ACTIVe3D-WS: a Web-services Based Multimedia Platform

Jean-Claude Simon, Christophe Cruz, Christophe Nicolle
Laboratoire Le2i
Aile des Sciences de l'Ingénieur
Université de Bourgogne
BP 47870 – 21078 Dijon Cedex – France

Abstract

In this paper, we present ACTIVe3D-Build, an electronic platform for the management of civil engineering projects. This platform is the result of a research that aims to represent semantics of computer graphics modeling within a relational database. The final goal is to provide Web-services on this platform to allow the development of distributed applications using our technology for storing and extracting 3D-scenes from our relational databases.

1. Introduction.

The 3D graphics computations require both complex software and high-performance hardware. Moreover, a large number of applications need to process more than one file at the same time. Sending these files over the Internet and displaying them at a client's computer is not possible without the use of dedicated language such as VRML (Virtual Reality Modeling Language) [16]. VRML allows to create 3D-scenes, and to navigate in these scenes using an Internet browser with an appropriate plug-in. Nevertheless, the manipulation of theses files is not easy due to the complexity of the VRML structure. In order to solve this problem, a new language has been developed by the W3C [20] for building 3D-Scenes: X3D (eXtensible 3D). X3D is based on XML where 3D-Scenes are represented as XML documents. The tree structure of XML allows us to manipulate and decompose the 3D-Scenes into basic elements corresponding to 3D primitives. Nevertheless, this representation does not allow computer graphic algorithms to manipulate many scenes at the same time.

In this article, we present an electronic platform, the ACTIVe3D-Build (Figure 1) [1], for the management of civil engineering projects. This web-based platform allows all participants of a project (electrician, plumber, etc.) to directly use and exchange documents in the projects via a simple Web browser. Moreover, a 3D-visualization of a building that is generated from databases allows each participant to move around in the building being designed. In this 3D-environment, players will be able to query all objects that compose the building and thus obtain information about them from the IFC database. The IFC contains business information describing the civil engineering project.

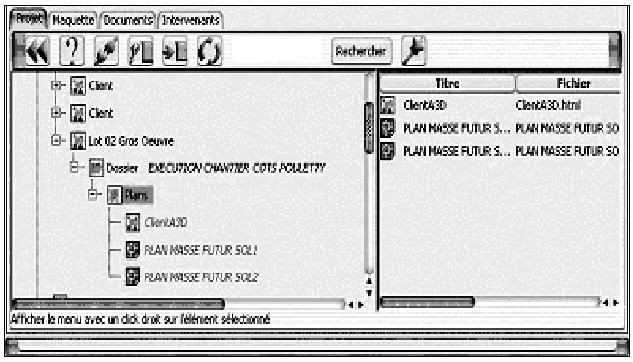

Figure 1: electronic platform

This platform is the result of a research that aims to represent semantics of computer graphics modeling within a relational database [3]. The semantic modeling is involved at two levels. The first level corresponds to creation of two databases, the first one stocking the IFC, and the other one stocking compositions of 3D scenes. The second level represents the association of the semantics of the IFC with 3D-elements of the 3D database. In order to permit this association, we represent all information (IFC, 3D scenes and management information) as XML trees.

The decision-makers of the company wish to sell some business skills developed on this platform. The final goal is to provide Web-services on this platform to allow the development of distributed applications using our technology for storing and extracting 3D-scenes from our relational databases (Figure 2). In this paper, we present

the architecture and processes to build Web-services from an information system. We applied this method to our platform: transforming 3D databases extraction and insertion business skills in Web-services.

