Classification: Restricted

IST-1999-10303
D602
Virtual Reality Information Front-end

Authors:

Frits TOLMAN	TU-Delft
Reinout van REES	TU-Delft
Joost FLEUREN	TU-Delft
Reza BEHESHTI	TU-Delft

Distribution	*Draft 0*	*Draft 1.3*	*Issued 2*
Date	24-12-2001	8-01-2002	16-01-2002
BETANET	*	*	*
CSTB	*	*	*
EPM	*	*	*
NEM	*	*	*
STABU	*	*	*
TNO	*	*	*
TUD	*	*	*
TW	*	*	*
WWW		*	*
CEC			*

Deliverable Reference		
WP WP6	*Task* T6200	*No.* D602
Status Draft/Issued/Revised Issued		*Rev.* 2
Date 16th January 2002		Release to CEC Jan 02

Document Control Sheet

Revision	*Status*	*Page Nos.*	*Amendment*	*Date*	*By*
0	Draft	Full Doc		24-12-01	TUD
1.0	Draft	Full Doc	List amendments	04-01-02	TW
1.1	Draft	Full Doc	Additions from Reinout	07-01-02	TUD
1.2	Draft	Full Doc	Final Updates	08-01-02	TUD
1.3	Release	Full Doc	Last additions	08-01-02	TUD
1.4	Final	Full Doc	Added examples	15-01-02	TUD
2	Issued	Full Doc.	Issued to the Commission	16-01-2002	TW

Table of Contents

Table of Figures

1 Goal

This deliverable describes the results of Task 6200. The goal of Task 6200 is:

- To develop software to extend the bcXML capabilities with multi media visualisation support for the *component* level,
- Prototype and demonstrate a number of applications of the component visualisation technology, and
- Study the implications of the new Internet and multi-media enhanced bcXML for inter and intra *project* communication for large-scale Building and Construction projects.

Disclaimer

In the initial project proposal the duration of WP6 was one and a half year. Due to the fact that: (i) the project duration has been decreased to two years, and (ii) WP6 work could only start *after* bcXML had been stabilised, the software developments had to be restricted to prototype implementations.

2 Summary

The report describes how bcXML can be enhanced by multi-media extensions like graphics using SVG and X3D (VRML). After a general introduction to the problem, the mechanics of producing SVG and VRML (X3D) output will be explained and the application of the technology for visualising building components in 2D and 3D will be evaluated for three application fields.

The applications implemented are (1) a catalogue publisher, (2) a multimedia *component* information front-end and (3) a multimedia *project* information front-end. The first two applications are dedicated to the simple bcXML based component buying and selling use case. The third application is somewhat futuristic, though very valuable for the European BC industry if running smoothly.

Besides the obvious advantages of 2D and 3D graphics support the applications clearly demonstrate the virtue of the capabilities of bcXML (multiple language, multiple view, decomposition, specialisation).

The last few chapters are dedicated to the extension of the technology to the building project level, i.e. to support electronic communication between different parties involved in the design, engineering, planning and/or execution of (parts of) actual building projects. The purpose is to have a first look into the applicability of the component centred technology for other uses.

One of the problems that has to be looked into is the automatic creation of SVG and VRML output from taxonomy data. Ideally adding new objects to the taxonomy should not require programming of shape information. The solution investigated here is to restrict the shape output to a very abstracted world where all the objects belong to 4 classes: point-like, line-like, face-like and volume-like, plus three relations: consists-of, bounds and locates. Such a simplified world can be generated without too much effort. Question is however how well it serves the purpose.

The second objective was to try to support (at least some) end-user input. Currently browsing a VR-world is strictly read only. If for example the technology is used in co-operative design/engineering users should be able to change types, properties, dimensions and locations of components. And that in a vendor independant way, and not, like the Nemetschek CAS example, in a vendor specific way (which is of course fine for a vendor type application). The research performed here tried to implement vendor neutral mechanisms that support end-user input and involvement.

For the project interaction work two types of problems have been chosen. The first application focuses on the co-operative design of wall with a door, window, light switch and electricity plug. This type of designs always involves people from different disciplines and is rather typical for a broad class of BC design/engineering problems. The second application focuses on inter and intra project communications (including the local authorities and the public) during the execution of a building project.

The last chapter contains the overall conclusions and gives some recommendations for further work.

3 Introduction

Virtual Reality (or more general multi-media) information front-ends display 2D and/or 3D images of physical objects or 'worlds' on a computer screen and support the users to navigate trough the VR world and to look at the images from inside and outside. A Virtual Reality information front-end is a piece of software that helps the user create and retrieve information about the physical object displayed on the screen.

VR images or 'worlds' can be created, navigated and interrogated using different technologies. One approach to VR creation is to use existing commercial VR tools. This approach undoubtedly produces superb quality images, but, at the down side, only for those users that have access to the tool. Another approach, with the same handicap, is to generate VR output from existing geometric modellers or CAD-systems. As discussed in D601 it is also possible to create VRML, or Java3D VR output using existing browser plug-in functionality. This is the approach implemented here. The first half of the report describes how to build VR *component* information front-ends for buying and selling components (materials, elements) like Hollow Concrete Floor Beams, Inner Doors, and such. The second half describes the current state of the work on VR *project* information front-ends.

A VR project information front-end shows an image of the facility to-be (called project worlds or project webs) that can be used to find information about certain hidden facility-related details. Users can enter these project worlds over the Internet, navigate to the area of their interest, click on objects of their choice and find whatever information they are looking for.

In the future construction projects will be represented in such a way that all parties involved – including the authorities and the general public – can navigate the virtual project world and find and provide the information of their interests. As bcXML is clearly a superb carrier of *meaning* the objective of WP6 is to investigate the various possibilities for extending the bcXML capabilities with multi-media enhancements.

The XML-based Communication Technology under development by eConstruct (bcXML) mainly focuses on *semantics*. It describes the words and language constructs required for electronic meaningful multi-lingual information exchange in the Building and Construction industry (BC). The bcXML communication language can be used in communications between (electronically connected) humans, between humans and computer applications, or between computer applications themselves.

The nice thing about XML is that it supports the distinction between *content* and *mark-up*. Content is what you want to communicate and mark-up is how you want to present the content on paper or on the screen. For eConstruct this XML feature is interesting because in multi-lingual communication the mechanism can be used to display the content of a message concurrently in different (European) languages. For this WP the interesting thing is that mark-ups are not limited to text, but may also be formed by graphical images, or speech generated output.

Though bcXML and its supporting taxonomies will enormously help the Building and Construction industry in it's effort to change its traditional paper based way of communicating into an electronic way of communicating, in many human communications '*a figure says more then a thousand words*'.

The objective of WP6 is therefore to investigate how bcXML communication involving humans can be enhanced by multi-media support, especially by 2D vector graphics and 3D Virtual Reality (VR).

To fully understand the aim and the reason for incorporating the work package in a European project, consider an average one-of-a-kind construction project in the public sector, involving participants, suppliers, authorities, and the general public. In some cases these projects involve thousands of people, all with daily requests for information. Then imagine a project where people from different nationalities are co-operating to reach the required result, and notice yet another source of miscommunication and consequently of cost of failure.

What-if all parties involved can have access to a virtual world describing the project in some detail? 'Some' detail, not great detail, because the VR world is used only to support information retrieval in general, not to retrieve geometrical or topological details. For example, somebody living in the neighbourhood of a project can enter this project world from his PC at home and look around. He can move to his home address and view the project from, say, his second floor window. He can scroll forward in time to see when he should plan his holidays because of the piling noise. Think about the many authorities that need up-to-date information about all kinds of aspects, like fire safety, durability, energy, compliancy with building regulations. With so many players and so much information floating around (including out dated or simply wrong information) it is no wonder that currently miscommunication and cost of failure plays such an important role[1] in the BC industry.

Improving electronic communication at large is the ultimate goal of bcXML and the work on SVG and VR information front-ends. However this goal is still far away. Before we can start to work on VR front-ends for large-scale projects we first have to understand the technology and implement it for eCommerce. In eConstruct we are developing bcXML that provides us with the right level of semantics. Suddenly we can talk about Wall Elements, or Hollow Concrete Floor Slabs and the Internet 'knows' what we are talking about. How interesting it would be if this promising communication technology can be further enhanced by multi media.

In order to study the question how this can be achieved and which results might be expected, we are looking into the technology and into the short-term needs in eCommerce support. We study how to use bcXML to generate 2D drawings and VR project worlds, and how the user can navigate through these worlds and find information that he needs, but only on a small scale.

As a proof of concept WP6 aims to develop and demonstrate the technology for bcXML supported Publishing and bcXML supported selling and buying of single components.

WP6 is also looking into the possibility to apply the technology developed for components related communication for VR project information front-ends. If the extension to the project level can be straight forward eConstruct might actually be able to produce a first step in the right direction.

[1] Cost of failure is about 5 – 10% of each project.

4 Creating an Image

This section demonstrates the transformation from bcXML to SVG/X3D. This transformation, starting with the LexiCon content as described in D501, produces SVG and VRML from bcXML using XSLT.

Following an example of how bcXML information is contained within the data layer:

```
<HollowTimberFlushDoorWithGlazing name="1">
           <availability>
                      <SingleValue>Y</SingleValue>
           </availability>
           <catalogueCode>
                      <SingleValue>04041208</SingleValue>
           </catalogueCode>
           <description>
                      <SingleValue>FD30 Firedoor - G02</SingleValue>
           </description>
           <productLine>
           ...
```

Information contained in this data layer will all describe *properties* of the specific product, i.e. availability, material type, length, etc.. For referencing what each value *represents*, users are directed to the *taxonomy* where this is described. The taxonomy defines for example about the property “material”: which part of the object is meant, or about the property “length”: which length (inner or outer) of the object is meant. The taxonomy level should also contain the multimedia instructions, just like it contains the multilingual information.

Regarding the multimedia visualization information we are talking about *data* and about *logic*. For example: the color used to fill a square should correspond to *blue* for a certain property but yellow for another property, and the length should be measured in mm or inch depending on the super-type. For this reason, the multimedia formatting data is in XSLT format, which is suitable for containing this kind of logic.

Let’s take a look to a piece of XSLT code from a Multimedia Formatting file:

```
<xsl:for-each select="catalogue/door[@name = $objectNumber]">
      <xsl:variable name="image-width" select="'500'"/>
      <xsl:variable name="image-height" select="'400'"/>
      <xsl:variable name="door-height" select="doorLeafHeight/text()"/>
      <xsl:variablename="door-thickness"
select="doorLeafThickness/text()"/>
      <xsl:variable name="door-width" select="doorLeafWidth/text()"/>
      <svg>
           <xsl:attribute name="width">
                      <xsl:value-of select="$image-width"/>
           </xsl:attribute>
           <xsl:attribute name="height">
                      <xsl:value-of select="$image-height"/>
           </xsl:attribute>
           ...
           ...
```

The bcXML information as data, and the Multimedia Formatting file as the describing logic about the visualization process, serve as input for the XSLT processor which processes these two to the end-result. This XSLT processor can be in the shape of one of the various available open source command line or server sided pieces of software. This could be schematically drawn like:

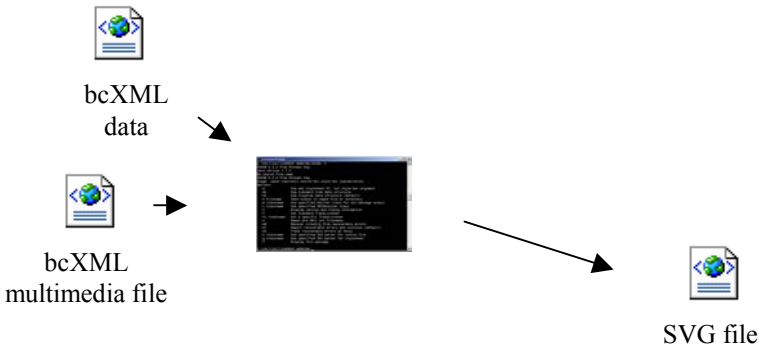

Figure 1 - bcXML visualization process (SVG)

