

INSTITUT DES SCIENCES DE L'INGENIEUR

C/U/S/T  
UNIVERSITE BLAISE PASCAL  
CLERMONT-FERRAND 2

***RAPPORT DE STAGE DE 2ème ANNEE  
DEPARTEMENT : GENIE  
MATHEMATIQUE ET MODELISATION***

---

***BLIS-XML CONVERSION TOOL***

---

SLAMA Lise  
GMM . SEPTEMBRE 2002  
VTT

## Acknowledgements

The work I made would not have been possible without the support and the technical help from a lot of people. I would like to acknowledge them. I think in particular at Yoshinobu Adachi, a specialist of the IFC who took a lot of time to follow my work, at Ilkka Heinonen who also help me a lot and Juha Hyvärinen.

I have received also a lot of help for my accommodation. I acknowledge Pekka and Jaana to have found an apartment for me and all the furniture I need.

Then I thank Janne who often helped me when I needed a translator, Pekka for his TV, Tina for her bike and all the other people for their help and advice.

# Abstract

For several years, some people have been working on the improvement of the interoperability between the different disciplines of the building industry. For that, an exchange format has been created based on a set of classes called Industry Foundation Classes (IFC).

At the beginning this format was respecting the STEP norm. The files respecting this format are called STEP Part21 files. Nowadays, the trend is to utilize the eXtensible Markup Language (XML) to describe this data. There are two syntaxes using XML : the first one is called ifcXML and the second one is BLIS-XML.

The aim of this work is to write a little program in the Java language, converting BLIS-XML data to IfcXML and STEP Part21. VTT already provides an IFC Model server on the internet where we can ask some data about a building. That data is in BLIS-XML format. My programme permits to have the data in the language needed by the user.

For this work, I have used specific tools for handling XML data. Firstly I needed some classes which parse and create a DOM object, then I needed XSLT files and the objects handling them. Finally I have used the classes which enable to create XML files respecting the SOAP protocol.

The work was completely made but there are still some things to improve. Actually I did not manage to solve the problem of memory that I was confronted to. Mais my programme is open source so the interested developers can improve my code, or just use the classes they need.

**Key words** : convert IFC STEP, XML, Java

## Résumé

Depuis quelques années des personnes d'un peu partout dans le monde, travaillent sur l'amélioration de l'interopérabilité entre les différents corps d'état du bâtiment. Dans ce but, un format d'échange a été créé, basé sur un ensemble de classes appelées Industry Foundation Classes (IFC).

Au départ, ce format devait respecter la norme STEP. On appelle les fichiers respectant ce format des fichiers STEP Part21. Aujourd'hui, la tendance est d'utiliser le langage XML ( eXtensible Markup Language) pour décrire ces données. Il existe deux syntaxes de fichiers IFC écrits en XML : la première est appelée IfcXML et la second BLIS-XML.

Le but de ce travail est d'écrire un petit programme en Java qui convertisse des données BLIS-XML en IfcXML ou en STEP Part21. VTT a développé un serveur IFC où l'on peut demander des données concernant une habitation. Ces données sont en BLIS-XML. Mon programme permet donc de convertir ces données dans le format requis par l'utilisateur.

Au cours de ce travail j'ai utilisé des outils spécifiques à la manipulation de données XML. Tout d'abord j'ai eu besoin de classes qui analysent les données XML et créent des objets DOM, puis de fichiers XSLT et des objets les manipulant. Enfin j'ai utilisé des classes faites pour faciliter la création et la lecture de fichiers d'échange XML respectant le protocole SOAP.

Ce travail fut mené jusqu'au bout mais il y a encore des améliorations que l'on peut apporter à mon programme. En effet, je n'ai pas résolu les problèmes de mémoire auxquels j'ai été confrontée. Mais mon programme est ouvert. Ainsi les développeurs intéressés peuvent apporter des améliorations à mon programme, ou simplement extraire les classes dont ils ont besoin.

**Mots clé :** convertir IFC, STEP, XML, Java

# Contents

<b>1</b>	<b>Finland and VTT</b>	<b>8</b>
1.1	Finland . . . . .	8
1.2	VTT . . . . .	8
1.3	VTT : Building and Transport service . . . . .	10
<b>2</b>	<b>The IAI and the IFC</b>	<b>11</b>
2.1	The building industry . . . . .	11
2.2	The IAI . . . . .	11
2.3	BLIS project . . . . .	12
2.4	What about now ? . . . . .	12
2.5	The projects . . . . .	12
2.6	The IFC limits . . . . .	13
<b>3</b>	<b>The languages used</b>	<b>14</b>
3.1	Java . . . . .	14
3.2	STEP . . . . .	14
3.3	XML . . . . .	17
3.3.1	Overview . . . . .	17
3.3.2	The important things to know about XML . . . . .	18
<b>4</b>	<b>My work</b>	<b>20</b>
4.1	First part . . . . .	20
4.1.1	Topic . . . . .	20
4.1.2	JBuilder . . . . .	21
4.1.3	XSLT . . . . .	22
4.1.4	What is it ? . . . . .	22
4.1.5	How to implement it ? . . . . .	23
4.2	The second part of the work . . . . .	25

4.2.1	Topic . . . . .	25
4.2.2	Two solutions . . . . .	26
4.2.3	DOM . . . . .	26
4.2.4	The differences between BLIS-XML and IfcXml . . . . .	29
4.2.5	My implementation . . . . .	31
4.3	The third part . . . . .	33
4.3.1	Topic . . . . .	33
4.3.2	SOAP . . . . .	33
4.3.3	My implementation . . . . .	38
4.4	Using this programme . . . . .	38
<b>5</b>	<b>Conclusion</b>	<b>40</b>

# List of Figures

1.1	Turnover in 2001 . . . . .	9
1.2	External income in 2001 . . . . .	9
3.1	STEP document structure . . . . .	15
3.2	XML file . . . . .	17
3.3	XML file modified . . . . .	17
4.1	Outline of XSLT based Converter . . . . .	21
4.2	My first Graphic User Interface . . . . .	21
4.3	XSL file . . . . .	22
4.4	XML file displayed by Internet Explorer . . . . .	23
4.5	Outline of DOM based Converter . . . . .	25
4.6	DOM representation . . . . .	28
4.7	Outline of BLIS-XML to STEP P21 converted with SOAP . . . . .	33
4.8	My final GUI . . . . .	38

# Introduction

I have gone and worked at VTT for 3 months as part of my studies in engineering school. VTT is an independent institut of research very big and quite famous, even in other countries. It covers a lot of domains from Biotechnology to Building and Transport. It participates in several international projects in particular inside the International Alliance for Interoperability (IAI).