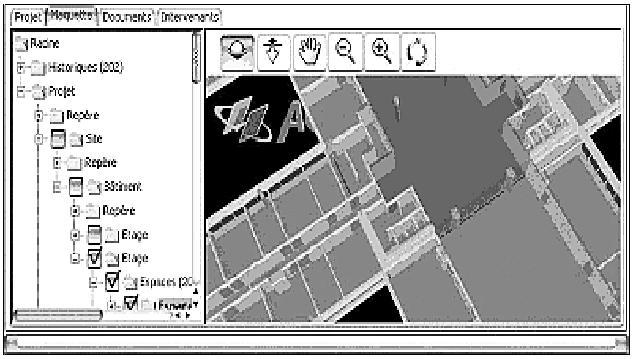

Figure 2: 3D scenes viewer

2. Web-services overview.

The Web-service is usually considered as a set of layers contained in a stack. These layers are dynamically defined following the user needs and are called through a set of Internet protocols. These protocols are different depending from the various architectures proposed in the literacy [4, 5, 6, 8, 9, 10, 19]. However, in all Web-services architecture, a pivot set of protocols is always used. This pivot set is composed of: SOAP [13], WSDL [18] and UDDI [15]. These protocols allow the discovery, description and the communication between Web-services.

SOAP (Simple Object Access Protocol) is a mechanism using XML for the exchange of structured and typified information between several actors in a decentralized and distributed environment. WSDL (Web Services Language Description) uses XML syntax to describe the methods and parameters of the Web-services. UDDI (Universal Description, Discovery and Integration) makes it possible to find and publish Web-services over the Internet. It is a global dictionary of Web-services.

All proposed solutions to defined Web-services are based on XML [21]. XML (eXtended Markup Language) is an open textual language, providing data with structural information and relative semantics [11]. Moreover, beyond its ability to separate the content and the presentation, XML has become the standard of data exchanges on networks.

Usually Web-services can be developed using technologies like Java, CORBA [2] and .NET [25]. Java XML Pack is Sun's [7] effort to encapsulate the various standards in the Java space. This pack, containing a collection of Java APIs for XML, is designed to support the standard APIs for Web services including SOAP, XMLP, WSDL, and UDDI. JAXP (Java API for Parsing) covers SAX (Simple API for XML) [23], DOM (Document Object Model) [22], and XSLT [24]. JAXB (Java API for XML Binding) is a mechanism for compiling XML data type definitions into Java classes capable of reading XML into Java objects and writing them back out again. JAXM (Java API for XML-based Messaging) is a SOAP-based protocol for sending messages. JAXR (Java API for XML Registries) is a specification that provides a unified interface for UDDI and ebXML registries, and conceivably other registries. JAX-RPC (Java API for XML-based Remote Process Communication) is a SOAP-based protocol for requesting operations on remote servers.

3. A building solution for the Web-services.

To build a set of Web-services, first it is necessary to define which services the company wants to sell. Usually the Web-service is the electronic representation of a specific business skill. The Web-service is considered as an abstract and behavioral view of the information system. In this section, we present a method to build automatically the programming base of Web-services. This method is articulate in three steps: Business skill definition, the low-level Web-service building, and the high-level Web-service building.

The **Business skill definition step** defines the set of business skills the company want to sell. This definition is made in three steps: The strategic step, the analysis step and the normalization step.

The strategic step concerns the political and management orientations. This level concerns the decision-makers. At this end of this level, a set of Web-services purposes is defined with corresponding development teams and pacification. "Use Case Diagrams" of UML are used to define the needs.

The analysis step concerns all static and dynamic aspects of the information systems, which are analysis and conceptually reorganized following the Web-services purposes.

The normalization step concerns the definition of norms, standards and planning for the development of Web-services.

The **low-level Web-service building step** is the intermediate layer in the Web-service building process. It corresponds in the translation of conceptual view of the information systems defined in the first step into a set of classes, attributes, links, and methods. Generally, the view of the schema, which corresponds to the business skill, can be directly transformed into Java Classes. The Classes contains attributes defines in the table and methods to create objects from an extraction of the databases or to store objects as table instances. Thus, at this end of this step, all views are transforming in Java Classes. These

classes form the programming base to develop the Web-services independently from the existing systems.