Let's take a look at a sample of such result code:

```
<svg width="500" height="400">
  <script type="text/ecmascript">
    function filter_object(evt)
          {
                    [...]
          }
  </script>
  <g onmouseover="filter_object(evt)" onmouseout="filter_object(evt)"
style="stroke:black; stroke-width:1; fill:tan"
transform="translate(140.944444444444,270)">
    <path d="M 0 0 L  0 -220.111111111111 L 93.1111111111111 -
220.111111111111 L 93.1111111111111 0 L 100.888888888889 0 L
100.888888888889 -227.888888888889 L -7.77777777777778 -227.888888888889 L
-7.77777777777778 0 Z"/>
  </g>
   ...
```

The whole of this snip of code will result in the following when we render the piece to a picture with an SVG visualizer like the Adobe SVG plug-in running under Internet Explorer:

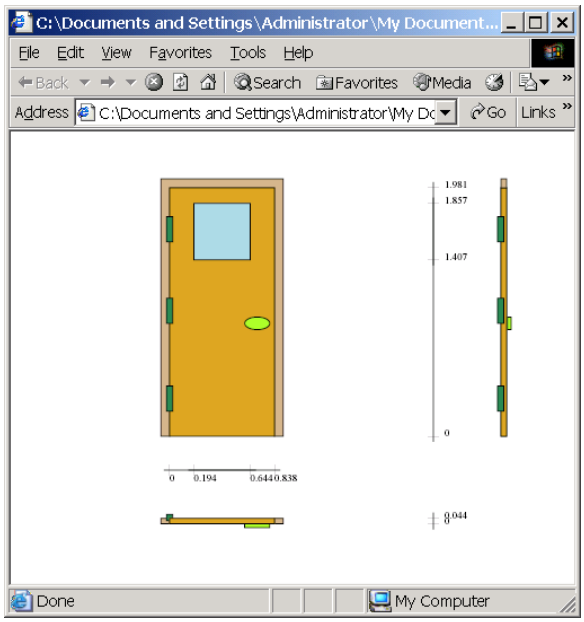

Figure 2 - Example SVG output

5 Buying and Selling Components

As a test environment for the component visualisation technology the following two application areas have been addressed: *information publishing* and *accessibility of information*.

- Focusing on information publishing will result in research on how to improve the way information suppliers should publish their information.
- Focusing on accessibility of information will result in research on how to improve accessibility of information published.

The results of these questions are worked out in the following chapters “catalogue publisher” and “multimedia front-end”.

6 Catalogue Publishing Support

6.1 Introduction

The focus of this application project was partly set on researching ways to improve the publishing (paper and electronically) of information. This results in a conceptual design and prototype to empower publishing of information. This prototype is called “the static catalogue builder”.

It will consist of more than just printing out the bcXML data in a readable form. It is desirable to be able to express the catalogue data accessible and in a pre-defined form, like for instance with one or more indexes based on different properties.

If we create a PDF catalogue for printing purposes an index in front of the catalogue and several lookup tables in the back would be desirable. Lookup tables that direct to the different catalogue pages of the different products, sorted on a different property in each index. So for instance an index with the page numbers of where doors with a specific fire resistance can be found could be created automatically.

But if we want to generate a HTML based catalogue, for online purposes or for on CD-Rom, we could benefit more of pre-processed index files. On the moment of rendering the catalogue, all possible indexes of the different properties could be generated, but hidden and accessible through indexes of the different indexes. So users won’t get lost in the different pre-generated search paths, because they click themselves a way thorough the information. This way we would already obtain some kind of interactive accessible catalogue by just statically generated HTML files. Just with bcXML data as basis no extra instructions are needed, no taxonomy is needed.

Besides style sheets that output PDF or HTML formatted catalogues, just as easily WML for wap-phones, or mark-up suitable for palmtops could be generated. This static catalogue creating could of course also integrate 2D SVG graphics and 3D VRML graphics, rendered per object.

Another welcome ability to the static catalogue builder would be to also apply lingual and context sensitive rendering of the bcXML data. Requirement for this is a corresponding taxonomy with the translation data and context data. This could be made by the end-user himself, but of course more easily obtained from for instance a branch organisation.

This multilingual capability would then enable you to not only generate an up to date paper catalogue on the fly for your own Dutch clients, but with one press on the button you would have exactly the same catalogue in Norwegian, Greek, Hungarian or Japanese. Imagine how in this way you are able to extend the scope of your information delivery and retrieval within our divided Europe and further.

With context sensible information rendering a catalogue can be generated on the fly with the information for that specific category or information seeker be presented first, while putting less interesting information for that case on the back. This way a constructor doesn’t need to find his way through irrelevant information like the colour of the object. And the safety engineer can browse through information being presented with what he is looking for: safety related information like fire-resistance.

6.2 Prototype

Firstly a prototype was build that could publish bcXML related data without using a taxonomy. Secondly a prototype was build which supported bcXML related information *with* a taxonomy. The overall architecture of this taxonomy related prototype can be represented following:

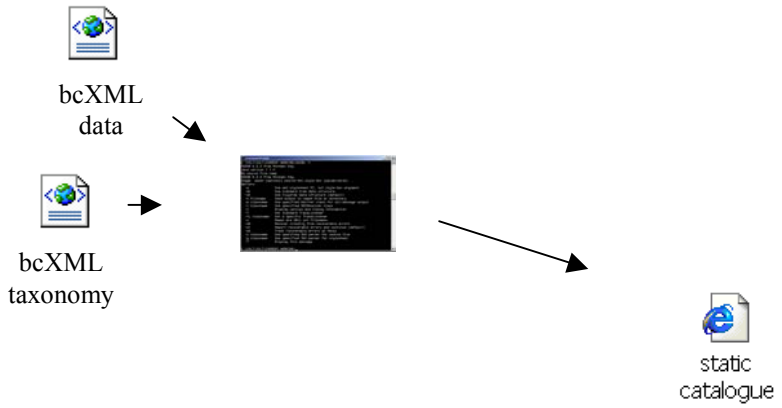

Figure 3 - bcXML data publishing

As input we have a bcXML data file and the bcXML taxonomy file. After processing (using command line XSLT processor) the result is a HTML catalogue file according to this data.

At first this catalogue is able to publish XML data with creating indexes to browse through the data. A taxonomy is not needed for this. The only requirement is that the data is structured in a specific way, so that the application knows the difference between for example sub elements and properties.

Secondly the prototype is able to output multilingual catalogues. In this case taxonomy information is needed. Based on the taxonomy structure, which holds the multi lingual data the output can be expanded. This will mean that an extra output *dimension* is added. Without the multilingual capabilities the catalogue builder outputs a structure consisting of data and indexes, which could be represented by the following drawing:

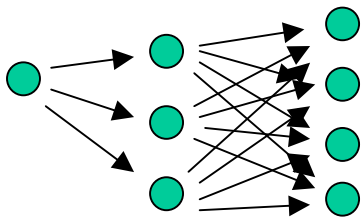

Figure 4 - Data structure without independent axe

With the multilingual extension, the data structure will be enriched with an extra dimension, the lingual one. This means that at all times you can change the value of this dimension, thus switching to a different language. This will bring us to the following representation for the same data structure with the lingual axis:

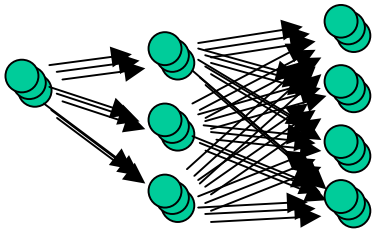

Figure 5 - independent context axis

Let's take a look to the screenshots of the application to view the generated catalogue.
The HTML containing a generated catalogue is attached to this report under the name "static catalogue.html"

In Figure 6 you can find a HTML representation of a building element, in this case a door.

Let us examine the figure and note two things: the language choice in the bottom, and the ID in the URL line of the browser, pointed out in Figure 7. Choosing another language using the bottom links will result in directing to the same object ID, but within a different dimension. If we press the Dutch link we will be linked to the part of the page with the same ID but with Dutch dimension indicated by the letters 'en' instead of 'nl' in the beginning of the ID. Figure 8 shows the auto generated Dutch version of the building component. Note that also the value of the properties are translated (see properties rotation and material).