The IAI is an association of companies who want to improve the interoperability between the different companies working together in AEC / FM industry. Their main goal is to establish a norm for the data exchanges. This norm would be based on a set of class called Industry Foundation Classes.

Nowadays the IFC files, i.e the file which describe data obeying the IFC architecture, exist. They are written respecting the STEP norm or in XML. They are called STEP Part21. Since XML seems to be a very powerful a format, an XML version have also been made. More exactly two XML versions have been made : IfcXML and BLIS-XML. A server of BLIS-XML file have been developed. It will be soon possible to get IFC data file by the internet. But that data is in the BLIS-XML format, and the users can have software reading IfcXML or STEP Part21 files.

My work is developing a program who convert a BLIS-XML file or BLIS-XML data from the IFC server into IFC-Xml or into STEP part 21.

# Chapter 1

## Finland and VTT

### 1.1 Finland

During my placement, I have discovered a very nice country, at least for the summer. Finland is a small country with only 5,2 millions of inhabitants. It has been an independent state since 1917. Nowadays it is famous thanks to their mobile phone : Nokia. Finnish are very good in new technologies because they invest a lot in : 3 % of the gross domestic is for the research and development. What 's more, as the country have been socialist for a long time and there were a lots of bombing, the finnish towns are built with cleverness. It is quiet and pleasant to live there

### 1.2 VTT

VTT is an independent and multidisciplinary technical research institute. It support the production of goods and services in Finland through research, development, testing, and technology transfer. It combines contract research and development services with long-term strategic research to meet the national end international customers.

With its 2907 employees, VTT provides a wide range of technology and applied research services for its clients, private companies, institutions and the public sector. It is divided in the following services: VTT Electronics, VTT Information Technology, VTT Industrial Systems, VTT Processes, VTT Biotechnology, VTT Building and Transport, VTT Information Service.

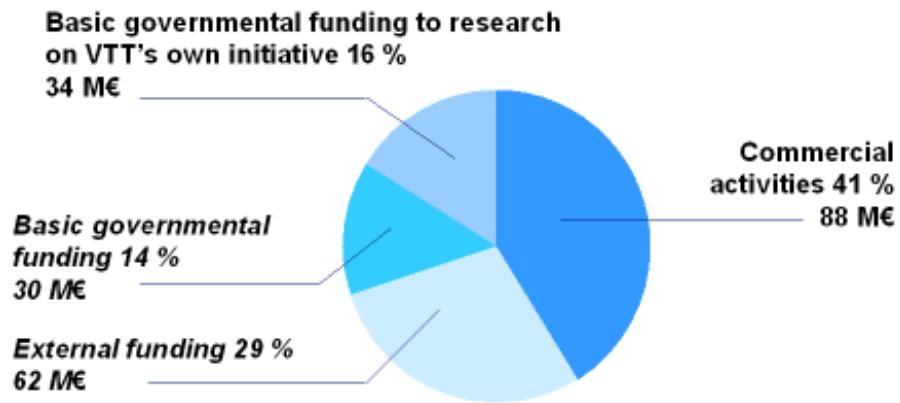


Figure 1.1: Turnover in 2001

Its turnover was 214 million euros in 2001. Its external income (146 M€) is principally from the private sector, as we can see figure 1.2.

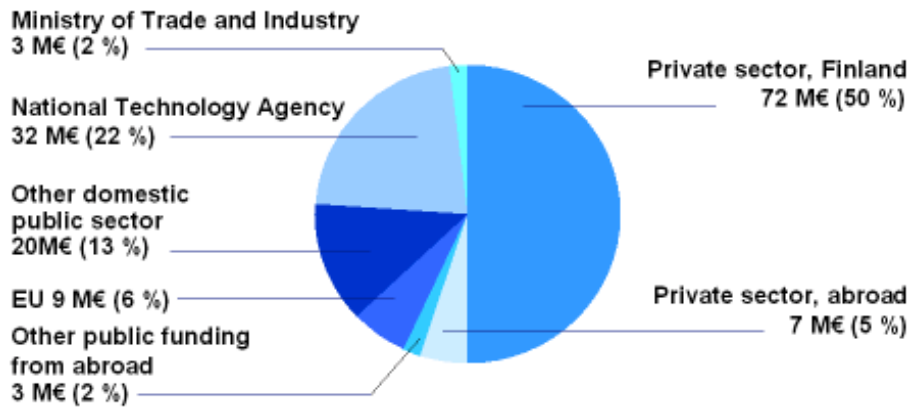


Figure 1.2: External income in 2001

### **1.3 VTT : Building and Transport service**

I was in the Building and Transport service. The field of VTT Building and Transport encompasses buildings and facilities together with their related products, systems, processes and functions; urban planning; transport and logistics; road, civil engineering and rock engineering structures; and environmental technology for communities. The number of its employees is 475. Turnover is about 39 million euros.

# Chapter 2

## The IAI and the IFC

My project belongs to a set of work around the IFC that were created by the IAI. Before looking my work, we need to see the reason why the IFC exist. So we are first going to see quickly the history of thoses classes.

### 2.1 The building industry

The world of the building is huge with a lot of proper fields. There are in the same time a lot of products that we can choose and a lot of people who work on the same building. Then for each particular field there are specific computer tools. So when two persons of different field have to exchange data about their project, they must define, or define again what format they use.

### 2.2 The IAI

We easily see that the complexity of the building industry implies a lot of problems for the data exchange. It was to solve a part of those problem that the International Organization for Standardization (ISO) decided to created the IFC. But research were very slow. So several companies decide to gather and to finish the work and created in this order an association called International Alliance for Interoperability (IAI). Since its formation in September 1995, the IAI has published 3 major releases of the Industry Foundation Classes (IFC). These releases consisted of specifications defining an object based data model for the AEC industry. The IAI's goal was that

these specifications would support software implementation by multiple vendors, such that the software of these vendors could exchange and interoperate on a commonly understood data model of a building project.

## 2.3 BLIS project

Then in 1999, some members of the IAI launch a project called "Building Lifecycle Interoperable Software" (BLIS). While IAI projects focus on requirements and specification of the IFC model. BLIS supports implementation of IFC specifications in software products. One part of BLIS interest us particularly. It is the BLIS-XML files. The participants of BLIS have thought that it was easier to add in existing software a part able to read and write XML files than STEP files. It is easier because it exists parsers which read XML file and create object in different languages such as JAVA, C++. We will see later more deeply what is the XML and what is the object resulting.

