Developing Semantic VR-shops for e-Commerce

Olga De Troyer, Frederic Kleinermann, Haithem Mansouri, Bram Pellens, Wesley Bille, and Vladimir Fomenko

Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussel – Belgium

Email : Olga.DeTroyer@vub.ac.be
Tel : +32-2-6293504

Abstract

Increased bandwidth, cheaper and faster hardware, dedicated technology and the success of e-commerce make VR-shops feasible. VR-shops are similar to the e-shops currently available on the Web, with this difference that the products are visualized as 3D objects in a virtual world. Although VR-shops do not require sophisticated VR technology, they should be very flexible: it should be easy to add, remove and rearrange products; and to add, change or remove functionality. Therefore, an appropriate approach that can be used by a non-VR expert and that provides a short development time and easy maintenance is necessary. Also usability is very important because this is crucial for the success of VR-shops. In this paper, we present an approach to develop VR-shops that meets these requirements. It allows specifying a VR-shop using high-level conceptual specifications and in terms of domain terminology; semantics are captured by ontologies; existing product information can be incorporated; and the actual code is generated.

Keywords: VR-Shops, e-shops, Ontology, Semantics, Web Services, Semantic Annotation, Semantic Search Engine, Shop-WISE

1 Introduction

Over the last fifteen years, the Internet has evolved from a simple text-publishing platform into a powerful distributed software environment enabling development and deployment of complex interactive, multimedia applications. Currently, these applications range from information and entertainment services to complex e-commerce, e-business, and e-government applications.

During the same period, 3D computer graphics technologies have also evolved to a level that makes it possible to use them in a diversity of real-life applications including Web applications. This has been possible due to (1) significant progress in hardware performance, including cheap 3D accelerators available in almost every contemporary computer and which enables faster rendering, (2) increased bandwidth of the networks, (3) VR standards specially dedicated for the Web (VRML/X3D), and (4) increased availability of 3D modeling tools and consequently also 3D content.

Today, there are different e-commerce applications available on the Web, a lot of them are e-shops that allow people to buy or inspect products online (e.g., Amazon [1], Dell™ [2], IKEA® [3]). In these e-shops, products are displayed using text descriptions usually complemented with a photo and organized according to some criteria. A more appealing way would be to display the products using Virtual Reality (VR). In such a Virtual Reality shop (VR-shop), the products are visualized as 3D objects in a virtual world where the user can walk around, pick up, and inspect the products [4]. A study of Haubl and Figueroa [5] shows that buyers tend to spend a greater amount of time viewing 3D products than if the products are visualized as still images. Chittaro and Ranon [6] point out the following benefits of VR for E-Commerce: (1) it is closer to the real world shopping experience (and thus more familiar to the buyer); (2) it supports buyer’s natural shopping actions (like walking, looking around the store); (3) it can satisfy emotional needs of buyers by providing a more immersive, interactive, and visually attractive experience; (4) it can satisfy social needs of buyers by allowing them to meet and interact with people.

Furthermore, VR-shops not only contribute much more to customer retention than ordinary e-shops, they also lend themselves better to incorporate well-proven marketing strategies. As visitors of a VR-shop are more likely to spend more time looking at items, there is also more chance that they will look at items they may not have intended to look for when they entered the shop. Promotional displays can be used, and commercial techniques like “massification” are also possible with VR-shop [6]. Furthermore, as sophisticated as they may get, 2D e-shops could be considered as being nothing more than a Web site. With a few clicks, users can easily navigate to other sites and might even end up at a competitor’s site. VR-shops, on the other hand, give users the impression of being “emerged” in the world. Users are “transported” from their familiar Web environment where they follow their own navigation laws, to that of the shop owner. Within such an environment, users are forced to follow the shop owner’s navigation rules (or leave it altogether). Therefore, from a mental point of view, it may be more difficult for users to navigate away from the shop once they are in it.