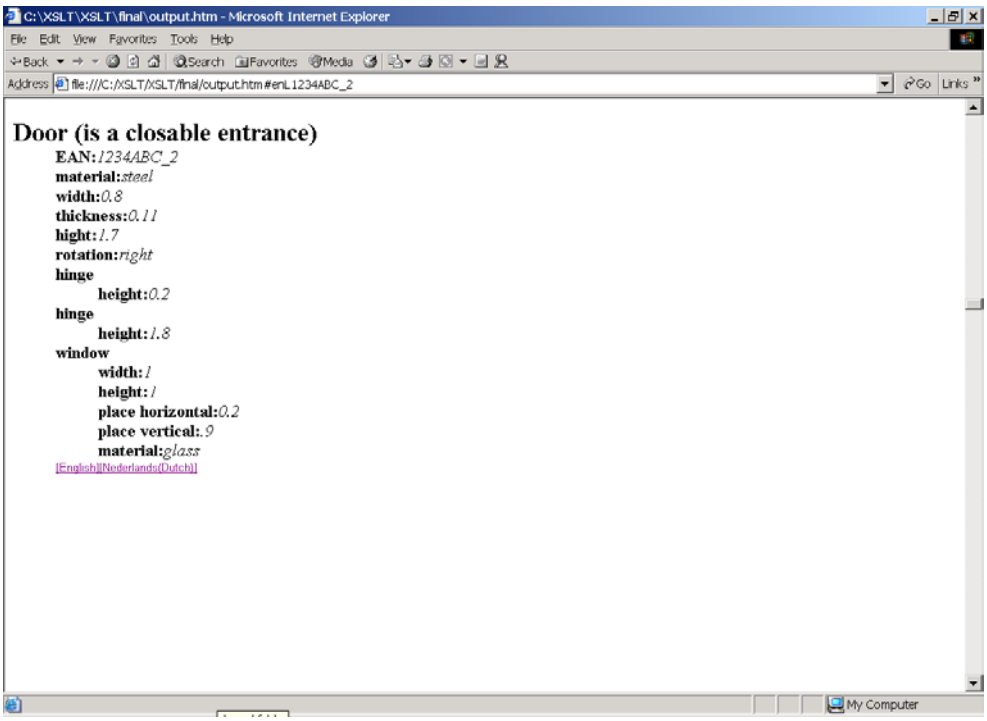

Figure 6 - Building element rendered to HTML

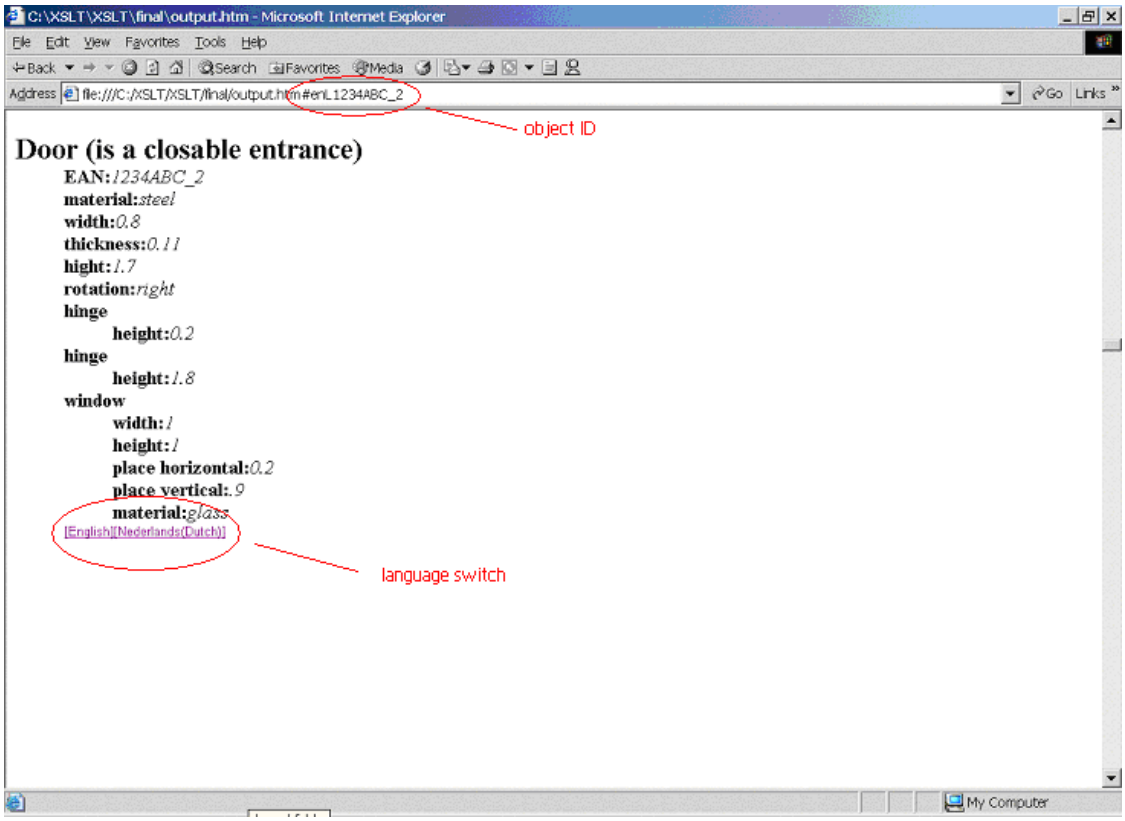

Figure 7 - Lingual switching point

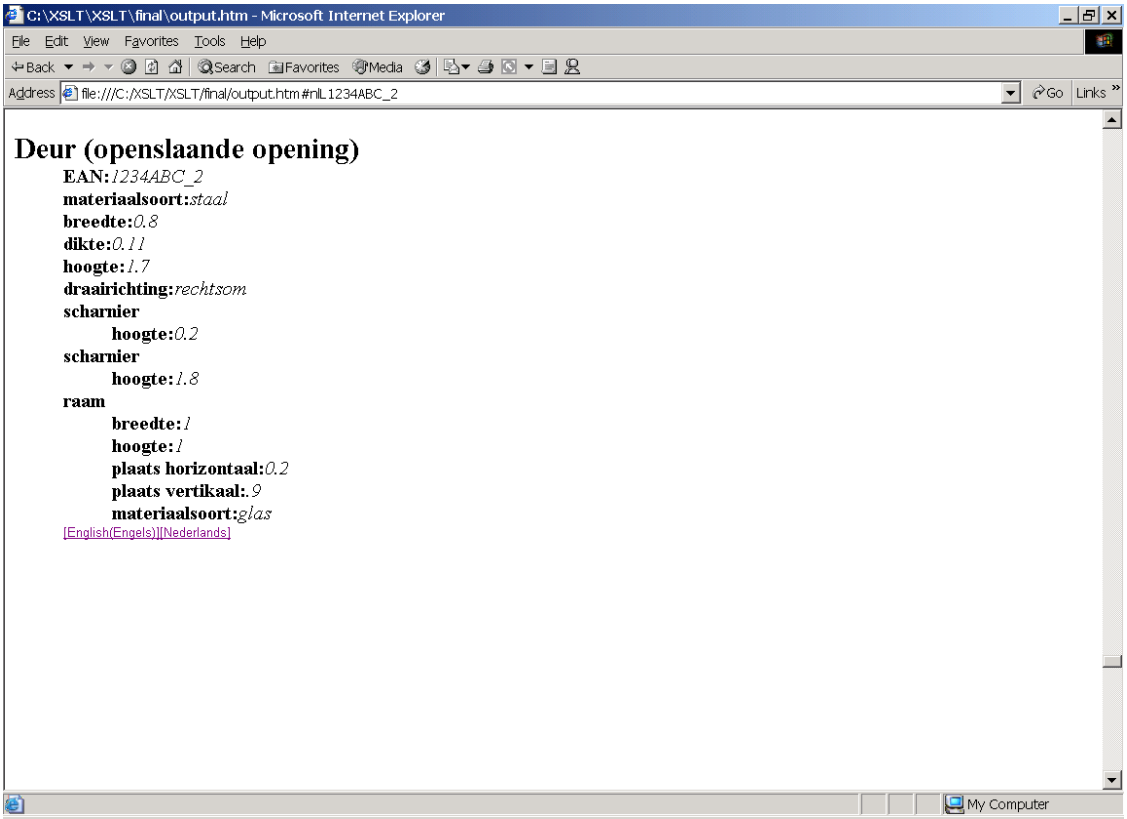

Figure 8 - Dutch version of building element

To make the information published accessible the catalogue builder automatically generates indexes based on each property. This is done without using the taxonomy information. In Figure 9 ou can find a selection index on the property ‘material door leaf’. Clicking the link ‘Door-material: steel’ will result going to the door shown in the previous figures.

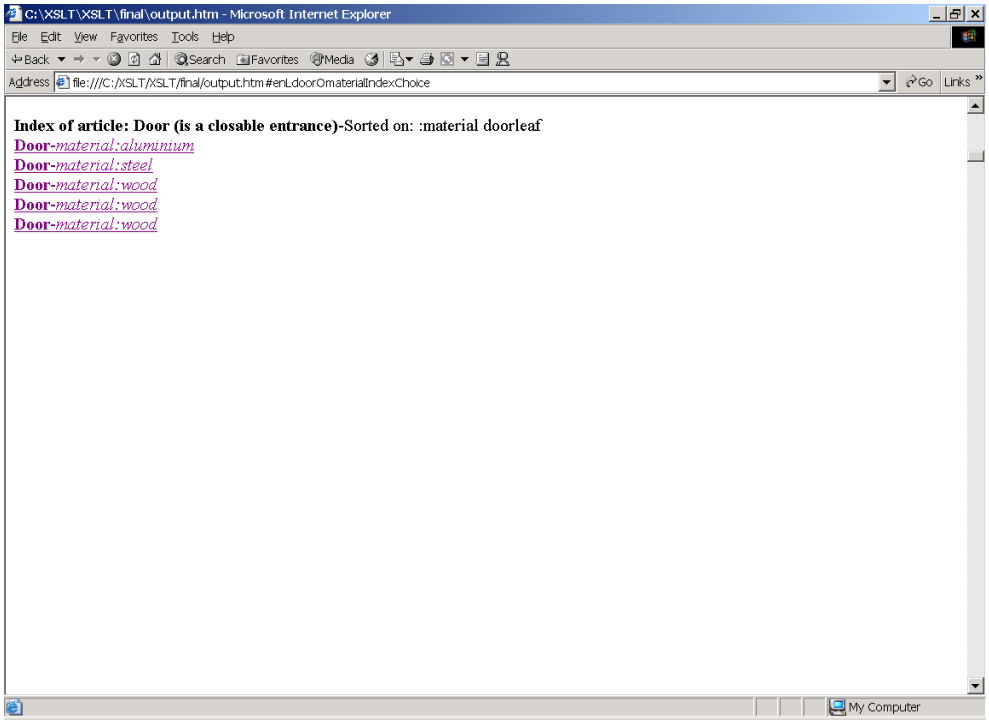

Figure 9 - Material selection

Presenting all of the indexes to the user would result in some kind of chaos: the user would get presented just as much indexes as there are properties within the catalogue. Therefore an index of the indexes is generated to enable the user to select on which properties he wants to make his selection. See Figure 10 for this page.

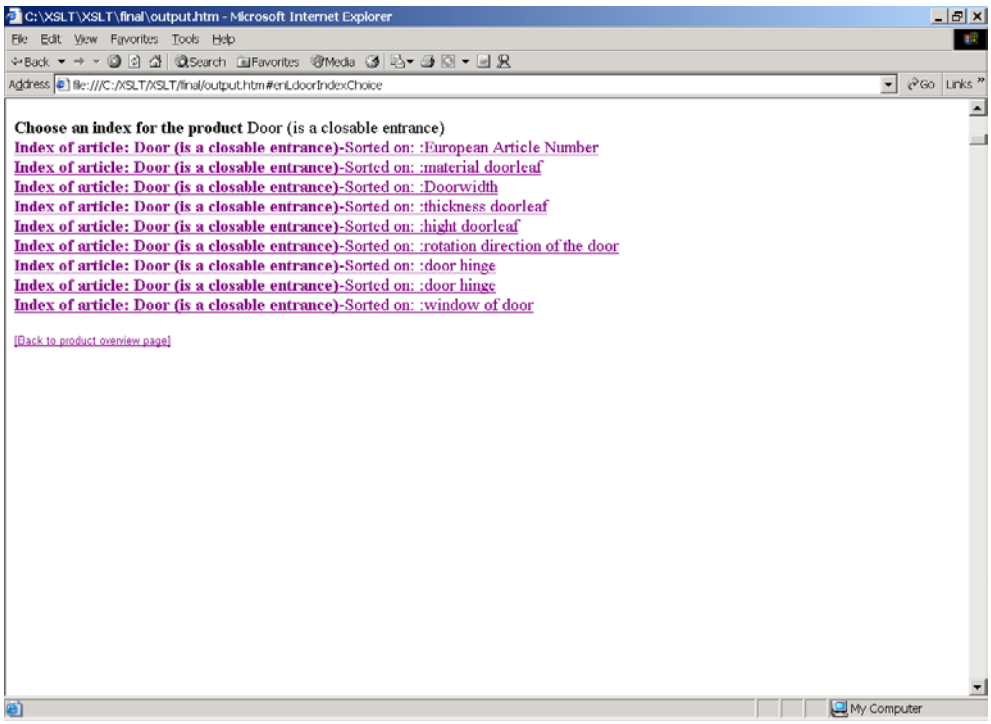

Figure 10 - Selection of indexes

Further also a page at which you can select the object you want to search in is generated, as also a general starting page with language choice. See Figure 11 and Figure 12 for these pages.