## 2.4 What about now ?

Nowadays the last release of IFC called IFC2x is recognized available. It means that they are able to describe every buildings. Of course, there still are projects of extension for thoses classes, for example in the field of the bridge [F. Boulaire (2002)] . So the model exists. What's more, some softwares using it have been developped. That is very important because they prove the validity of the model.

## 2.5 The projects

One of the next steps to facilitate the interoperability is the creation of a IFC server. Then the different persons who work on the construction of a building have access whenever they want to the file describing the building. One of my "colleagues" was charged to develop this server. So I had the opportunity to use this server in my program like you are going to see in the next part.

We have to add that nowadays the IAI try to make the ISO recognize the IFC which are just a standard for the moment, as a norm.

## **2.6 The IFC limits**

Thus a lot of things have been done for the interoperability in the building industry. Nevertheless it is still quite experimental for the main reason that people who should use those tools do not know them yet. There are a important work to do to inform them what are the IFC, what IFC will change in their habits and what the interests of IFC are.

# Chapter 3

## The languages used

The goal of the placement was realizing a programme which converts BLIS-XML file into IfcXML file or into STEP Part21. This programme is written in Java. To understand what it does exactly, we have first to learn the Java language, the STEP norm and the XML.

### 3.1 Java

Java is a language object-oriented very similar to C++ which has been created in 1991 by Sun. Its particularity is that the source is not encoded in machine code but in a universal language, formed by byte code. This guarantees the portability. It means that those bytes codes can be read by all the implementations which have a Java Virtual Machine (JVM). Moreover, Java defines exactly the characteristics of floats and integers. Thus that assures that the same programmes executed within different environments will return the same results. All that qualities make the Java language very useful for applets in web pages.

We can find all Java knowledge base in [C. Delannoy (2000)] which is written in French.

### 3.2 STEP

STEP is the name of a standard developed by the International Organization of Normalization (ISO). We can see in the figure 3.2 the STEP document

structure

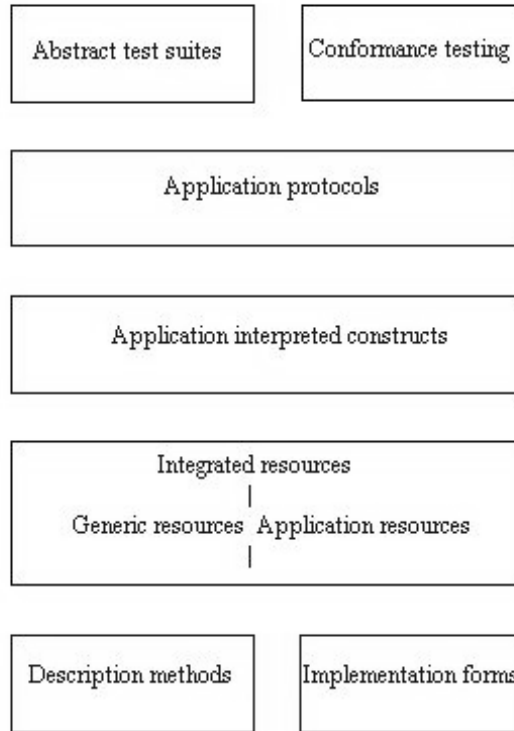


Figure 3.1: STEP document structure

The part containing data called "implementation form" is in Express language. Express is a very simple language able to describe object such as classes in C++. Each object has an associated number. We can use this number as a reference of this object. So when a number appears in the part into the brackets it means that the object defined into this line has as attribute the object associated to the number.

We can see a very short example of STEP Part21 file in the table 3.1.

```

ISO-10303-21;
HEADER;
FILE_DESCRIPTION (('Handmade eaxple IFC file'), '2;1');
FILE_NAME ('IFC2x_example_1.ifc', '2002-03-18T22:37:43',
('NA'), ('NA'), 'Notepad', 'Windows System', 'rtejph');
FILE_SCHEMA (('IFC2X_FINAL'));
ENDSEC;
DATA;
#1 = IFCORGANIZATION ('MS', 'Microsoft (TM) Corp.', $, $, $);
#2 = IFCAPPLICATION (#1, '5.0', 'Notepad 5', 'Microsoft (TM)');
#3 = IFCPERSON (rtejph, Hyvarinen, Juha, $, $, $, $, $);
#4 = IFCORGANIZATION ($, 'VTT', $, $, $);
#5 = IFCPERSONANDORGANIZATION (#3, #4, $);
#6 = IFCOWNERHISTORY (#5, #2, $, .NOCHANGE., $, $, $, 1016483863);
#7 = IFCSIUNIT (*, .LENGTHUNIT., .MILLI., .METRE.);
#8 = IFCSIUNIT (*, .AREAUNIT., $, .SQUARE_METRE.);
#9 = IFCSIUNIT (*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#10 = IFCSIUNIT (*, .MASSUNIT., $, .GRAM.);
#11 = IFCSIUNIT (*, .TIMEUNIT., $, .SECOND.);
#12 = IFCSIUNIT (*, .THERMODYNAMICTEMPERATUREUNIT., $, .DEGREE_CELSIUS.);
#13 = IFCSIUNIT (*, .LUMINOUSINTENSITYUNIT., $, .LUMEN.);
#14 = IFCUNITASSIGNMENT ((#7, #8, #9, #10, #11, #12, #13));
#15 = IFCREPRESENTATIONCONTEXT ('Plan', 'Design');
#16 = IFCPROJECT ('2MKmiT1abCYfZJyRedPW7f', #6,
'Ifc2x implementation example', $, $, $, $, (#15), #14);
ENDSEC;
END-ISO-10303-21;

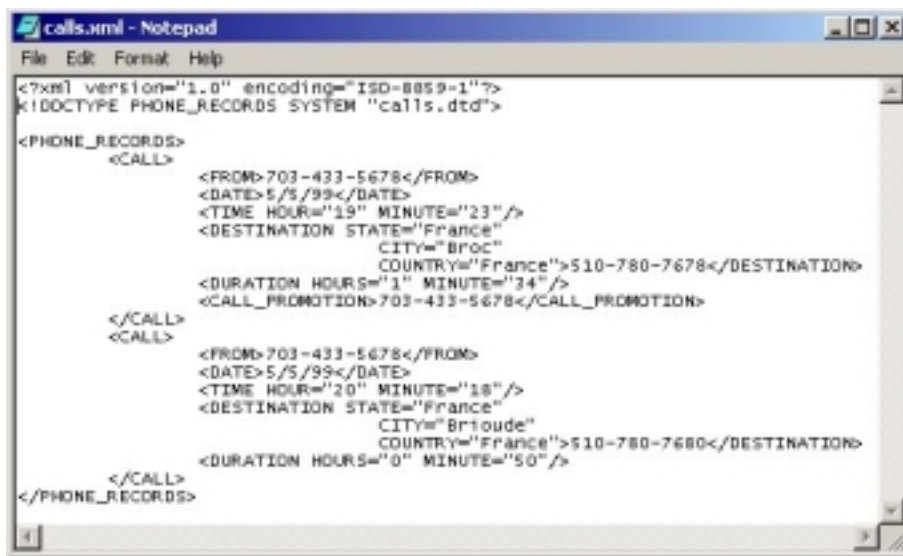
```

