mentioned earlier, there are two generic types of a Web service. Ones that provide access to data and ones that do computations. Example of the first case could be sending and an order or receiving an invoice. Example of the second case could be requesting price for sending a package of specified size and weight from place A to place B. Web services that access data could be defined in terms of concept views that we present in this paper. Web service is therefore defined as a concept view enriched with documentation texts and other information related to level of access to objects within the service. We call this enriched description as service description. Each service thus defines what view of interrelated objects is available within that service. For each concept in the service definition defines what operation (list, create, edit, delete) is allowed. The service may define implicit filters on accessible objects and other information needed for proper documentation or visualization. Further constraints on what objects could be accessed by what users within given service are put by a separate mechanism (Web Services Access Control) mentioned earlier.

## 3.4 Metasystem

In the previous sections we have discussed ways to represent data about a system (metadata). From the metasystem point of view there are serveral types of metadata. Metadata that define the data in the systems (concepts), metadata that define views on data with their hierarchical relationships (concept views), and metadata that define Web services (service descriptions). In a way, service description in its complete form may take form of a personal process. We define personal process as the activity that the user performs in order to take a full advantage of a service. For example, if we may look at an example service “Send purchase order“. From the data perspective we know that we need to send an XML document containing proper order head and order items. For any user, preparing such a document is a process of entering order head information and adding items. This personal process information is also part of our service description, so that suitable editor may be generated for each service.

In the near future we will also attempt to add matadata that will define organizational processes. We see an organizational process as the process of using different Web services by different users in order to reach an organizational goal. Usually, organizational processes are executed by individual users implicitly and they have no explicit definition, or they are hard-wired in the system implementation. By making organizational processes explicit by representing them on the metasystem level, we expect to be able to better control continuous growth of the underlying system.

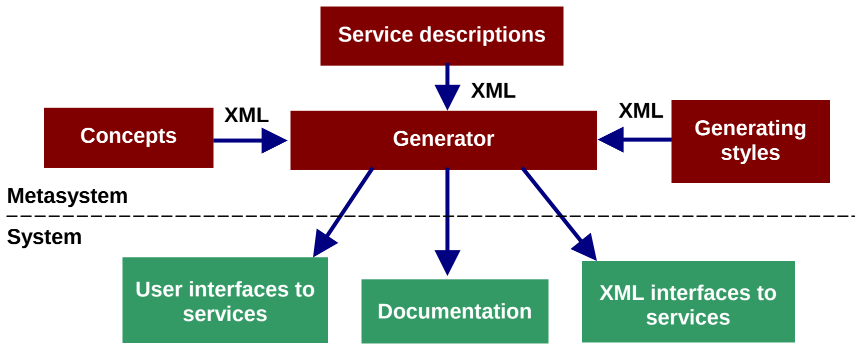

Figure 2: System Generated by the Metasystem

The current implementation of the metasystem as shown on Figure 2 uses concept definition to define objects manipulated by the system, uses service descriptions (with concept views included) to define Web services with their corresponding personal processes. Using the Generator, metasystem generates several parts of a system. The XML interfaces to services are the full-blown Web service listeners presented in the previous section. The User interfaces to services are the user Presenters for each Web service. All the system parts are generated from a universal description to a specific system program code using the Generating styles. There are special styles for generation into different target environments. The current version of the metasystem generates eBrain Application Server applications, but it is possible to add generating styles to generate applications in ASP, PHP, WebSphere, WebLogic, or JBoss.

Based on both concept definitions and service definitions the metasystem is able to generate not only proper interfaces, but also documentation. In development and maintenance of a system offering wide range of services and encompassing many different types of data there is high need for good documentation. For this reason, the concept definitions are further complemented with human readable descriptions of all properties and concepts themselves, so the complete information as what the data means is presentable.

## 4. Current Challenges

The ongoing works take several directions. Fundamentally, there are data and service being described. The corresponding concept definitions and service definitions are being entered into the metasystem. The metasystem is fully integrated in to the system it generated, so that there is no break between the works on defining the system and the works done with the system itself.

It should be noted that the user interfaces are completely generated based on the meta-data. Therefore, if the computing center staff adds couple of concepts and a service definition, the dynamic browser pages will be generated also for the new service. The design of the Web pages will thus be generated based on the styles defined. With central change of the styles it is possible to change the look and feel of the user interface for all of the services at one time. On the other hand it is possible to have different styles for different sets of services. Each service has a name and a root concept. In the detail it is possible to define descriptions and of the personal processes related to each service and to enter human readable documentation and navigation texts.

It is planned that the system will be continuously amended with new services for different types of users. Each user sees her own dynamic menu that contains only services she has access to. It is thus possible to add services on the run and just authorize proper groups or individuals to use them. Finally, for each service defined there is the XML interface that other systems may use to get, create, or update selected data.

## References

1. Hruška, T., Máčel, M. G2 - Component Architecture of Object-Oriented Database System. ISM 99 Workshop Proceedings, 1999, 119-124
2. Smolík, P., The Importance of Extensible Markup Language. ISM 99 Workshop Proceedings, 1999
3. Smolík, P., Tesáček, J. Data Source Independent XML Data Access. ISM Conference 2000 Proceedings, 2000, 17-22