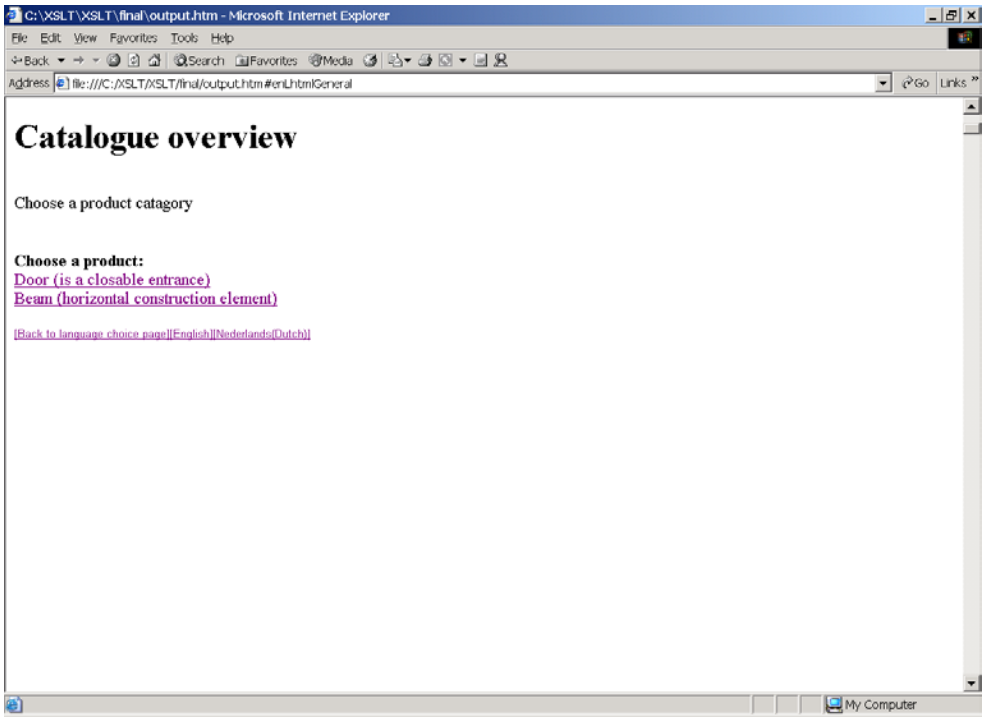

Figure 11 - selection of object

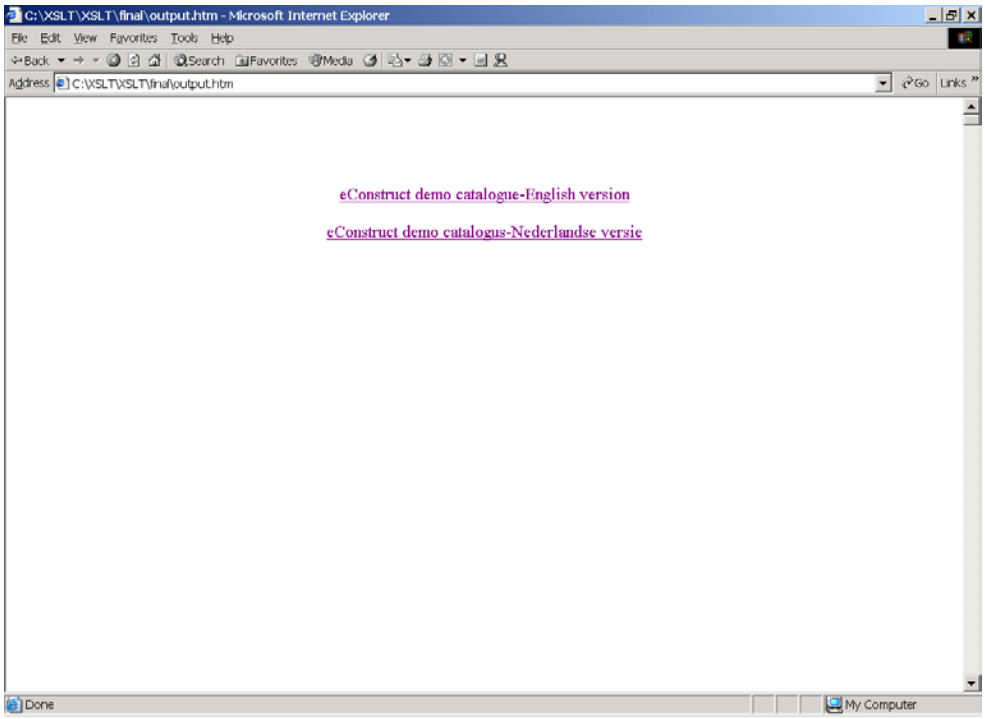

Figure 12 - starting page with language choice

6.3 Conclusions

The static catalogue builder application proves that it is possible to *automate* data publishing on the basis of bcXML. With taxonomy-data available it also proves the bcXML enabled multi-lingual publishing capabilities; which has been done by generating a HTML file with internal links forming indexes and lingual switches.

This prototype core for bcXML related publishing could be extended to output paper based catalogues, catalogues for palm-top clients, for cell-phones. And for on-line publishing, with not only indexes to browse though the information, but also for being able to specify queries on different properties.

7 Multi-media Publishing Support

7.1 Introduction

The multimedia front-end part focuses on displaying bcXML information. Here it is the objective to improve accessibility of this bcXML information, which will be done by investigating the possibility to access information in different multimedia formats and other methods.

The different views that enable users to access the information are broader than just different media (text, 2D, 3D, speech). But also the concept of accessing information from a different viewpoint will be covered. This will mean that the information could be displayed from different contexts. These contexts will influence which data, and in which form the data will be visualised. In the demo two *independent* context axes will be covered, knowing a language context and a profession context. Together with the output format context this will mean that three independent choices can be made regarding rendering the information for the user – media choice, language choice and viewpoint context choice.

The first choice is obvious: it will determine whether the output will be textual, graphical 2D, graphical 3D or a combination of these.

The second is the language context. Depending on the language choice the information will be rendered in that specific language and character set.

The third could be the *viewpoint* context. For example in the profession context of a painter it could display information regarding square meter surface and surface material by printing the surface amount and sort material in the figure, while indicating the target colour by the colour in the figure. In the case of a site-manager context, a different kind of information would be important and displayed. Like what are the delivery conditions of a specific object, and how much does it cost. In this case it could be interesting to express the delivery time of an object by a colour range. So that in a glance the delivery condition can be seen while browsing through a set of objects.

Viewpoint context information formatting is not only limited to figures, also regarding textual views this way only that information important to the viewer can presented first. When an object has a wide range of different properties and sub-objects, textual information not of interest of the viewer can be hidden or put at the end.

The challenge is *not* to define a technology that handles the different output formats for one specific object. There is nothing new to having a catalogue with texts in Dutch, English, Greek, Norwegian and 10 more languages. Supported by 2D and 3D figures and in different versions for different information seekers. To be presented on paper, CD, internet, Personal Digital Assistant (Palmtops, etc.), mobile phones. It can all be done if we just put the time and specialists (in mark-up-formats and natural languages) on that specific piece of information.

The added value would be if we could generate these different output-media on the fly. That we just put in the bcXML information and on demand let it generate the Greek, Dutch or English version just on our request. And choose to look to a 2D or 3D output if we want, and with the view context we desire.

That this core could provide a Greek door supplier to just enter his product data in the system in the bcXML format in his native language and character set. And then on the fly generate those catalogues for its new potential clients in France and Spain. Imagine how this technology could benefit our communication that is bcXML compliant.

7.2 Multimedia Visualiser Prototype

One online multimedia information front-end for bcXML information has yet been delivered. It is developed in the form of a server-sided application that renders the bcXML information of a door on the fly – according to the three independent contexts.

Let us first examine our door in textual form in English in Figure 13.

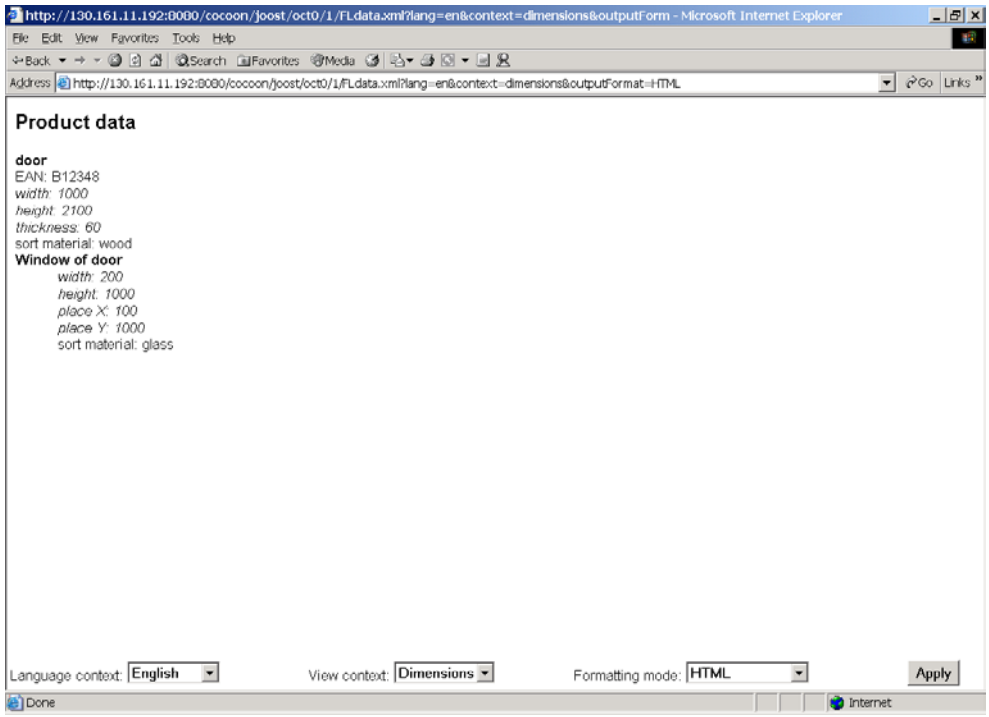

Figure 13 - product data HTML

Changing the language context to Greek will now also provide access to this information by someone speaking Greek. For a lot of others probably not. In Figure 14 you can find this rendering to Greek.

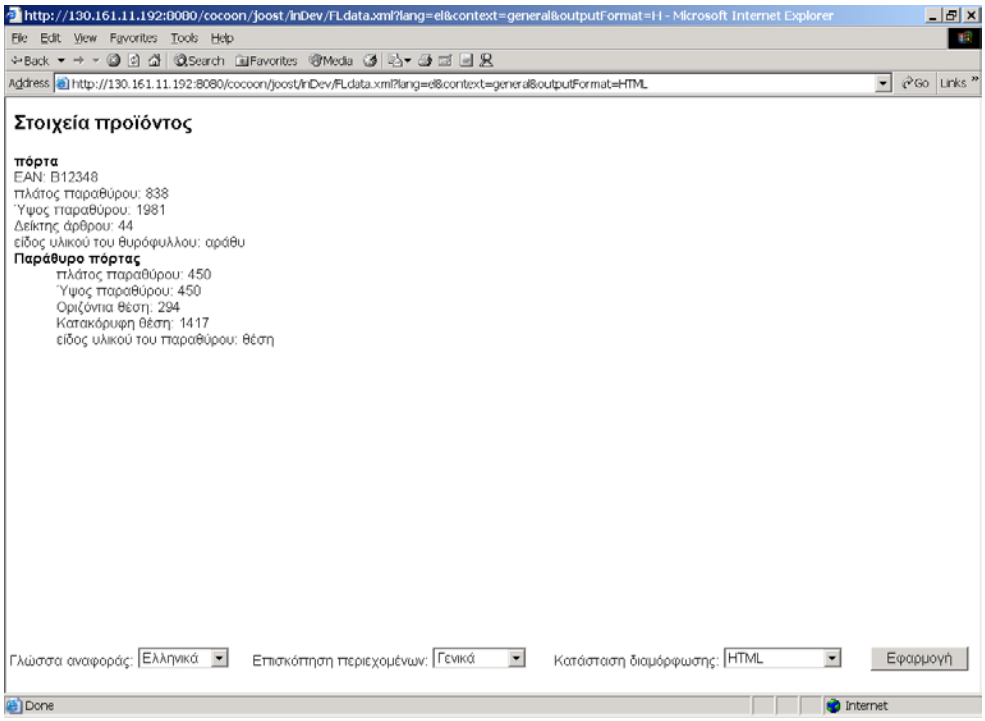

Figure 14 - product data in Greek

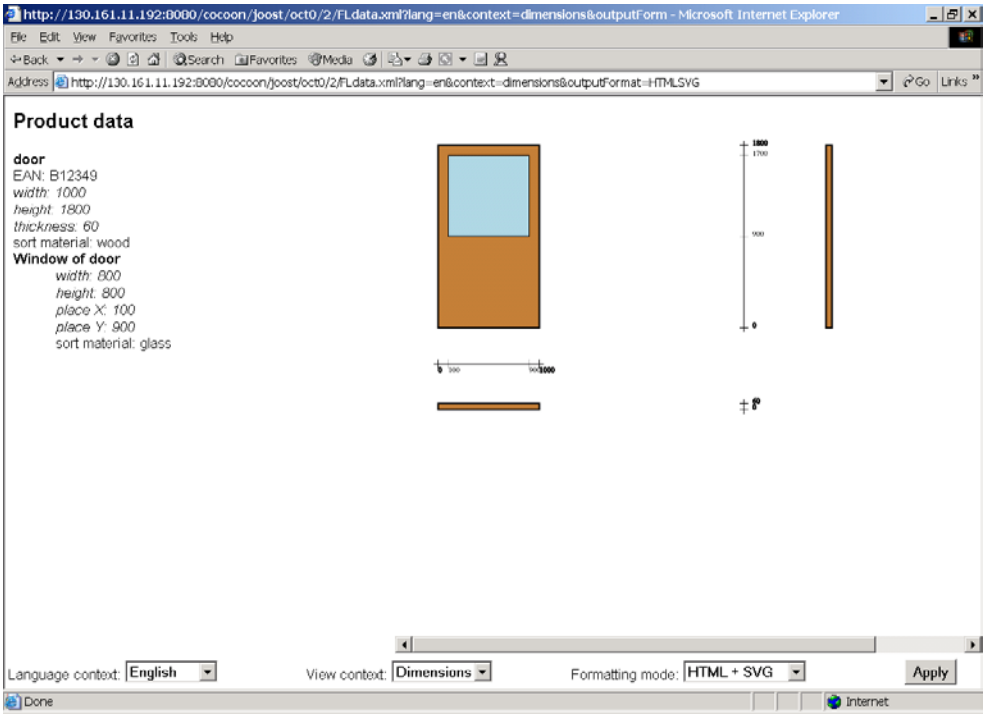

Figure 15 - product data with 2D figure

In Figure 15 you can see 2D drawing support for a specific door. In Figure 16 you will find the same view, but than with different data. Note how you interpret in a glance that you are dealing with a different door, and what those differences are.