Table 3.1: Example of IFC file

## 3.3 XML

### 3.3.1 Overview

XML is the eXtensible Markup Language. XML is Markup because it is combination of text (content) and textual codes (tags or markup) that describe the nature of the content. XML is eXtensible which means that we are not confined to a predefined set of tags, we can define our own to suit our own application. We can see a simple example of XML file figure 3.2. Contrary



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE PHONE_RECORDS SYSTEM "calls.dtd">
<PHONE_RECORDS>
  <CALL>
    <FROM>703-433-5678</FROM>
    <DATE>5/5/99</DATE>
    <TIME HOUR="19" MINUTE="23"/>
    <DESTINATION STATE="France"
      CITY="Broc"
      COUNTRY="France">510-780-7678</DESTINATION>
    <DURATION HOURS="1" MINUTE="34"/>
    <CALL_PROMOTION>703-433-5678</CALL_PROMOTION>
  </CALL>
  <CALL>
    <FROM>703-433-5678</FROM>
    <DATE>5/5/99</DATE>
    <TIME HOUR="20" MINUTE="18"/>
    <DESTINATION STATE="France"
      CITY="Brioude"
      COUNTRY="France">510-780-7680</DESTINATION>
    <DURATION HOURS="0" MINUTE="50"/>
  </CALL>
</PHONE_RECORDS>
```

Figure 3.2: XML file

to HTML which seems to be similar, XML file does not give any information about how to display its content. But we can give the instructions for the display in an eXtensible Stylesheet Language (XSL) file that we associate to the XML file adding one line as we can see figure 3.3.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="calls.xsl"?>
<!DOCTYPE PHONE_RECORDS SYSTEM "calls.dtd">
```

Figure 3.3: XML file modified

So depending on the use, we can choose a particular XSL to describe the same data.

### 3.3.2 The important things to know about XML

XML is for structuring data, it means things like spreadsheets, address books, configuration parameters, financial transactions, and technical drawings. It is a set of rules for designing text formats that let you structure your data. It is not a programming language, and we don't have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous.

As we have seen, it is text. One advantage of a text format is that it allows people, if necessary, to look at the data without the program that produced it; in a pinch, we can read a text format with your favorite text editor. Text formats also allow developers to more easily debug applications.

Since XML is a text format and it uses tags to delimit the data, XML files are nearly always larger than comparable binary formats. That was a conscious decision by the designers of XML. We just have seen the advantages of a text format, and the disadvantages can usually be compensated at a different level. Disk space is less expensive than it used to be, and compression programmes like zip and gzip can compress files very well and very fast. In addition, communication protocols such as modem protocols and HTTP, the core protocol of the Web, can compress data on the fly, saving bandwidth as effectively as a binary format

XML is a family of technologies. XML 1.0 is the specification that defines what "tags" and "attributes" are. Beyond XML 1.0, "the XML family" is a growing set of modules that offer useful services to accomplish important and frequently demanded tasks. for instance, XSL is the advanced language for expressing style sheets. It is based on XSLT, a transformation language used for rearranging, adding and deleting tags and attributes. The DOM is a standard set of function calls for manipulating XML (and HTML) files from a programming language. There are several more modules and tools available or under development and we can find information about from <http://www.w3.org/XML/>. In particular, it is very easy to handle XML file

within a Java programme [K. Ahmed and al. (2001)]. We can find more details about XML and the tools around XML from <http://www.w3.org/XML/>.

Last but not least, XML is license-free, platform-independent and well-supported. By choosing XML as the basis for a project, we gain access to a large and growing community of tools and engineers experienced in the technology. And since XML is license-free, we can build your own software around it without paying anybody anything. The large and growing support means that we are also not tied to a single vendor.

# Chapter 4

## My work

Then the development of the programme can be divided into three parts. The first one concerns the conversion from BLIS-XML files to STEP Part21 files. In the second one, we focus on the conversion from the same files to IfcXML files. Then in the last part we are adding a function that recover BLIS-XML data by internet. After that, those data are converted like the previous BLIS-XML files. The last thing to do is to be interested by the future use of this programm.

### 4.1 First part

#### 4.1.1 Topic

This implementation aims to realize the conversion from BLIS-XML to STEP Part21 file format by utilizing XSLT as we can see figure 4.1. We are going to see what XSLT is exactly, but first let's take a look to my main tool : JBuilder.

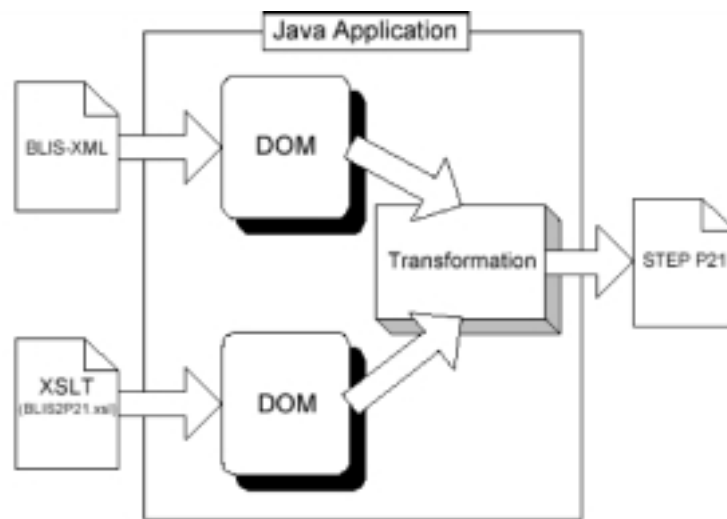


Figure 4.1: Outline of XSLT based Converter

### 4.1.2 JBuilder

JBuilder is development environment which contains a Java compiler with lots of options helping the development. We use JBuilder like Visual C++ that I have already known. Thus I have not had a lot problems to create my first java project. We can see my first graphic user interface figure 4.2.