In our environment, we start from the X3D grammar, which is converted into a relational schema. From this schema, we generate java classes allowing the manipulation of 3D-Data.

```
<element name= "TextureTransform">
  <type content= "empty">
    <attribute default = "0 0 0" name = "center"/>
    <attribute default = "0 0 0" name = "rotation"/>
    <attribute default = "1 1 1" name = "scale"/>
    <attribute default = "0 0 0" name = "translation"/>
    <attribute name = "DEF" type ="ID"/>
    <attribute name = "USE" type ="IDREF"/>
  </type>
</element>
```

Script 1. Grammar of element "TextureTransform"

For example, the "TextureTransform" element represented in Script 1 is translated into a "TextureTransform" Table where all attribute element become attribute of the table. An identifier attribute is automatically generated during the process.

```
TextureTransform (TextureTransform _id, center,
rotation, scale, translation, id, idref)
```

At the low-level web-services building, this "TextureTransform" table is automatically converted into a java class "TextureTransform". The composition of this class is presented in Script 2. At this level, we have developed a tool, which generate automatically the creation script of the X3D database and classes. These classes are connected automatically to the database. This tool is the core of our low-level Web-service building. It is an API, which gives us methods to create, update and manipulate X3D databases without SQL queries.

In the **high-level Web-service building**, the development team builds the Web-services. For this, they build a "Class Diagram" of UML. This diagram contains classes, which are derived from ones defined in the previous step. Thus, the code for the database access and the attribute are reused. The resulting Class Diagram is directly transformed as Java classes organized in a Java applet. Moreover, the XML-Schema grammar of the Class Diagram is generated. The Web-services export their results as XML documents.

In our solution the extraction and insertion web-services presented below are composed of 543 java classes, which are composed of abstract classes and X3D element classes. We have also high-level classes, which permit high queries on the databases as operations of joint between tables, or operations of selection on tuples according to attributes values.

```
public class TextureTransform  extends X3DObject{
   //private attributes
   private String TextureTransform _id;
   private String center;
   private String rotation;
   private String scale;
   private String translation;
   private String id;
   private String idref;
    //constructor
   public TextureTransform (String
TextureTransform _id){
   //this constructor seeks information in the database
   … }
   public TextureTransform (String center, ….) {…}
   public TextureTransform () {…}
   //access methods
   public String getTextureTransform _id(){…}
   public String setTextureTransform _id(String){…}
   …
   public boolean storeDB(){
   //this method update database } }
```

Script 2. Class "TextureTransform"

In this section, we present (chart 1) the performance evaluating of the X3D database (insertion and extraction). The source files are VRML files. These tests are made without database or network optimization The following tests concern 1/ the time to convert a VRML file into a X3D document, to create the corresponding insertion script, and to execute this scrip in the database, 2/ the time to extract data and build a X3D document from the database.

File Size (in Kb)	315	1962	4039	8767
Creation of the Insertion Script	1'	4'	5'	40'
Insertion Script Running	6'	25'	30'	25'
Extraction of the X3D scene	15'	1''20'	2''20'	2''05'

Chart 1. Results of process

These results show that times of process depend on two criteria: The size of the file and the structure of the scene. For this last point, the scene 4 contains less tag than the scene 3. Nevertheless, its size is more important. The graphic elements are described by a set of points rather than a set of primitives (cube, sphere, etc.) Some elements of the scene 4 possess more than 15000 characters. We note that the extraction times are very important in

relation to the insertion times. It is owed to the Oracle optimizer that detects the best path in the database schema and then rewrites the query before executing it.

4. ACTIVe3D Web-services architecture.

The ACTIVe3D-Build architecture consists of three distinct layers based on XML (Figure 3): the "Data" layer stores information generated during the life of a civil engineering project. Moreover, this layer stores XML documents in the specific relational databases. The "Broadcast" layer provides the distributed communication support of information sharing through the Internet network. The "Behavioral" layer provides mechanisms of the information process. This layer is a set of Java Classes that manipulate several XML streams between databases.