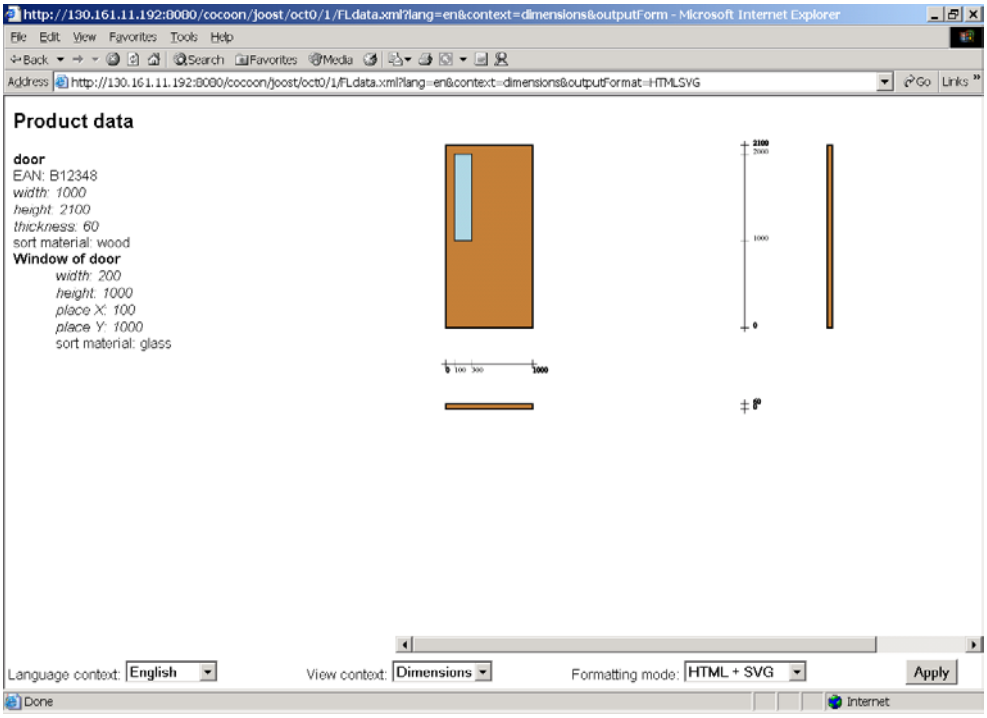

Figure 16 - product data with 2D Figure

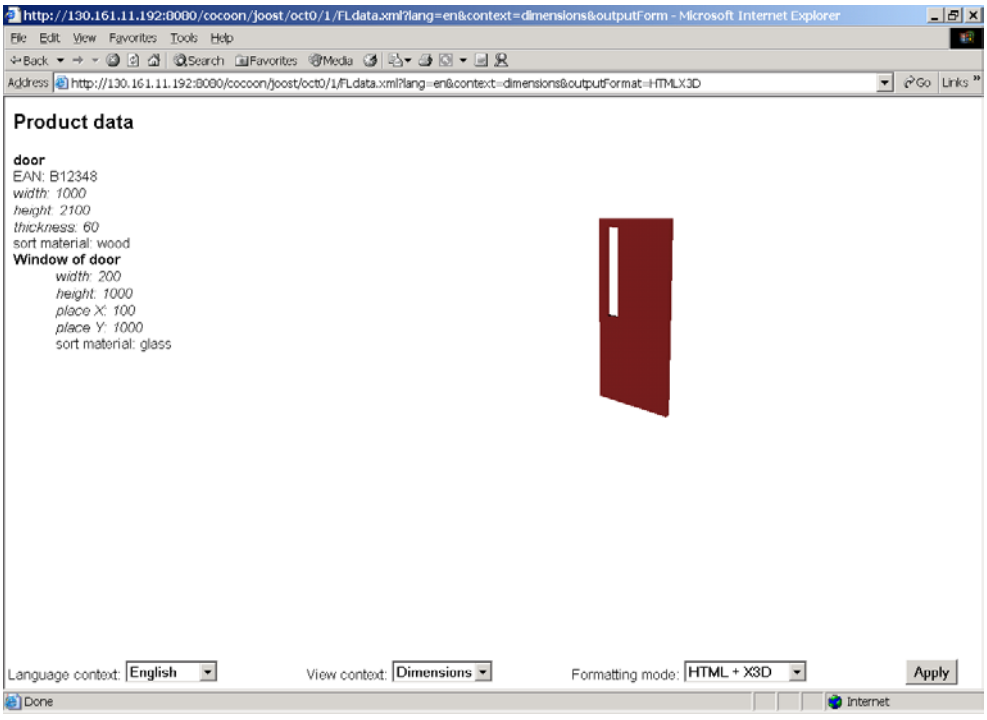

Figure 17 - product data with 3D Figure

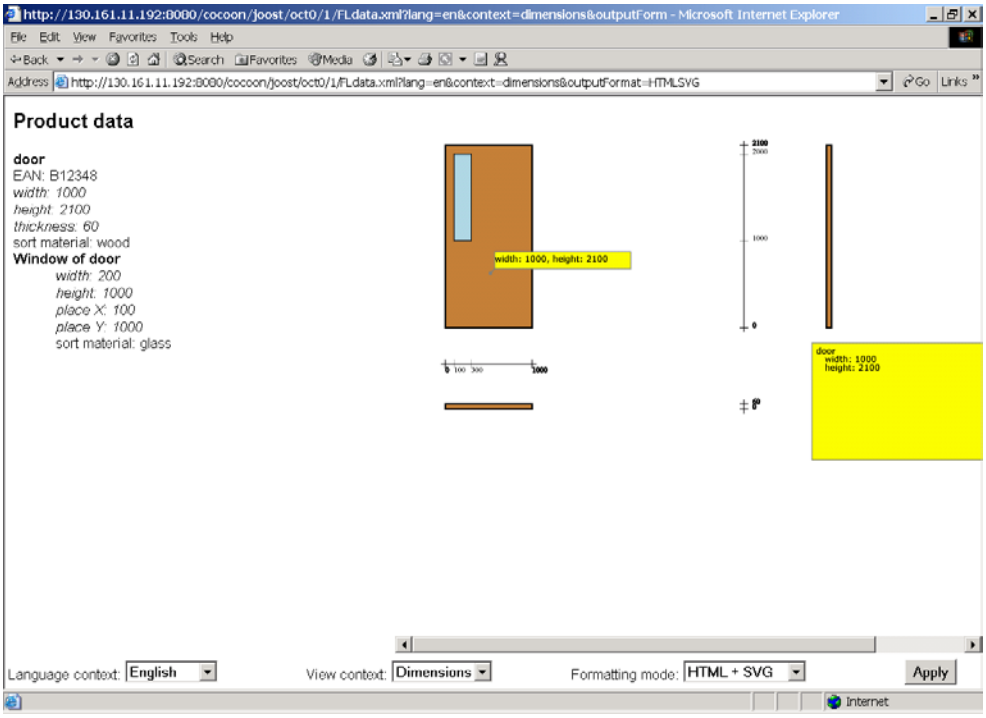

Figure 18 - information hiding in context 'dimensions'

In Figure 18 and Figure 19 you can see the result of changing contexts regarding information retrieval. Based on the context in which the information is rendered, different information can be requested by clicking on a specific part of the object, in this example the door leaf.

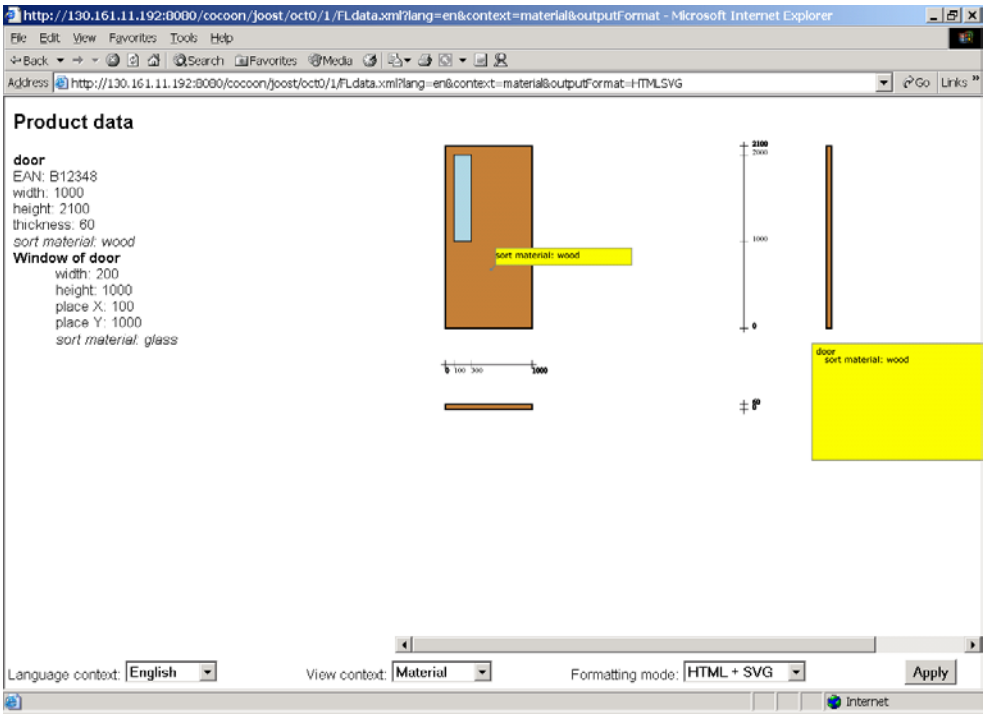

Figure 19 - information hiding in context 'material'

7.3 Visualiser built into the bcXML reference architecture

7.3.1 Introduction

The reference architecture of eConstruct (eConstruct document D201) has at it's core a taxonomy server (dealing with the definitions of objects and properties and their multi-lingual information) and a supplier catalogue server (storing catalogues and allowing searches in them). Those two are augmented by a visualisation module, displaying the information in them.
The idea is to keep every part of functionality isolated, thereby building several inter-operable parts that do only one thing, but do it very well. For instance, the catalogue server searches for information in it's catalogues and hands the resulting data to the visualiser module that only transforms the data to a human-readable format.
At the moment of writing, only html output is generated, but it is expected that some of the svg and x3d visualisation capabilities from the earlier proof-of-concept prototype will be merged into the visualiser by the end of January.