Now let's see what the code may be put behind. For that, we need to know more about XML, in particular about one sort of XML files named XSLT.



Figure 4.2: My first Graphic User Interface

```

<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/TR/REC-html40"
>
<xsl:template match="PHONE_RECORDS">
<html><head> <title> Phone listing</title></head>
<body>
<h1>Phone call records</h1>
<table border="1">
<thead><tr>
<th>source number</th>
<th>destination</th>
<th>date</th>
<th>time</th>
</tr></thead>
<xsl:apply-templates />
</table>
</body>
</html>
</xsl:template>
<xsl:template match="CALL">
<tr>
<td><xsl:number /></td>
<xsl:for-each select="FROM|DESTINATION|DATE">
<td><xsl:value-of select="."/ ></td>
</xsl:for-each>
<td><xsl:apply-templates select="TIME" /></td>
</tr>
</xsl:template>
<xsl:template match="TIME">
<xsl:value-of select ="@HOUR"/>:
<xsl:value-of select ="@MINUTE"/>
</xsl:template>
</xsl:stylesheet>

```

Figure 4.3: XSL file

### 4.1.3 XSLT

#### 4.1.4 What is it ?

XSLT means eXtensible Stylesheet Language for Transformations. It is a transformation language used for rearranging, adding and deleting tags and attributes. First we can have a look to an simple example; we look at the display which is one of the simplest transformation. We can see figure 4.3 the file called "calls.xsl" which have been made to transform "call.xml" that we have seen before, to a html file.

Then when we open the "calls.xml" file with a browser we obtain the frame figure 4.4. For that, we need an XSLT compiler but all recent browsers have one.

But how it runs? The primary XML processing feature in XSLT is to apply "template" procedure to matching XML element in the source document. An XSLT template directive is specified with the `<xsl:template>` element. First the compiler look at the template element which treat the root.

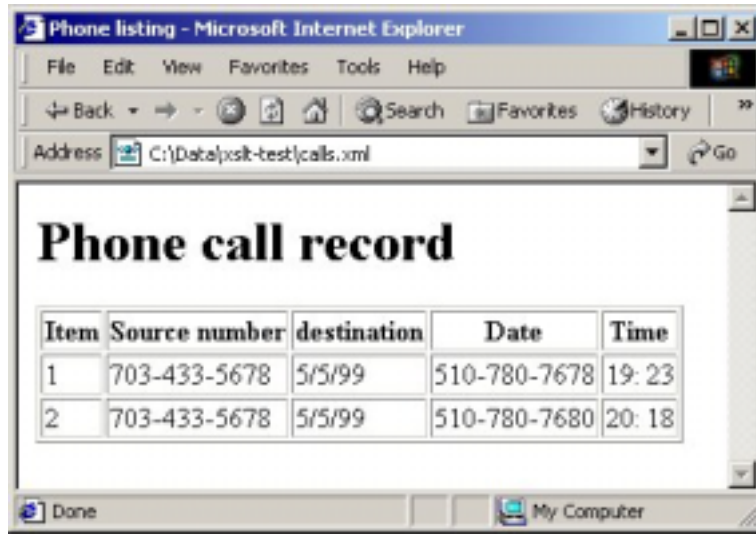


Figure 4.4: XML file displayed by Internet Explorer

In this `< xsl : apply - templates/ >` call the others template element associated to the element which are child of the root.

Inside this, the HTML syntax is used. We can also use as in the next part of the file some function such as `xsl:number` or `xsl:value-of`. They are functions which are gather in a namespace called `xsd`.

So XSLT enable to transform the XML file in a structured manner with a set of functions. For more details please see [K. Ahmed and al. (2001), chapter 7].

My programme uses two XSLT files to convert BLIS-IFC file into STEP Part21. Ther two XSLT files to make possible the conversion of the IFC2.0 as well as the IFC2x's. They have been realized by Yoshinobu Adachi. They are quite big, about 150 KB, so unfortunately I cannot print it in this report. But we can find it from <http://cic.vtt.fi/projects/ifcsvr/bliscon/index.html>

#### 4.1.5 How to implement it ?

Java owns classes which make transformation of XML file by XSLT. We just need to find them and introduce in our project. After that, the code is simple. The library used is called Xalan, and we can find it from:

<http://xml.apache.org/xalan-j/>. Then we write a specific class for this conversion. The main line of the unique function of this class are table 4.1.

```
TransformerFactory tFactory = TransformerFactory.newInstance();
Transformer transformer; if (version==0)transformer =
tFactory.newTransformer(new StreamSource("blis2p21_IFCR20.xslt"));
else transformer = tFactory.newTransformer(new
StreamSource("blis2p21_IFC2x.xslt"));
    transformer.transform(newStreamSource(sourceName),
    new StreamResult(new FileOutputStream(targetName)));
```

Table 4.1: Implementation of XSLT

The complete code is in the appendix.

## 4.2 The second part of the work

### 4.2.1 Topic

This implementation aims to develop DOM (Document Object Model) parser based application that converts BLIS-XML to ifcXML. The application imports BLIS-XML by DOM, and converts to ifcXML format as a DOM tree. Figure shows the outline of DOM based BLIS-XML to STEP P21 converter.

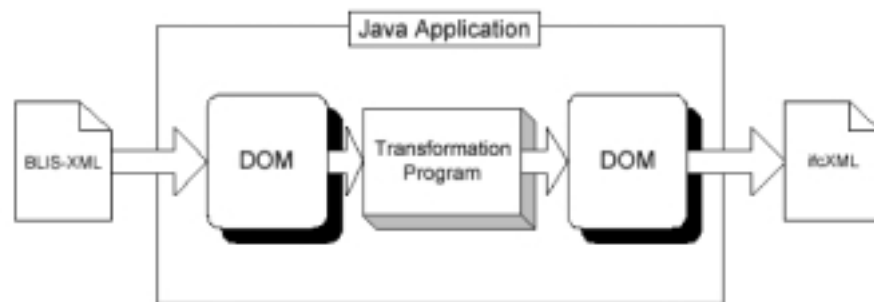


Figure 4.5: Outline of DOM based Converter

### 4.2.2 Two solutions

There are two possible way to do that. The first one is to implement the transformation in Java. The second one is to use an XSLT file. So in this way, we have creates an XSLT file for this specific transformation and introduce it as in the previous part. Its interest is that the XSLT file can be used again for lots of others application. With the first manner, we use the DOM classes. Its the opportunity to discover another technology handling XML. What is more, this manner is easier. Since I had not a lot of time, I decide to do this way.