From the user’s perspective, VR-shops also offer many advantages compared to the traditional 2D ones. Users cannot only contemplate the displayed products from every angle but, depending on the possibilities of the application, they could also alter some of their features (like color or size) and immediately see the results of their actions [6]. Obviously, this facilitates product selection and customization. In the case of a virtual furniture shop for instance, the application could offer the possibility to display some chosen items together, e.g., in a single virtual room. Users would then be able to visualize how a specific combination of furniture would look like and if it would fit in a customized environment. They would also have the possibility to rearrange the different pieces of furniture according to their taste and they can experiment with different colors or types of furniture in order to choose the best combination according to their needs and taste. In the context of a clothing store, VR-techniques can be used to compose customized products (e.g., shirts or pants).

Although VR-shops do not require sophisticated VR technology, they should be very flexible, as they need to respond to the ongoing daily changes in terms of products and prices. Therefore, it must be easy to add and remove products, and to rapidly change product parameters like prices and delivery time. In addition, it should also be easy to add, adapt or remove functionality, e.g., adding the functionality of reserving products, changing or extending the payment method. And finally, it should be easy to adapt the virtual shop itself. Like real shops, they should be brushed-up regularly and products should be rearranged to fit special occasions or to draw attention to new promotional items for instance. Another important issue is their usability. This is a big challenge for VR-shops as they are targeting the Web community and therefore, they need to be usable by a large audience with very different backgrounds. This means that the navigation and the interaction must be easy and intuitive.

For all of these reasons, an appropriate approach for the development of VR-shops is needed. In addition, we believe it is important to target an approach that can be used by a non-VR expert because this will facilitate the development of VR-shops, as companies will no longer be forced to pass by a VR-expert to implement their ideas. In summary, this means that we are looking for an approach suitable for non-VR skilled people that provides short development times, results in systems that are easy to maintain and adapt, and makes provision for good usability and re-usability.

In this paper, we present our approach to develop VR-shops, called Shop-WISE that was developed to meet the specified requirements. The approach allows the developer to specify the VR-shop using high-level, conceptual specifications and in terms of concepts from the shop-domain. Semantics are captured by means of ontologies. Existing product information can be re-used. Finally, the actual code is generated from the high-level specifications. This means that in order to adapt a VR-shop, the modifications can be made at the conceptual specification level and the adapted shop can be easily re-generated. In this paper, the approach is illustrated by means of a furniture shop. The user can walk through the VR-shop, look at products, ask for information and buy or reserve products. In addition, a semantic search engine is provided to find products based on their semantic properties.

This paper is structured as follows. Section 2 discusses related work concerning VR-shops and VR semantics. Section 3 describes how a VR-shop is designed using the Shop-WISE approach. Section 4 explains how the approach makes provision for a semantic search engine. Section 5 describes an example furniture shop that we developed with the approach, while section 6 evaluates the approach and discusses further work. Section 7 presents conclusions.

2 Related Work

This section is divided into two parts. The first part gives an overview of related work on VR-shops. As our approach also provides the benefit of generating semantic information (called semantic annotations) usable at run-time, the second part of this section will review research work related to the use of semantic information for enhancing the usability of a VR-application in general.

2.1 VR-Shops

Some research work oriented towards 3D VR-shops is already available. Here, we will discuss the work relevant to the work presented in this paper.

The Alice project [7] is a relevant project, as it uses ontology for representing knowledge related to online shopping making it more like visiting a local corner shop than browsing or searching long lists. It uses ontologies for describing products, shopping task and customers. Alice provides way of making dynamic queries so that the user can quickly find relevant items by displaying the result of queries. The use of ontologies makes the interface more intuitive not only for the customer but also for the shop manager. Similar as in our approach, Alice uses ontologies to make online e-shopping, however it does not provide a 3D virtual environment like we do.

The project 3D-dvshop (3D-dynamic virtual shop) [8] presents an approach to generate personalised VR-shops. In this work, the user can select 3D products from an existing list of products and a 3D VR-shop will then be generated automatically containing the selected products. In this approach, the end-user is only responsible for the generation of the content of his VR-shop. He cannot decide how the products are placed, or decide about the layout of the VR-shop, which is possible in our approach. Some additional information besides the properties related to the graphical representation of the virtual objects is held in a database. However, this information is not used within the virtual shop itself, it is only outside the shop area that these information items can be retrieved. Furthermore, this information is not used for searching virtual objects inside the virtual environment. In our approach, this kind of information is integrated in the virtual environment and can be used for retrieving and searching products or locations.