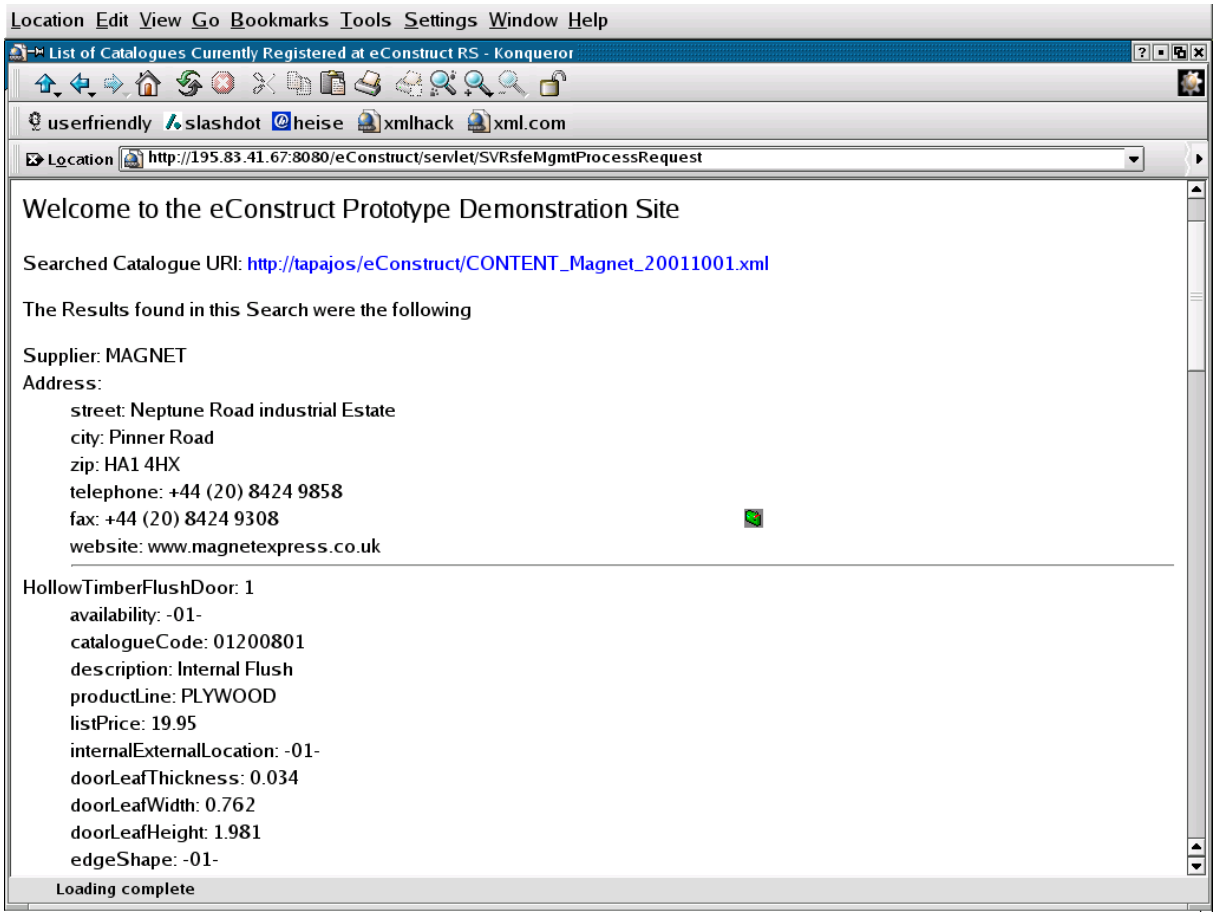

Figure 20 - bcXML catalogue visualisation example

7.3.2 Technical implementation

The html visualisation of bcXML data is basically only a mapping from the bcXML tag (like `Door`) to the word *door*, *deur* or *porta*, depending on the desired language. This information is contained in the taxonomy
While it is possible to write a program that contacts the taxonomyserver and downloads the desired information in order to display the correct translations, this turned out to be a lot of work (apart from the time it takes to complete the action). Therefore the decision was made to do all this work beforehand. Namely to generate an XSLT stylesheet (a file containing instructions on how to modify an XML document) which adds the information contained in the taxonomy to a given bcXML input file. So an incoming

```
<Door>
  <height m="2.40"/>
</Door>
```

would be transformed into

```
<Door>
  <html:span>door</html:span>
  <height m="2.40">
    <html:span>height = 2.40 m</html:span>
  </height>
</Door>
```

This results in one single XSLT stylesheet, capable of formatting any given bcXML document. And without contacting external taxonomyservers, it works stand-alone, which helps integration into other programs.
A simple stylesheet can be used to filter out all html information, adding a customised header and footer in the process, to create the html page.

7.3.3 Future maintenance and adaption

The eConstruct project decided to publish some of the results as open source software to better facilitate use and re-use of the project's results. This was also prompted by the increased attention in "Brussels" for open source software and it's advantages. The visualiser part of the reference architecture is one of the parts that's distributed as open source.
The source code can be obtained using CVS from http://sourceforge.net/projects/bcxml.
Automated tests have been written to ensure the correct conversion. This makes it more feasible to adapt the software, as adaptations are automatically tested for correctness. The tests are available with the source code.

7.4 Conclusions

The multimedia visualizer proves to be able to render bcXML information according to different output formats – 2D, 3D and textual.

It also demonstrates textual translations, just like the static catalogue builder. With as extra that it also applies native character sets, in the case of the Greek translation.

In the context viewpoints it illustrates that the context determines which information can be retrieved regarding the different viewpoints.

8 From Components to Projects

8.1 Introduction

The next goal of WP6 is to study the possibility to apply the bcXML + component related multi-media technology on the Building Construction project level.

A Project Information Front-end (PIF) can be used to support electronic meaningful communication between all parties (humans and applications) involved in a project. Typically a VR PIF is communicating with a project database that holds a detailed description of the project. In eConstruct such a project database can for example be supported by the EDM system. If that approach is followed the IFC-browser is readily available and a VR Project Information Front-end has been delivered.

However, the approach used here is not that simple. Though the advantages of the IFC-based approach are clear (PDT-interface, shape info) there are also some disadvantages. One disadvantage is that the IFC world only contains a limited number of objects mainly focusing on buildings. Civil engineering projects like high-speed railroad projects, or harbour projects are largely out of scope. Other problems are: (1) that large-scale BC projects involve many parties and consequently not every project will use the IFC-approach, (2) many parties like Electrical and Mechanical work in different industries and consequently are unable to follow Building standards and (3) not everybody will use the IFC browser.

In the approach of WP6 we will look into the possibility to use bcXML and the standard browser functionality of the Internet Explorer and Netscape Navigator for presenting multi-media extensions developed for the VR Component Info Front-end without IFCs.

The idea is that such an interface will probably have some restrictions not posed by the IFC-approach, but with added values like for example: universal applicability, and unlimited availability, which are strong user requirements for intra project communication (see next section).

To fully understand the aim and the reason for incorporating the work package in a European project, consider an average one-of-a-kind construction project in the public sector, involving participants, suppliers, authorities, and the general public. In some cases thousands of people, all with daily or even hourly requests for information. Then imagine a project where people from different nationalities are co-operating to reach the required result, and notice yet another source of miscommunication and consequently of cost of failure.

What-if all parties involved can have access to a virtual world describing the project in detail? Some detail though, not great detail, because the VR project world is used to support general information retrieval not detailed shape. Real CAD data is IFC's terrain!

8.2 Project Information Front-ends

Project Information Front-ends (or Project Webs) can come into different flavours. Ideally a PIF combines an image of the surrounding of a project with an image of the as-is and to-be states of the facility under construction.

Figure 21 shows a solution for a PIF where the furnished facility to be (a road construction) is mapped on a picture of the surrounding, both taken from the same point of view.

The advantage of this technology is the photo-realistic image of the surrounding and the road furnishing. The disadvantage is of course the time it takes to perform the required

calculations, the lack of interactivity - point and click is not available - and the fact that only a few discrete images can be provided.

Figure 21 - photo-realistic VR image of a road project

Though the image quality of this approach is superb, it is not what Task 6200 is about. The goal of Task 6200 is to produce simplified VR project worlds based on bcXML that can be viewed and navigated from every PC and interrogated by a simple point and click GUI build with standard VR browser plug-in technology. Both the surrounding and the furnishing aspects are out of scope, though 2D and 3D Geographical Information Systems (GIS) and furnishing objects can play a role in the future, as shown in the following figure of a concept design visualised in it's surroundings (part of a Dutch city, taken from GIS data):

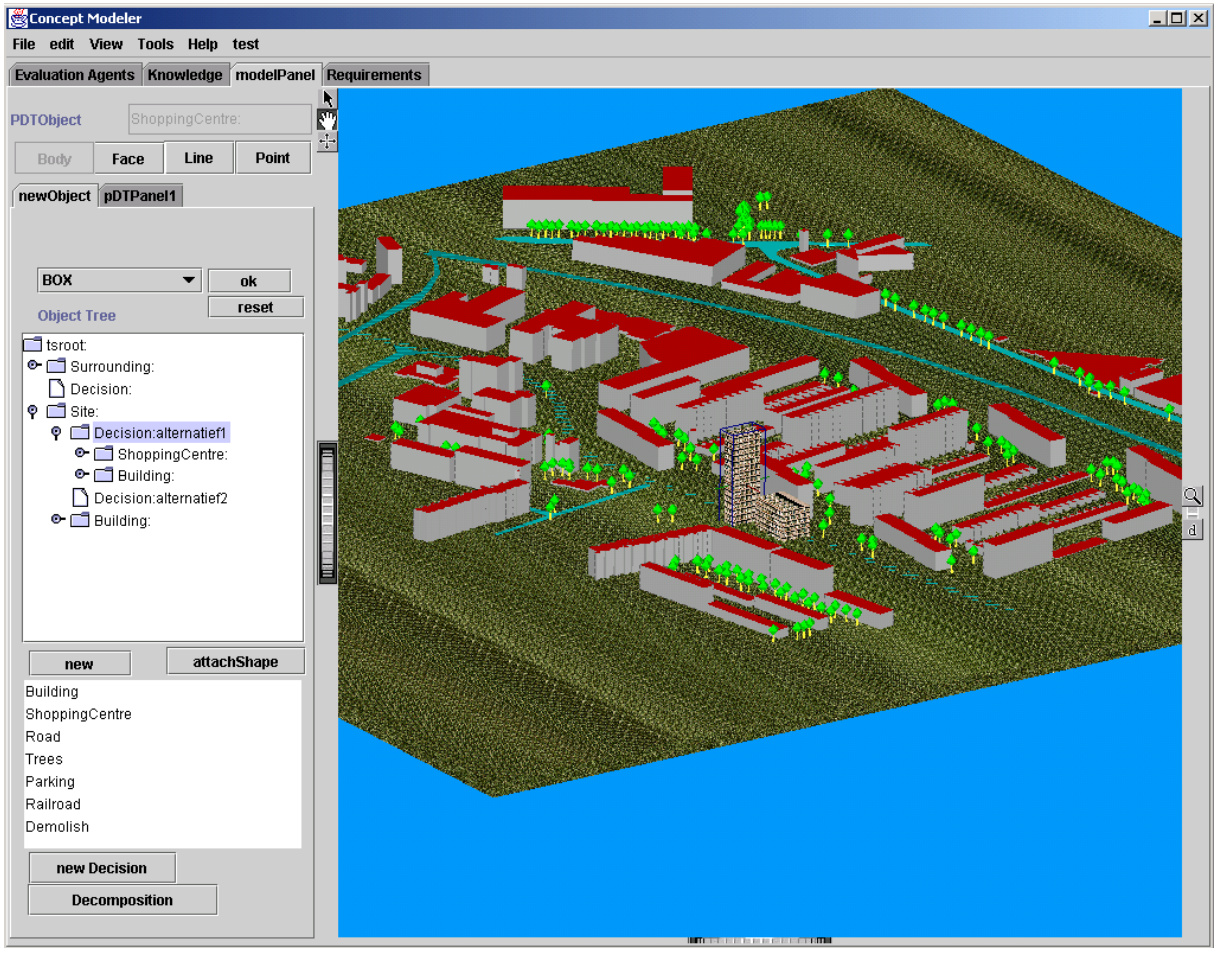

Figure 22 - example combination GIS and simple visualisation

8.3 Requirements

As discussed in D601, the ultimate goal of a VR Project Information Front-end is to support the inter and intra project communications required during the project life cycle and beyond, not only involving project partners and suppliers, but also the authorities and the general public.

These requirements do not imply that the VR project world displayed on the screens should be very detailed and that the underlying shape model formally defines the geometry and topology of the facility! Abstracted shapes and textures are in most cases sufficient and topology (i.e. how a beam is exactly connected to a column) is not relevant, those aspects can be hidden from most users.

The following requirements have to be met by the VR Project Information Front-end:

- Unlimited availability
  - Only a standard browser plug-in and standard XML vocabularies like SVG and X3D should be applied.
- Supports browsing a meaningful project world
  - If a user clicks on a column the system should know it's a column, should know its geometry, and should be able to display all its relevant properties.
- Applies information hiding

  - Applying principles of information hiding like decomposition and views is mandatory otherwise too much irrelevant information obscures the screens. Also facilities for progressive detailing, as required in the global to detailed design stage, should be supported.

- Is selective
  - It is also necessary that the system can be more or less selective and open for different types of users.
- Is multi lingual
  - As large-scale projects nowadays always involve partners and people with different backgrounds and native languages, the PIF should support communication in different languages and alphabets.
- Is bi-directional
  - The PIF should – at least to some extend – be able to support user input.

The list of requirements can of course be extended, for example with the possibility to scroll back and forwards in time, or with the possibility to use voice controlled input, or to hear the sound of the piling works next summer. Though these and other extensions are not being looked into, it seems obvious that there are no hard reasons that indicate that they cannot be implemented.

Most of the requirements listed above can be realised by using bcXML and the component related multi media extensions developed earlier. Not yet solved is the problem of creating a meaningful project world and the question: how to implement bi-directional communication.