### 4.2.3 DOM

The Document Object Model (DOM) is an Application Programming Interface (API) for HTML and XML documents. It is not implementation but simply a package of Java Interfaces defining the API. The DOM closely resembles the structure of the document it models; let's take a look at a simple example, figure 4.6.

The DOM Document appears to have a tree-like structure. Each rectangle is a node and we can ask it to give us a reference to its children or its parent. Here, if we had a reference to the first PART, we could access its parent, the BOOK node, or its children, the CHAPTER nodes. So we see that the manipulation of the XML's element is easy with DOM [C. Delannoy (2000)][chapter 3].

We can download the library needed, the Apache Xerces-J Parsing (JAXP) from <http://xml.apache.org/xerces>, but it is already contented Xalan.

```

<BOOK>
  <TITLE>La revolution russe</TITLE>
  <AUTHOR>LEON TROTSKY</AUTHOR>
  <PART NUMBER="1" NAME="La revolution de fevrier">
    <CHAPTER NUMBER="1" NAME="Le developpement
    de la Russie"/>
    <CHAPTER NUMBER="2" NAME="La Russie tsariste et la
    guerre"/>
  </PART>
  <PARTIE NUMBER="2" NAME="La revolution d'octobre">
  </PARTIE>
</BOOK>

```

The DOM represents this BOOK like this:

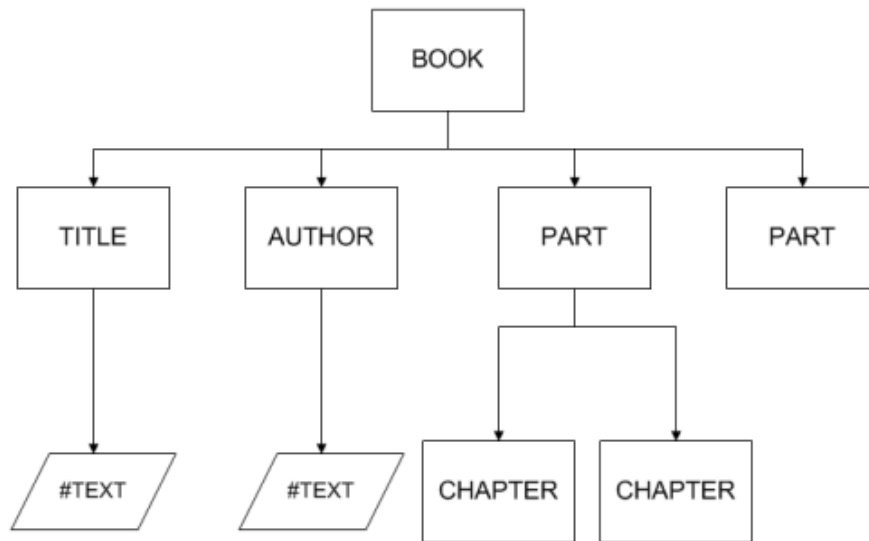


Figure 4.6: DOM representation

#### 4.2.4 The differences between BLIS-XML and IfcXml

BLIS-XML and IfcXml are very similar : they are XML files and they describe exactly the same thing. But their syntax are different.

First we have to note that IfcXml is not a format but a methodology. It means that it proposes several approaches to describe data. We are going to consider the description of the utilized units it means the class UnitAssignment. This one has got some attributes which belongs to the class SIUnit, which is defined by thoses three attributes : unitType, prefix, name. Each object has an identifier called "id". Now we can see the declaration of millimetre, square metre and cubic metre as global unit assignments, using the containment approach.

```
<UnitAssignment id="_1001">
  <units>
    <SIUnit id="_1002">
      <unitType>lengthunit</unitType>
      <prefix>milli</prefix>
      <name>metre</name>
    </SIUnit>
    <SIUnit id="_1003">
      <unitType>areaunit</unitType>
      <name>square_metre</name>
    </SIUnit>
    <SIUnit id="_1004">
      <unitType>volumeunit</unitType>
      <name>cubic_metre</name>
    </SIUnit>
  </units>
</UnitAssignment>
```

And the same data can be described with the reference approach:

```
<UnitAssignment>
  <units>
    <SIUnit href="_1002" xsi:nil="true"/>
    <SIUnit href="_1003" xsi:nil="true"/>
  </units>
</UnitAssignment>
```

```

        <SIUnit href="_1004" xsi:nil="true"/>
        <SIUnit href="_1005" xsi:nil="true"/>
    </units>
</UnitAssignment>

<SIUnit id="_1002">
    <unitType>lengthunit</unitType>
    <prefix>milli</prefix>
    <name>metre</name>
</SIUnit>
<SIUnit id="_1103">
    <unitType>areaunit</unitType>
    <name>square_metre</name>
</SIUnit>
<SIUnit id="_1104">
    <unitType>volumeunit</unitType>
    <name>cubic_metre</name>
</SIUnit>
<SIUnit id="_1105">
    <unitType>planeangleunit</unitType>
    <name>radian</name>
</SIUnit>

```

Now, we look at the BLIS-XML format. In BLIS-XML all the elements are at the same level, except the FILE\_HEADER\_SECTION element which have children elements. In the others cases, the information about an element is not another element content, but an attribute within the markup. Let's how the previous example is written in BLIS-XML :

```

<IfcSiUnit XMLID="i1102" UnitType="LengthUnit" Prefix="MILLI"
Name="METRE" />
<IfcSiUnit XMLID="i1003" UnitType="AreaUnit"
Prefix="" Name="SQUARE_METRE" />
<IfcSiUnit XMLID="i1004" UnitType="VolumeUnit" Prefix=""
Name="CUBIC_METRE" />
<IfcSiUnit XMLID="i1005" UnitType="PlaneAngleUnit" Prefix=""
Name="RADIAN" />

```

```
<IfcUnitAssignment XMLID="i961" Units="i1002 i1003 i1004 i1005" />
```

This structure is nearer to the reference approach. I chose to have this approach. The difficulty is now to create a structure with several levels.

Another thing to change is the name. In BLIS-XML, the name of the objects start by "Ifc". We just have to remove those three letters.