VRCommerce [9] is an integrated solution for creation, operation and navigation of 3D malls and stores. It tackles the problem of continuous navigation between separately designed and managed stores. However, it does not address the issue on how to design 3D malls and stores in an intuitive way. Although you can add products inside the shop, it is not clear how easy it is to position them inside the shop. Furthermore, there are no real explanations about the management of the shop.

Chittaro and Ranon [10][11] present a set of guidelines for designing VR e-commerce sites. They introduce the concept of walking products to aid the user to find products in the shop. They also talk about “Massification” and how to attract the attention of customers. In [6], Chittaro and Ranon present a general approach to build adaptive 3D websites called AWE3D. Its usefulness is illustrated by means of a 3D e-commerce case study. A so-called “VRML Content Database” is used to create a personalized VR-shop considering the user data. The approach includes a number of nice features (like navigation and interaction) that help the user to find what he is looking for inside the VR-shop. Although the parts of the shop are self-adapting according to the user’s behavior, the shop is actually pre-defined and the adaptability only exists through modifying the parameters of the VRML PROTOS node. In our approach, each shop can be designed individually and the design as well as the maintenance is at a much higher level than the VR language used to the implementation.

Robles et al. [4] have also considered adaptive 3D online stores. They provide adapted versions of the e-shop for each user by means of a user mode and a set of rules. However, they do not address the issue on how to position products easily inside the shop and how quickly they can add (or remove) products.

2.2 Semantic Annotations

Semantic annotation is information that is added to some media to enrich it with a well-defined meaning. Semantic annotations are especially important in the context of the Semantic Web because they make the content of the Web machine-processable and enable computers and people to work in cooperation. In the context of VR, semantic annotations can be very useful for enhancing the interaction and the usability of the VR application. Already some research has been performed dealing with annotations and semantics for VR.

In [12], the world is annotated using information from the 3D structure of the virtual world. The annotations take into account: viewpoints, areas of interest, objects, persons, and text. The annotations are used as semantic data to help the user to navigate inside the virtual world. Although the navigation improves, their annotation is mainly related to the 3D structure of the virtual world and they do not provide semantics of the real world objects like e.g., price, quality, or delivery time. In our approach, the semantics added during design are not related to the 3D structure of virtual world (these type of annotations are generated automatically) but are concerned with real-world semantics. These real-world semantics will be used to improve the usability of VR-shop.

In [13], the notion of smart object is used to provide not only the geometric information necessary for drawing objects on screen, but also semantic information useful for manipulation purposes. Here, the semantic information in the smart object is used by virtual characters to perform actions on/with the object e.g., grasping, moving, operating. Using the semantic information, the user is much more aware of the sort of manipulation that he can perform. For this reason, our approach also provides these features by using the semantic information added at design time. However, in our approach, it is the domain expert (i.e. the shop builder) that specifies this information.

In [14], the authors provide a semantic representation capturing the functions, characteristics and relationships between virtual objects. The model they propose is designed to turn the objects in a Virtual Environment into autonomous and reusable entities that they call digital items. Our approach exploits a similar idea by building a semantic layer, which will keep track of relations between the virtual objects. In our approach, this semantic layer will be exploited by a search engine for finding virtual objects inside the VR-shop.

3 Developing VR-shops using the Shop-WISE approach

Shop-WISE, the approach used for developing VR-shops is based on a more general approach for developing VR applications, which is called VR-WISE [15][16][17][18][19]. The development process in the VR-WISE approach consists of three (mainly) sequential steps, namely the *Specification Step*, the *Mapping Step* and the *Generation Step*. For the Specification Step, VR-WISE uses high-level modeling concepts and (domain) ontologies to specify a virtual world. The result of the specification step is a high-level description of the objects in the virtual world; how they are related to each other; and how they will behave and interact with each other and with the end-user. In the mapping step, the conceptual specifications are mapped on VR-implementation primitives and in the generation step actual code is generated. The VR-WISE approach has been customized for the development of VR-shops. This means that a number of object (type)s and behaviors have been pre-defined (and provided in the form of (shop) domain-specific ontologies); also provision for plugging-in shopping functionality (e.g., buying, reserving products) has been added. In this section, we explain the different steps for the development of a VR-Shop using the Shop-WISE approach. Figure 1 illustrates the different steps and the different modules involved in each step. Subsection 3.2 explains the Specification Step, subsection 3.3 the Mapping Step and subsection 3.4 the Generation Step. We start with a short subsection about ontologies.