8.4 VR Project Front-ends

Extending the research on bcXML based information front-ends beyond the component level is only possible if:

(1) bcXML is able to communicate the required data, i.e. the occurrence data

(2) if visualising additional components or building elements does not require additional programming

(3) if the resulting abstracted project visualisation is still detailed enough to be useful

(4) if the component visualisation technology (including information hiding, views, etc.) can be applied.

(5) if bcXML based multiple-lingual multiple-view bi-directional communication can be provided

8.5 XTD

What bcXML is able to communicate is defined in the XML Taxonomy Definition (XTD).

The bcXML meta-schema is totally independent of BC semantics. The meta-schema is therefore referred to as the XTD, see Figure 23 for a UML description of the XTD. The XTD is a BC-independent meta-schema used in eConstruct for modelling content-aspects for eCommerce and eBusiness transactions. More information on the eConstruct project and bcXML can be found on www.eConstruct.org.

This XTD bridges the level of XML Schema Definition (XSD) and the actual industry-sector specific schemas, like bcXML for the description of goods & services between demand and supply in both supply-chains and goods & services development life cycles. It allows for the explicit and semantic definition of objects, properties and their interrelationships.

This XTD meta-schema is seen to be complementary to and/or a potential valuable extension for the more transaction-oriented ebXML modelling work. If combined it allows the coherent and consistent development of schemas defining specification messages (demand and supply) within an ebXML envelope.

The XTD format is closely harmonised with the ISO/DPAS 12006-3 effort, which provides us with good interoperability with specification efforts like undertaken in the LexiCon [LexiCon]. The taxonomy data used within eConstruct is mainly exported from the current LexiCon database maintained by STABU.

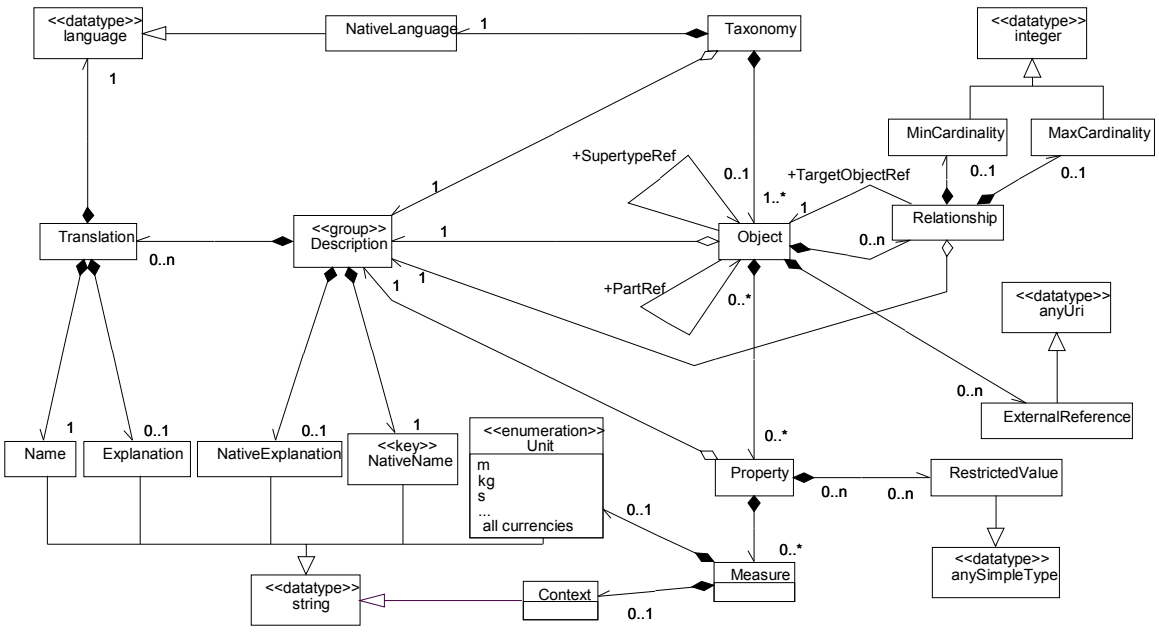

Figure 23 - XML Taxonomy Definition (XTD) meta-schema.

Taxonomy Concepts that are supported:
1. Multi-lingual data.
2. Objects.
3. Object specialisation.
4. Object decomposition (including maximum number of occurrences for different types).
5. (Assigned) Properties.
6. Units.
7. Restricted Values (i.e. "allowed values").
8. General Relationships
9. External references.

Taxonomy Concepts not supported:
1. No separate object-independent properties. Assigned Properties only.
2. No complex (aggregate) properties (atomic properties only).
3. No explicit placement of parts in wholes (other than via Properties).

The XTD is used to define the objects and their properties. This is the sole purpose of this data format. The actual data (requirement specifications, catalogues) is placed in an XML format modelled after the data in the XTD file. That format is purpose-built for easy communication according the taxonomy in the XTD file. This two-level modelling process integrates well in the multi-level ebXML developments for the transactions.

This is achieved by this four-level method:
1. The start is the XTD format definition (.xsd).
2. The taxonomy is expressed in that XTD format (.xml).
3. From that XTD-formatted taxonomy a purpose-built format definition is generated (.xsd).
4. A requirements message or a catalogue containing the actual data is made, expressed in the format generated in step 3 (.xml).

The result is a lean, small, readable format without unneeded complexity with the right amount of expression power to be able to create catalogues and requirement messages according to the definitions in a certain taxonomy. The possibility to mix definitions from multiple taxonomies provides the needed flexibility. It is not difficult to see that only simple XSLT-processing is needed to visualise this nicely to an ordinary end-user. The key-factor for obtaining the required simplicity is the strict division between Definitions (all taxonomy related items) and Specifications by generating a purpose-built schema from the taxonomy data (which is fully automated). Specifiers and catalogue builders often do not even ever see the XTD format.

9 Handling Shape and Topology

9.1 Introduction

The component VR information front-end described above is particularly suited for communications about specific types of components (no instances). One does not buy a particular door, but a particular *type* of door. For the PIF however more occurrence information is required.

Another problem is that visualizing new components or assemblies should not require programming. This means that most shape and topology information should be inherited from a small set of basic building blocks.

9.2 Generalisation

Starting point for the development of a general solution was the idea of Tolman [Tolman et al. 2002] that the world can be viewed as a collection of four groups of real world objects that can be roughly grouped according their geometry:

- point-like objects (e.g. ElectricityOutlet, WheelBarrow)
- line-like objects, (e.g. Column, Beam, AccessRoad)
- face-like objects and (e.g. Wall, Roof, SiteArea, Window)
- volume-like objects (e.g. Building, FloorSpace)

Furthermore objects can be composed of sets of smaller objects that follow the same classification. Figure 24 shows the basic set-up.

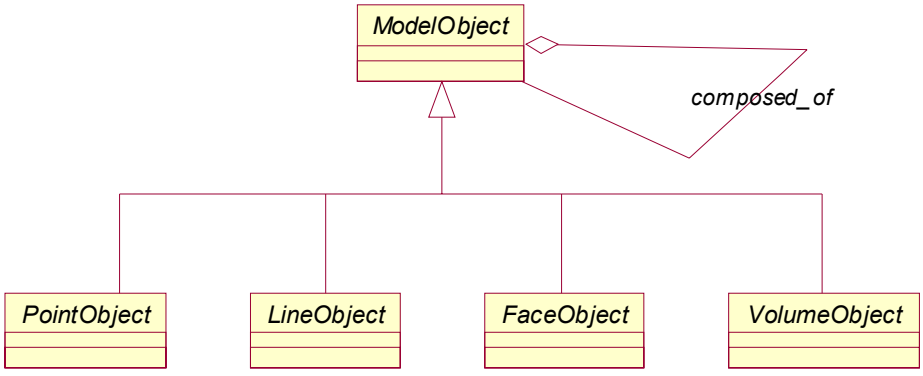

Figure 24 - The world consists of 4 types of objects.

As shown in Figure 24, PointObject, LineObject, FaceObject and VolumeObject are *abstract* classes. The basic idea is that all real world objects can be classified according to these four object types and thus can be defined as subtypes of these four *semantical* entities[2].

[2] Note the difference between the statement that a door *is* a face-like object that inherits its shape properties from the abstract class, and the statement that a door *has* a face-like shape. In the second statement shape is a property and in the first statement objects are only identified as to belong to certain general classes.

For example the class Wall inherits from FaceObject, the class Road inherits from LineObject, and the class Building inherits from VolumeObject. Note that the four basic objects are *not* shape objects but very abstracted 'things' that you can see in and around a construction project and thus also objects that can be entered in the bcTaxonomy and treated accordingly.

PointObject

Origin

X-axis, Y-axis, Z-axis

LineObject

Length

Thickness

Width

SurfaceTexture

FaceObject

Thickness

Surface Texture

FaceObjects are currently limited to flat rectangular faces without holes. Holes can be visualised however by inserting another smaller face with a different colour and a *located_on* relation.

VolumeObject

Length
Width
Height

VolumeObjects currently have been limited to rectangular boxes.

9.3 Small taxonomy made

To support the communication of above information, a small taxonomy has been made containing those objects and properties. The two pictures below are screenshots from the visualisation by the taxonomy server:

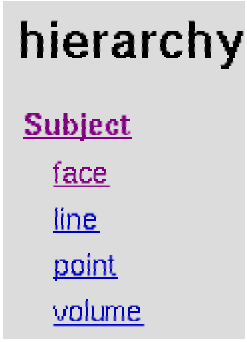

Figure 25 - screenshot first version visualisation taxonomy

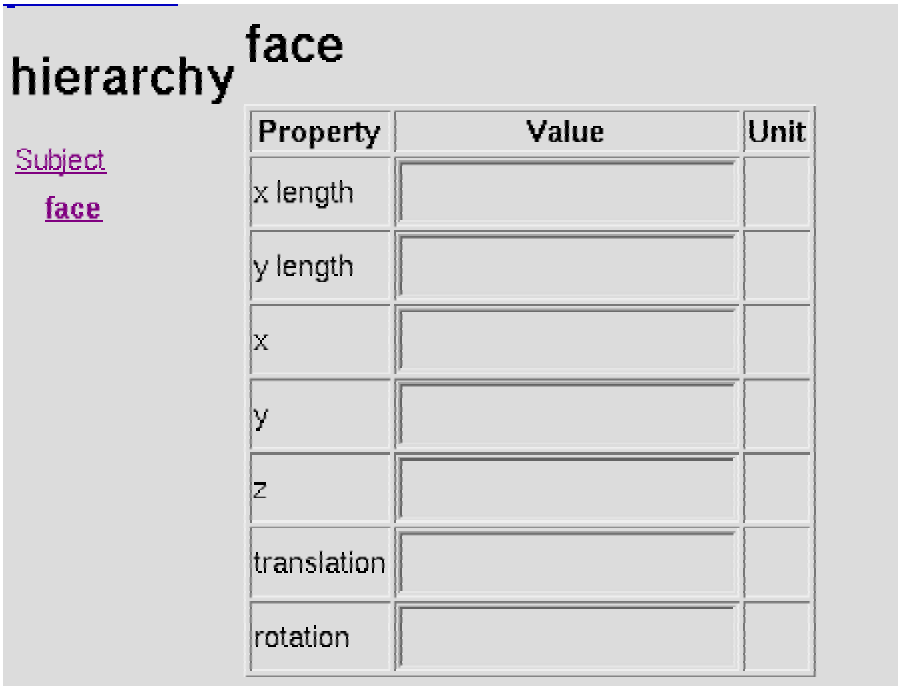

Figure 26 - properties for face (example)

The mixing of the "regular" bcbuildingdefinitions taxonomy used by eConstruct and this small taxonomy allows us to add shape information to a bcXML file:

```
<Door>
  <height unit="m" value="2.10"/>
  <width unit="m" value="0.80"/>
  <Face xtdReuse="ForProperties">
    <x value="3.20"/>
    <y value="6.00"/>
  </Face>
</Door>
```

The *xtdReuse="ForProperties"* indicates that the *Face* isn't a component, but is only used to include it's properties (this is a bcXML mechanism to deal with multiple taxonomies).