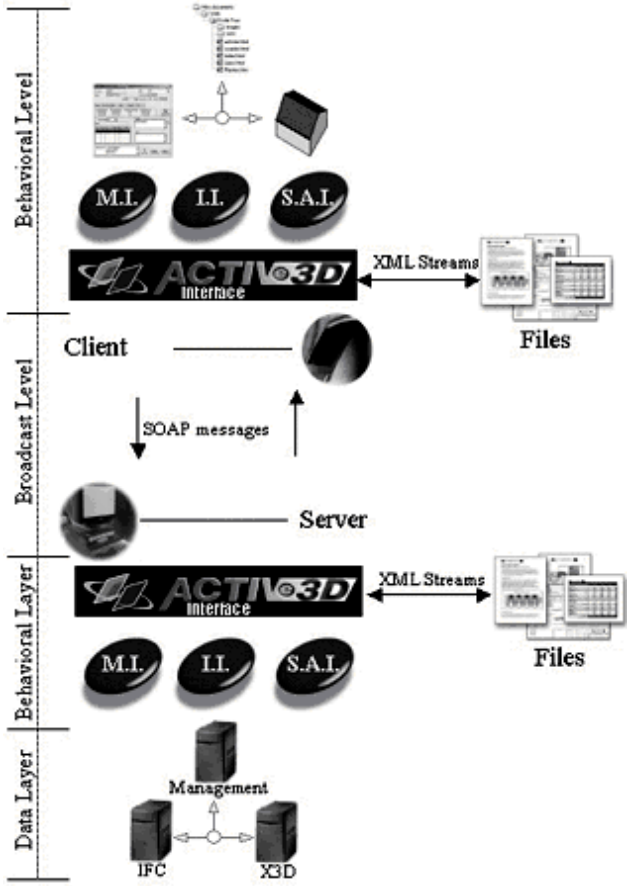

Figure 3: The ACTIVe3D-Build architecture

The **"Data" layer** consists of three relational databases (Figure 3: Management, IFC and X3D). The Management database contains all information about all the participants (professional information: roles, rights and actions in the project). Furthermore, it contains all information about documents that are exchanged during the life of the project (creation date, release date, information updates, deletion date, etc.) The IFC and X3D databases are used to store information usually kept in ASCII monolithic files. An analysis method developed by our team has permitted to convert these files into a relational format (primary keys, simple and mono-valued attributes, foreign keys). The set of databases represents more than 500 tables to store management, IFC and X3D data. The use of databases instead of files permits an increased flexibility of manipulation and avoids problems of the volume of file sizes when transferring files through the network. The IFC database is generated from the XMLifc language of the IAI. This language is a translation of the XML schema that contains production rules of the IFC files corresponding to the STEP specification [14]. The database X3D is generated from the XML schema of the X3D, which has been developed by the Web3D Consortium [17]. X3D is the XML translation of the VRML language.

The **"Behavioral" layer** is composed of three Java packages. The first Java package manipulates management information. A Java class is defined for each table of the relational schema in the Management database. Java objects are initialized from the database (extraction process) or by data inserted by the user (insertion process). This module is called Management Interface module (MI in the Figure 3). The second Java package permits to manipulate IFC. It is named the IFC Interface module (II in the Figure 3). The third Java package is intended for manipulation of X3D scenes. It is the Scene Access Interface module (called SAI in the Figure 3). We developed our own SAI module that is derived from the more recent Schema X3D [12].

The **"Broadcast" layer** is structured as a Client-Server. On the client side, a Java applet is used as a SOAP client. On the server side, a Java server is used. The architecture is developed entirely in Java. The communication between the client and the server is carried out using SOAP. SOAP uses the protocol HTTP to exchange the XML files and to bridge the gap of the firewalls. This Web-services oriented architecture makes the XML documents exchange easier than the previous developed architecture. This last one used a secured object stream between the server and the client whom involve problems with firewalls, proxy, etc. Both the client and the sever use a common ACTIVe3D interface that regroups the three interfaces (MI, II, SAI).

5. Application to our multimedia platform.

To illustrate this paper, we present two business skills in our company. First, we create object model from XML-Schemas written in Java and also relational data model for relational databases. Those methods allow generic generation of structures for memory or database

manipulation. Second, we create API, with the help of JDBC, for data's persistence. This API is a bridge between data memory model and database model. Thus, a constructor who seeks information in the database initializes Java objects. Moreover java objects are backed up in the database by calling storing methods. JDBC is an API that lets you access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases. This technology allows realization of complex data's manipulation as well in memory as into databases.