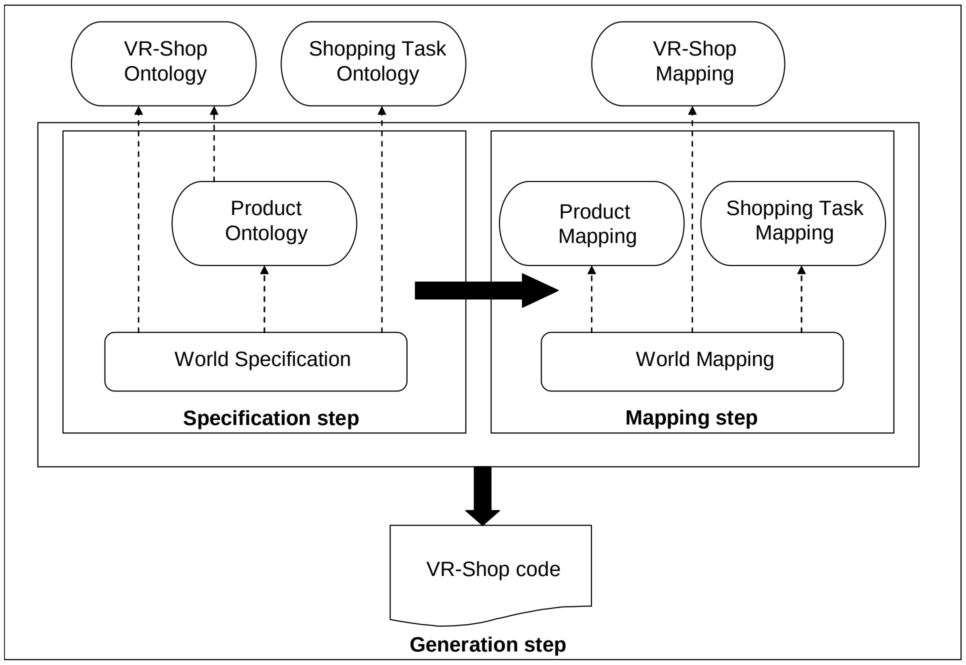


Figure 1. Overview of the development process of Shop-WISE

3.1 Ontologies

In the field of computer science, the term ontology is used in the context of knowledge representation. In [20], an ontology is defined as a formal, explicit specification of a shared conceptualization of a domain of interest. A conceptualization is an abstract, simplified representation of the world in terms of objects, concepts, and other entities that are assumed to exist and the relationships that hold among them. The term shared implies that the conceptualization is shared among different stakeholders i.e., they all must agree on a specific representation of the world. As it is the purpose of an ontology to represent knowledge (semantics) in a machine-readable and machine-interpretable manner, the conceptualization should be specified formally and explicitly. The semantics of concepts and relationships between these concepts should therefore be expressed in a formal language. Currently, different ontology languages are available. The best known is OWL [21], the W3C Recommendation language, which is the successor of DAML+OIL [22]. DAML+OIL was the result of the cooperation between the DAML project [23] and the OIL language group [24].

3.2 The Specification Step

Specifying a VR-shop is done at two levels: the type level and the instance level. The type level allows specifying the different types of products (e.g., lamps, desks, beds) that will be available in the shop. At the instance level, the actual shop is composed: a virtual world in which the products can be placed is specified (i.e., the layout and the infrastructure of the shop); actual products (instances) are specified and placed in the world; and the shopping functionality that will be provided (e.g., buying products, reserving products, ordering products) is selected.