Last but not least, the references in BLIS-XML start by "i" and can be enumerated, as we saw in the last example. This enumeration appears in a data type that is aggregation of reference. In IfcXml, the references start by "\_" and there is not such kind of enumeration. It means that we had to read the string of the enumeration, recognizes and cut the different references within.

We can see in the appendix, an example of BLIS-XML and the IfcXML file obtain with the programme.

#### **4.2.5 My implementation**

A new class has been made for this conversion. It contents four methods ( for the moment but a last one will be added for the conversion from the XML data downloaded ). The following tabular sums up what they do:

convertFromFile	<p>This function converts BLIS-XML file to ifcXML file using the DOM parser. It creates 2 DOM objects, one by parsing the source file, the other is created empty then is completed by calling the functions "creation" and "completeName". A Transformer object is use to write this second DOM object in the target file.</p>
creation	<p>It is a recursive function which creates the elements of the ifcXML DOM object. For that, it calls "read". Ones of them do not have their final name but are called TempName</p>
read	<p>It returns the number "nb" of strings found in input "string". It also modifies some character that string may contain such as "&lt;", "&gt;" etc, because they cannot be used in a XML data</p>
completeName	<p>this function replace the element "TempName" by the same element with the final name that we find thank to the attribut href</p>

## 4.3 The third part

### 4.3.1 Topic

This implementation is similar to the case of BLIS-XML to STEP P21 with XSLT. The difference is that BLIS-XML is not a file, it comes from IFC Model Server by SOAP directly without a file. The SOAP library, or classes in Java, invokes IFC Model Server Web Service method, and catches BLIS-XML as a partial model data. The SOAP class passes the BLIS-XML data to the DOM, after that XSLT is applied for the transformation to the DOM.

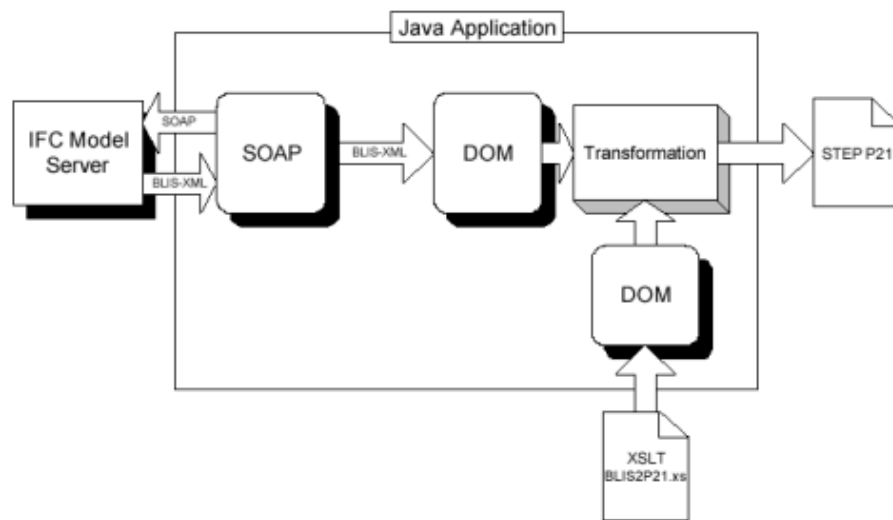


Figure 4.7: Outline of BLIS-XML to STEP P21 converted with SOAP

### 4.3.2 SOAP

#### Protocol

SOAP is the Simple Object Access Protocol. A protocol is a set of formal rules describing how to transmit data, especially across a network. Low level protocols define the electrical and physical standards to be observed, bit- and byte-ordering and the transmission and error detection and correction of the bit stream. HTTP is an example of low level protocol. High level protocols deal with the data formatting, including the syntax of messages, the terminal

to computer dialogue, character sets, sequencing of messages etc. SOAP is high level.

## Overview

SOAP is an XML based protocol, design to enable computers to talk to each other, regardless of their operating system, object model, or language environment. SOAP does not define a programming model or implementation specific semantics, such as an Application Programming Interface (API). It defines a simple mechanism for exchanging messages, by providing a modular packaging model and a way of encoding data within message.

The W3C (World Wide Web Consortium )acknowledged the submission of the protocol in May of 2000. So it is quite recent. It is the first protocol independent of any vendors. What ´s more, the older protocols are dependent of an operating system or a language platform. We can find more details about the specifications from <http://www.w3.org/TR/SOAP>.

SOAP messages are fundamentally one-way transmissions from a sender to a receiver. All SOAP messages are XML documents with their own schema, they include proper namespace on all element and attributes. SOAP defines two namespaces, **SOAP envelope** and **SOAP serialization** or **encoding**.

```

<SOAP-ENV:Envelope SOAP-ENV:encodingStyle=
'http://schemas.xmlsoap.org/soap/encoding/'
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'>
  <SOAP-ENV:Body>
    <SOAPSDK1:GetProjectList xmlns:SOAPSDK1=
      'http://tempuri.org/message/'>
      <strUserName>user1</strUserName>
      <strPassword>user1</strPassword>
    </SOAPSDK1:GetProjectList>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Table 4.2: Example SOAP file

As an XML document, SOAP messages consist of three section : the SOAP envelope, the SOAP header and the SOAP body. The SOAP header is optional.

### **Example of SOAP request for the IFC Model Server**

Let´s take a look of the XML file table 4.2.

The first and the last element are the envelope. The rest is the body. So the syntax is chosen by the programmer of the specific SOAP server.

We are now looking at the case of the IFC Model Server (IMS). The role of the IMS is to contain some project descriptions, and permit to the people working on one of those projects to extract the data they need. They also can change the data and then change the record of database of the IMS. Of course, this implies problem of user permission, but we are not going to interest more on the writing permission as well as the IMS is still a prototype, so all the practical problem are not yet solved. The IMS supports basic user access right control functions at the moment.

For the extraction, IMS proposes functions which permit to select the qualities of the object that we are interested in. The message contain the name of the function we want to call and the parameters needed. Let´s see deeper what is the body of our example:

```

<SOAPSDK1:GetProjectList
xmlns:SOAPSDK1='http://tempuri.org/message/'>
  <strUserName>user1</strUserName>
  <strPassword>user1</strPassword>
</SOAPSDK1:GetProjectList>

```

The function is "GetProjectList", and it requires two parameters which are "strUserName" and "strPassword". Their value is in this case "user1".

So, it is now quite easy to understand how is made a SOAP answer. Consequently, it is not necessary to develop this subject. Let's take a quick look of the response:

```

<?xml version="1.0" standalone="no" ?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body> - <SOAPSDK1:GetProjectListResponse
xmlns:SOAPSDK1="http://tempuri.org/message/">
  <Result>
  <IMS_Response>
    <project project_name="simple_model" schema_version="IFC20_LONGFORM" />
    <project project_name="blis_model" schema_version="IFC20_LONGFORM" />
    <project project_name="vtt_model" schema_version="IFC20_LONGFORM" />
    <project project_name="working" schema_version="IFC20_LONGFORM" />
  </IMS_Response>
  </Result>
</SOAPSDK1:GetProjectListResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## The Java tools

Java provides the classes which create the SOAP request and treat the SOAP answer. We usually use it in a web application but we can also use it in an application. For that we need :

- the SOAP library from <http://xml.apache.org/dist/soap/>. In our case we use the release 2.2.
- the JavaMail library called mail.jar from <http://java.sun.com/products/javamail/>
- the Java Bean Activation Framework called activation.jar from <http://java.sun.com/products/javabeans/glasgow/jaf.html>

### 4.3.3 My implementation

I have tried to write this part in a way that we can use again the classes in other cases. My programme has to get the project list and then, for one project, it have to get the the data of one special project. So there is one class to get the project list and one to take a complete project. In addition, one other class, the Soap class, calls the methods of those classes. It also use the classes made before to convert the data. It is an interface between the frame and the other classes. We can see a flowchart which summarize the relationship between the classes in the appendix.

## 4.4 Using this programme

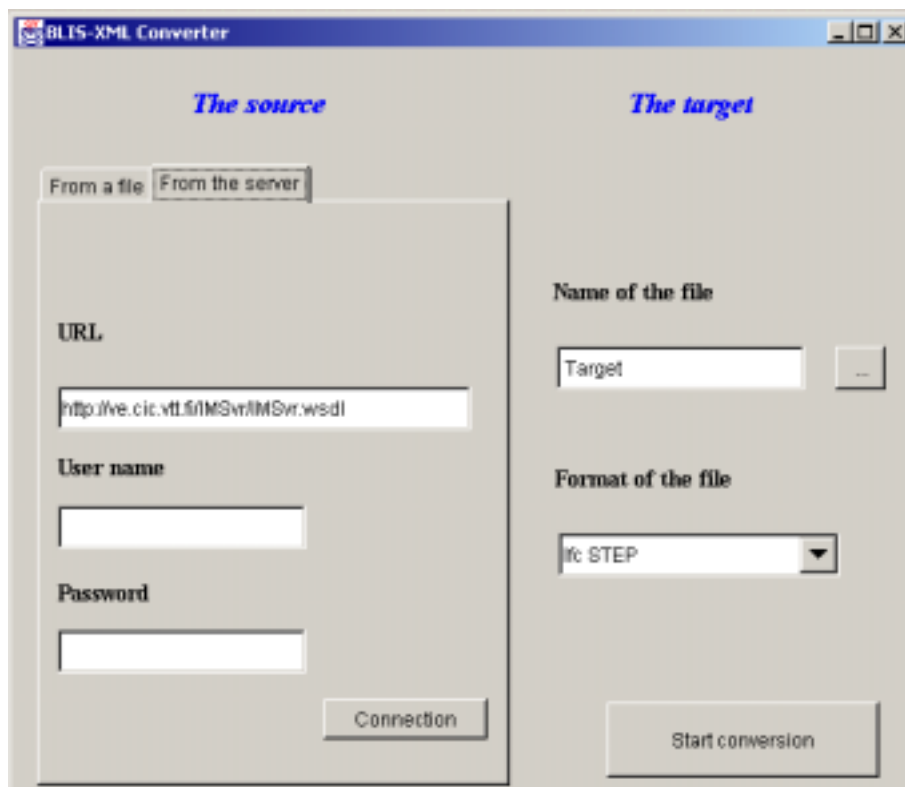


Figure 4.8: My final GUI

We can see figure 4.8 what this programme finally looks like. Its utiliza-

tion is quite easy in the meaning that there is not a lot of options. What is more there is a notice, which shows step by step what we have to do. We can download or consult this notice from <http://cic.vtt.fi/projects/ifcsvr/bliscon/index.html> where we can also download the programme, the library needed, and in a separated part the XSLT files.

# Chapter 5

## Conclusion

The goal of the programme is to implement in Java the four following functions:

- Converting BLIS-XML file into IfcXML file
- Converting BLIS-XML file into STEP P21 file
- Getting BLIS-XML data from the IFC Model Server
- Saving BLIS-XML data into file

I achieved to do that. So now people who want to convert their file can do it. What is more, the Java programmer who want to have access to the IMS can reuse my java classes for their own application.

This work was the opportunity to discover a lot languages : from the Java language, which is interesting for its portability, to the XML and ones of its particular aspects. I have also seen some new tools which handle the XML data like DOM. And it was the occasion to understand what the SOAP protocol is.

Now the programme is finished and downloadable from the web site of the IMS. This brings a new tool for the IAI programmers. I hope that will help the practical application of the IFC softwares.

# Lexicon

**DOM** Document Object Model : an application Programming interface for HTML and XML document.

**GUI** Grafic user interface

**IAI** International Alliance for Interoperability

**IMS** IFC Model Server

**JVM** Java Virtual Machine

**ISO** International Organization of Normalization :The International Organization for Standardization is a worldwide federation of national standards bodies from more than 140 countries, one from each country.

**JVM** Java Virtual Machine : Java interpreting software

**SOAP** Simple Object Access Protocol

**W3C** World Wide Web Consortium : it is a forum for information, commerce, communication, and collective understanding. it develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential.

# Bibliography

- [F. Boulaire (2002)] F. Boulaire (2002) "Extension of the IFC model into bridges"
- [C. Delannoy (2000)] C. Delannoy (2000) "Programmer en Java".
- [M. Birbeck and al. (2001)] M. Birbeck, J. Duckett, O. Gauti Gudmundsson, P. Kobak, E. Lenz, S. Livingstone, D. Marcus, S. Mohr, J. Pinnock, K. Visco, A. Watt, Z. Zaev, N. Ozu (2001) "Professional XML 2nd Edition" chapter 24.
- [K. Ahmed and al. (2001)] K. Ahmed, S. Ancha, A. Cioroianu, J. Cousins, J.M. Crosbie, J. Davies, K. Gabhart, S. Gould, J. Hart, R. Laddad, S. Li, B. Macmillan, D. Rivers-Moore, J. Skubal, K. Watson and S. Williams (2001) "Professional Java XML" chapter 1, 3 and 7.