Now, we focus on 3D services, which are proposed as Web-Services. First one is 3D storing-Service and second is 3D Extraction Services.

5.1. 3D-storing service.

The storage process of an X3D document requires two steps. The first consists in carrying out a memory model of a document XML called X3D memory model (X3DMM). A parser SAX realizes this translation. We define for that an XMLHandler, which gives the translations rules of the X3D document elements into Java object. The Java classes, which allow definition of X3DMM, are generated from X3D grammar. The second step stores the X3DMM into database. This one resides in an easy way of programming using the recursively. The element root starts the algorithm while being stored in the data base then sends the command to each one of these children, so that they do as much of it and so on.

Once scenes are stored in the database, it is very simple to manipulate them with SQL queries (of with X3DMM). Values of all scene attributes can be modified. Insertion, modifications and deletions of 3D objects in several scenes can be performed simultaneously. It is possible to make all the attributes of scenes evolve dynamically such as to modify the coordinates of light source in a scene by simple SQL queries.

The 3D-storing service is a Web-service based on the business skill that store VRML or X3D files into a relational database. This section presents first, the mechanism of the business skill, and next the translation of this business skill as a Web-service.

The storage step being realized (Low-Level web-service building), it is advisable to set up the Web Service. This consists in the evolution of the class Diagram to build an EJB (Enterprise Java Bean) allowing the insertion from the web (high-level web service building).

5.2. 3D-Extraction service.

The extraction process is based on the same procedures as the insertion process.

First of all, it is necessary to know what it is possible to extract from the X3D database. For that a simple interrogation of the base makes it possible to know identify it each scene contained. It is also possible to thus select an element according to his name, and of carried out a catalogue of object 3d. Once the given element root, as is identifier, it is possible to initialize a Java object, which extracts information from the element. This object launches the recursive extraction call to carry out a X3DMM of the under tree X3D. To generate the X3D document, the same recursive method is used. The element root writes in a buffer the values of these attributes, and then launches the extraction command to each child. Once arrived at the end of the algorithm, the buffer is transmitted in a flow of file. Then a style sheet is applied to this X3D document maps it into a VRML file.

The extraction process is not limited to the display of 3D-scenes. The SQL extraction queries can also manipulate objects appearing in scenes. Inside the database, the scenes are not stored as a single file but they are decomposed into many tables corresponding to all components composing the scenes. One of the advantages of this organization is the possibility of extraction of objects shared by several scenes. Thus, many scenes can be modified at the same time. Instead of interrogating different stages one by one, it is sufficient to use one SQL query to modify objects composing the scenes stored in the database.

A scene is modeled as a tree. In addition, objects themselves are represented as trees. An object contains a number of components. For example, a table consists of a tray and four legs. It is possible to define appearance attributes for each of these elements or for broadly speaking of all elements. In fact, a tag called «Shape» which contains the physical and visual descriptions of an object defines 3D objects: its dimensions, its color, etc. Therefore, it is possible to define a name for every "Shape" tag. With this name, it is possible to access the elements and features of an object. The modification of attributes is performed by SQL queries. In the below, we give several examples of SQL queries that allow to manipulate scenes.

```
Select Shape_id, DEF SHAPE from SHAPE
where DEF is not NULL;
```

This request permits to select all Shape tags that compose all scenes stored in the database and to display certain features when DEF attributes of Shape is not Null. In the following query, we update a specific instance of "Material" tag by changing the value of its transparency attribute.

```
update Material set TRANSPARENCY = '0.3'
where material_id =’10’ ;
```

The major advantage of this method lies in the fact that

such a modification operates either on a selected scene or on all scenes in the databases. The automatic execution of the modification reduces the cost of manipulation of scenes.

The construction of the 3D-Extraction Web-Service reuses the low-level 3D-Insertion Web-Service building because the same tables are manipulated. This Web-Service exports its results as X3D documents which are converted in VRML or other structures using XSL. A Set of specific method is defined for the partial extraction of 3D objects from the scenes. Currently, we are developing an interface allowing a final user to query directly the scenes through a Web-service using a set of parameters.