The specification typically starts with the type level. The specifications of the different types of products are captured in the *Product Ontology*. Next, the instance level is specified by means of the *World Specification*. To support and ease the specification of a VR-shop, two ontologies describing the VR-shop domain are provided. Useful VR-Shop concepts are pre-defined in the *VR-Shop Ontology*. Useful shopping tasks are pre-defined in the *Shopping Task Ontology*. Each of these ontologies is described into more detail in the following subsections.

3.2.1 VR-Shop Ontology

To support and ease the specification of a VR-shop, a number of concepts from the VR-shop domain are pre-defined. These are the concepts that can be used for building the actual shop environment (the shop infrastructure) as well as generic concepts useful for each VR-shop. As already indicated, they are defined and made available by means of an ontology.

A first group of concepts in the VR-Shop Ontology groups concepts usable for specifying the infrastructure of the shop. Examples of these concepts are *Wall*, *Floor*, *Ceiling*, *Area*, and *Rack*. As an example we consider the *Wall* concept. The *Wall* concept has three properties for defining the size of individual wall-instances namely *height*, *width* and *thickness*. *Wall* also has additional properties like *color* and *texture* to be able to specify the type of material of a wall-instance. In a similar way, the concept *Floor* is defined. It also has three properties for defining its size namely a *length*, a *width* and a *depth*. As for the concept *Wall*, the properties *color* and *texture* can be used to specify the material of a floor. The concept *Area* can be used to define areas in the shop that have a specific purpose. Some areas are pre-defined like the *Entrance*. Another important concept for a

VR-shop is the shopping *Cart* concept used in almost all e-shops to collect the items the user wants to buy. If needed, new shop concepts can be added to this VR-Shop Ontology.

Further on, each VR-shop will contain products. These are the products for sale. Therefore, in the VR-Shop Ontology, the (abstract) *Product* concept is provided. The concept *Product* is pre-defined with some (possible optional) properties that are usually applicable for products. Example properties are:

- *Name*. Name of the product.
- *Description*. General (textual) description of the product.
- *Category*. The category (or categories) that can be used to classify this product. E.g., "electronics". A hierarchy of categories can be used to classify a product. E.g., "Computers" inside “Electronics”.
- *Origin*. The region of origin of the product. E.g., “Belgium”.
- *Color.* Color(s) in which the product is available.
- *Material.* Materials of which the product is made. E.g. “leather”.
- *Weight*. Weight of the product.
- *Quality*. Quality label(s) attached to the product.
- *Price*. Regular price of the product.
- *Unit_of_Sale*. The unit at which the product is sold. E.g.: “per kilogram” and “per piece”.
- *Price_Comment.* Additional factors specifically affecting the price.
- *Sales_Term*. Conditions that determine the type of sale. E.g., "Delivered sales", "F.O.B".
- *Season.* The crop-year of the commodity, based on the harvest start date.
- *Packing*. Container or package in which the product is sold. E.g., "Cartons Tray Pack".
- *Storage*. Storage or other external factors affecting the product. E.g., “Controlled Atmosphere Storage,” “Regular Storage” and “Unwashed”.
- *Compares-to*. Other products that are comparable with the product.
- *Add-ons*. Products that can be related to the product. E.g., wine and cheese.

- *Demand*. Represents the immediate or current desire for the product and the ability and willingness of the buyer to buy it. Can be useful for statistics.
- *Import_Export*. Indicates whether the sale is for domestic consumption or for exporting only, or for both.
- *Only_for_Adults*. Indicates if the product should only be available for adults. E.g., alcohol, cigarettes.
- *Visualization*. This allows specifying the visual (3D) representation for the product (see also section 3.3).

All product types that will be available in the shop should be defined as subclasses of *Product* (see section 3.2.3). In this way, they will inherit the common properties defined for the concept *Product*.