9.4 Relations

In order to implement at least *some* degree of end-user input, the model of Figure 27 has to be extended with a set of simple relations[3]. These four types of objects have two types of relations: (1) *bounding* relations and (2) *location* relations.

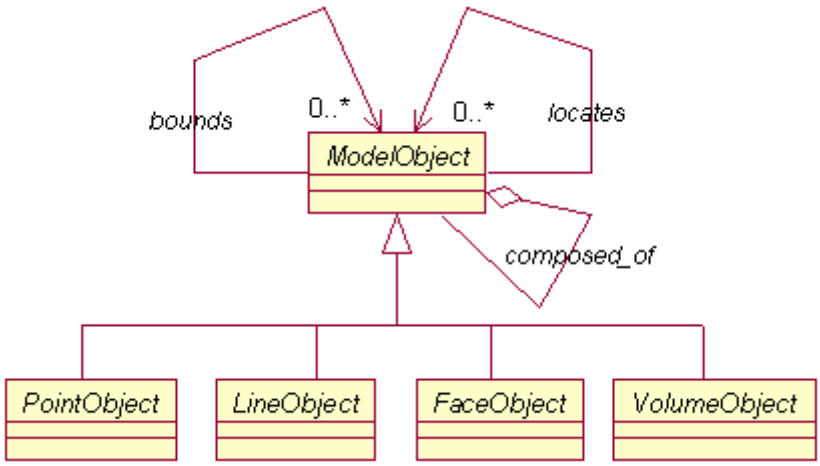

Figure 27 - ModelObjects can have bounds and locates relations.

The "bounds" and "locates" relations can be used to express the topological facts of interest, i.e. those facts that might require communication or user manipulation (like a car located somewhere on a road). Note that the purpose of Task 6200 is not to develop a traditional non-manifold type of shape model but to extend bcXML with some shape and topology information that suites the user requirements (primarily finding information but also some interaction).

9.5 Illustrative Wall Example

Figure 28 below shows what is meant by the difference between a *component* visualisation and a *project* visualisation. A component visualisation is like the prototype from Chapter 7, visualising just one item at the time. A project visualisation displays multiple objects together, in their relationships.

[3] As shown in the XTD, bcXML can also communicate general relations.

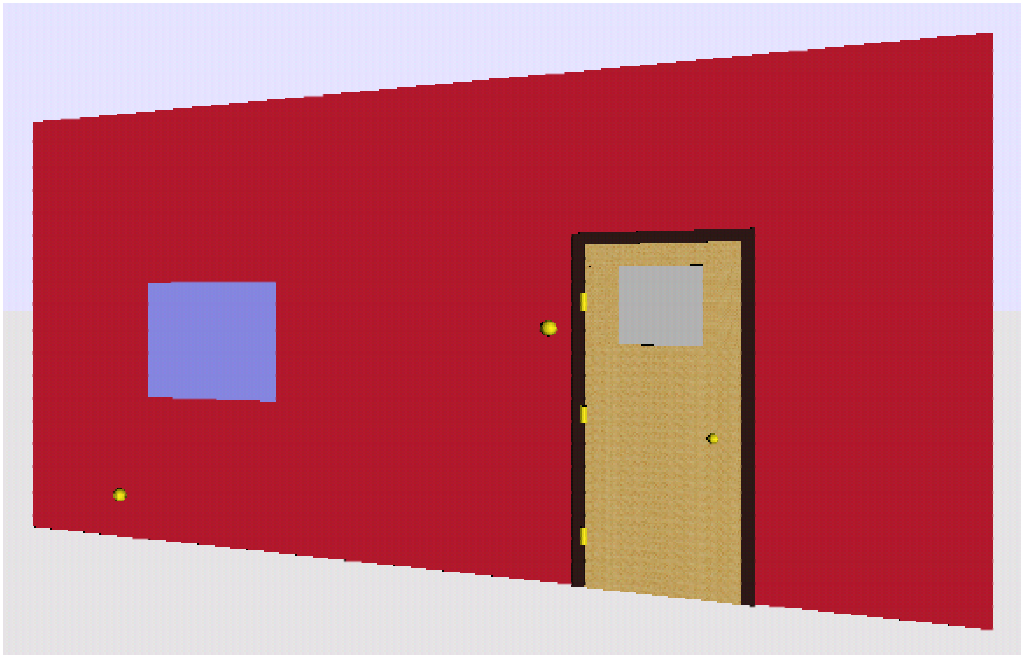

Figure 28 - illustration of a project visualisation of a wall

What you see here is a picture of a particular wall with a door, a window, a light switch and an electricity outlet. Also part of the wall is tiled. The window and door etc. are components that together form a part of a building. The bcXML file for this wall example looks like this:

```
<Wall>
  <height unit="m" value="3.00"/>
  <width unit="m" value="8.00"/>
  <Face xtdReuse="ForProperties">
    <x value="0.0"/>
    <y value="0.0"/>
  </Face>
  <Door>
    <height unit="m" value="2.10"/>
    <width unit="m" value="0.80"/>
    <Face xtdReuse="ForProperties">
      <x value="3.20"/>
      <y value="6.00"/>
    </Face>
  </Door>
etcetera etcetera
```

9.5.1 Specific Requirements[4]

One of the objectives of the study was to enable the user (the reader) to interact with the model, i.e. change the colour of the tiles, change the type and dimensions of the door and window and change the position of the door, window and electricity-outlet.

[4] Most of the component visualisation support features can be provided also for the PIF, i.e. language and view conversion, decomposition (the door in the example if in fact the same door as the previous example. Unfortunately the software is not yet working and the results will only be available at the time of the final review.

What is needed for this visualisation and interaction is that bcXML allows us to communicate the information about the local wall axes and dimensions together with the information required to display the geometry and topology of the door, window etc with the right dimensions at the right place. Also it was necessary to extend the number of objects with construction site and construction equipment related objects (like a Crane).

Though bcXML was not really designed for this type of application in mind there were no problems encountered. This simple type of project visualisations is quite important in the BC industry as very often the design or realisation of some part of a project (also in the case of the wall example above) involves people form different disciplines that are located in different offices, cities and indeed countries.

Applying the multi-media enhanced bcXML technology can be used to speed up the co-operative design/engineering process and reduce the error rates.

9.6 Building Project Example

An illustration of the ultimate envisioned VR Project Information Front-end is shown in Figure 29. The building is the World Port Centre in Rotterdam. At the point in time the reinforced concrete structure of the 6th floor (of the 24) was under construction. Two cranes and a number of sheds were used.

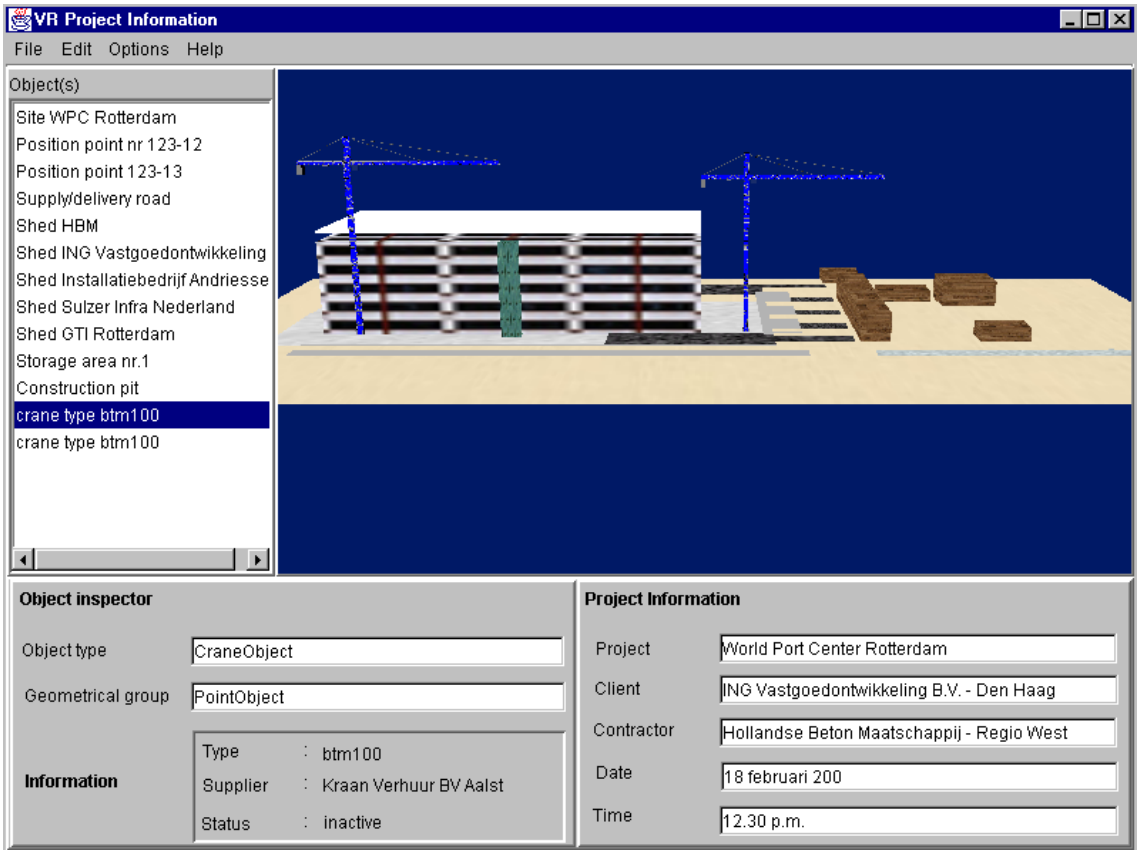

Figure 29 - VR Building Project Information Front-end

The information displayed in Figure 29shows the selection of a crane and gives some information about it. The crane's jib, pillar and weight inherit their shape properties from LineObject and PointObject.

The implementation that created Figure 29 used an existing Java 3D binding instead of the XSLT/VRML approach. The advantage was that we could use a well-established component that uses the same simplified shape objects. The down side was that some of the bcXML functionality, like view conversion, was lost. The reason for incorporating the figure was that it does illustrate the idea of bcXML supported PIF quite well.

In the future TU-Delft will develop a working PIF for general use as part of its R&D program and make it available for free.

9.7 Conclusions

VR Project Information Front-ends will play a role of increasing importance in the European Building and Construction industry, particularly if they meet the requirements discussed in section 8.3.

All (most of) the shape information required can be produced by inheriting shape visualisation methods implemented for four basic objects (PointObject, LineObject, FaceObject and VolumeObject). Together with the decomposition relation and the general topology relations it seems possible to create simplified but adequate images of complex projects under construction that can be viewed by all parties involved.

BcXML together with some of the component visualisation software has been used to create the picture of Figure 28. Figure 29 clearly illustrate the idea of a public available bcXML enabled PIF.

10 Final Conclusions

This deliverable presents the initial results of the work on bcXML based Virtual Reality Information Front-ends. Though the research will continue for a couple of years some interesting initial results have been made. The following conclusions seem to be validated[5]:

1. BcXML and related components provides a sound basis for communication improvement in (European) large-scale Building and Construction.

2. Application of low-cost powerful bcXML based communication can decrease the cost of failure of the European Building and Construction industry with at least 50%.

3. Multi-lingual multi-media enhancements combined with information hiding and context control greatly increase bcXML's capabilities and attractiveness.

4. Multi-media (VR & SVG) supported bcXML provides the European Building and Construction industry with a brake-through technology for eCommerce related communication on the component level.

5. Other multi-media extensions, for example for speech processing, and indeed also the combination of speech processing and VR, is interesting as a means for speech controlled navigation and such ("show me…tell me").

6. BcXML will be able to provide the European BC industry with the required low-cost but powerful communication technology, provided however that the work on BC taxonomies will be continued. It is strongly recommended that a number of bcXML compliant taxonomies (for Building and Civil Engineering), each containing a basic set of object definitions, will be made available in all the languages and alphabets of the current and future member states.

7. The European Commission should take a close look at the eConstruct results and try to get consensus about the recommendation above as this recommendation calls for a new type of EU-project involving governmental representatives of each of the (future) member states. As bcXML based electronic communication is a prerequisite to the unification of the BC industry, it is recommended that the Commission should organize such a project.

[5] As the project duration has been shortened from 3 to 2 years and WP6 could only start after bcXML had been stabilised the time span was too short to produce thoroughly investigated results and conclusions.