6. Conclusion and future work

In this article, we have presented a method of modeling and storing of 3D-scenes in a relational database. We use the X3D language as an underlying model. This language, based on the XML standard, is compatible with VRML. We have constructed a functional architecture of insertion and extraction of 3D-scenes into and from a database. This architecture permits us to manipulate 3D-scenes by SQL queries.

In the setting of an industrial project, we currently strive to set up an interoperability architecture that associates our 3D-database with standard databases. The final objective is to build an Internet environment.

To achieve this goal, we use Web-services. These Web-services give access the database transparent of way for the users. We will develop the use of the standards, which constitute the pivot set of the Web-services. The next step will be the use of WSDL and the creation of a directory UDDI. This directory will be a Web-service, which will be to implement into Java and with the DBMS managing our three databases (Management, IFC and X3D). An axis of significant development will be the Web-services security. Initially, we will modify our architecture by replacing protocol HTTP by its secured version (HTTPS). After, a solution to this problem includes encryption technologies (like XML Encryption [26]) and digital signature technologies (like XML Signature [27]).

7. Bibliography.

[1] Active3D: http://www.active3d.net

[2] CORBA, http://www.corba.org

[3] Christophe Cruz, Christophe Nicolle, Marc Neveu, ACTIVe3D : Interrogation de scènes 3D en SQL, Medianet, 2002, Tunisie.

[4] HP web services platform: a comparison with hp e-speak executive, 20001, http://www.bluestone.com/downloads/pdf/espeak_webservices.pdf

[5] IONA Technologies PLC, April 2002, White Paper Orbix E2A XMLBus Edition Technology Overview, http://www.xmlbus.com/learn/webserviceswp.pdf

[6] IBM Web Services Architecture Team, Web Services architecture overview: The next stage of evolution for e-business, by, September 2000, ftp://www6.software.ibm.com/software/developer/library/w-ovr.pdf

[7] Java XML Pack, http://java.sun.com/xml/javaxmlpack.html

[8] Microsoft Developer Network, A Platform for Web Services, by Mary Kirtland, January 2001. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/websvcs_platform.asp

[9] Meta Group, http://www.bea.com/events/integrate/pdf/METAParis.pdf, 24 October, 2001, Dr. Bjorn Tuft Vice President International Enterprise Architecture Strategies

[10] Judith M. Myerson, Web Service Architectures, http://www.webservicesarchitect.com/content/articles/webservicesarchitectures.pdf

[11] W J Pardi, XML, in Action, 1999, Microsoft Press, http://www.microsoft.com/france/mspress.

[12] X3D Scene Access Interface (SAI), http://www.web3d.org/TaskGroups/x3d/sai/SceneAccessInterface.html

[13] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000, http://www.w3.org/TR/SOAP/

[14] Standard for the Exchange of Product model data (STEP): http://www.nist.gov/sc4/www/stepdocs.htm

[15] Universal Description, Discovery and Integration, http://www.uddi.org/specification.html

[16] International Web3D specifications VRML97 Standard: http://www.vrml.org/fs_workinggroups.htm

[17] Web3D Consortium: http://www.web3d.org

[18] Web Services Description Language (WSDL) 1.1; W3C Note 15 March 2001, http://www.w3.org/TR/wsdl

[19] W3C workshop on Web services: Position papers11-12 April 2001 - San Jose, CA, USA, http://www.w3.org/2001/03/WSWS-popa/

[20] The World Wide Web Consortium (W3C) develops interoperable technologies: http://www.w3c.org

[21] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[22] Document Object Model (DOM), http://www.w3.org/DOM/

[23] About SAX, http://www.saxproject.org/

[24] XSLT (1999). XSL Transformations (XSLT), Version 1.0, W3C Recommendation 16 November, 1999, http://www.w3.org/TR/xslt

[25] What Is .NET?, http://www.microsoft.com/net/defined/default.asp

[26] XML Encryption WG, http://www.w3.org/Encryption/2001/

[27] XML Signature WG, http://www.w3.org/Signature/