A third category concerns pre-defined types of behaviors, which may typically be available in VR-shops. For example, it may be useful for the user to be able to turn around a product or to select products and put them in the shopping cart or to remove them from the shopping cart. Therefore, behaviors like *turn*, *changeColor*, *addToCart* and *removeFromCart* are also pre-defined. These behaviors can later be associated with specific products. *showContent* and *makeEmpty* are also pre-defined behaviors to allow showing the content of a shopping cart and to make it empty. Note that although these behaviors are pre-defined, the VR-Shop Ontology does not specify how they should be implemented (further explained in section 3.3).

3.2.2 Shopping Task Ontology

Most e-shops provide the same kind of functionality: information about products can be requested; products can be bought; sometimes it is possible to reserve products; the user can register; he can ask to be informed about actions and promotions; etc [7]. These different tasks are collected and defined in the Shopping Task Ontology. Examples of tasks defined are:

- *Get price.* Get a product's price.
- *Get info.* Get a product's description.
- *Choose currency.* Allows the user to choose his preferred currency and use it from then on.
- *Convert price.* Convert a product's price to a specific currency.
- *Check stock.* Check the availability of a specific product.
- *Compare with similar products.* This task should allow users to compare prices and features with other similar products in the same shop.
- *Post product review.* Allows a user to write a small review about the product in question and/or his experience with it.
- *Calculate shopping cart total.* Return the total amount to pay for all items in the shopping cart.
- *Calculate product-shipping cost.* Calculate the shipping fee for one specific product.
- *Calculate total shipping cost.* Calculate the total shipping cost for all the products in the shopping cart.
- *Buy.* Buy one or more products in the shopping cart.
- *Order product.* If a product is not available, order and reserve it.

For these tasks, we have defined the required parameters and for complex tasks, we have defined how they are composed of other tasks. Similarly as for behaviors, it is not specified in the Shopping Task Ontology how they are implemented.

3.2.3 Product Ontology

Each type of product that may be for sale in the shop should be defined as a concept in the Product Ontology. For the VR furniture shop, we have defined concepts such as *Cupboard*, *Chair*, *Lamp*, *Sofa*, *Bed* and *Table.* As explained in section 3.2.1, these concepts should be defined as subtypes of the generic concept *Product* defined in the VR-Shop Ontology. In this way, they inherit the common properties defined for *Product*. For each product type additional relevant semantic properties and information can be specified. In our example of the furniture shop, for the product *Sofa* the following additional properties have been defined: *length*, *depth*, *height*, *numberOfPlaces*. Depending on the products available, it may be useful to build a product hierarchy (next to the categorization-properties for Product). For example, a furniture shop will usually have different types of lamps like hanging-lamps, table-lamps and desk-lamps. They can be defined as subtypes of the product *Lamp*. This allows for automatically inheritance of properties. We will see that this kind of semantic information can be used by the end-user to search for products. Other types of semantic information can also be specified to allow, later on (in the actual VR-shop), more advanced searching. To support this, the notion of *Rule* is introduced. A rule provides some adjective together with a meaning so that products can be queried using this adjective. E.g., we can specify a rule to allow judging if ‘a study lamp is non-expensive’. A rule has four elements: an adjective (e.g., non-expensive), the property of the product on which this adjective applies (e.g., *price*), the comparison operator (e.g., less than) and a judgment value that provides some meaning for the comparison (e.g., 15 EUR). When a value is given to the properties involved in a rule, the rule will be evaluated and the appropriate semantic information (the adjective) will be associated with the product. Later on, a search engine for VR can use this information to answer end-user queries like ‘find all lamps that are non-expensive’. In this way, only simple forms of rules can be expressed. However, it is possible to replace this rule concept by a mechanism that is more powerful, e.g., OWL Rule language [25] or SWRL [26].

3.2.4 World Specification

When the type level is defined, the designer can start to specify the VR-shop infrastructure together with its layout and place product instances inside it. To create the virtual world that represents the shop, the designer should instantiate concepts from the VR-Shop Ontology that are relevant for the architecture of the shop. For example, he may instantiate the concept *Wall* a number of times to create some walls (e.g., a *BackWall* and a *LeftWall*). Each of these *Wall*-instances will inherit the properties defined for the concept *Wall*. The designer should specify the values of the properties of these instances (or use the defaults). For instance, one of the *Wall*-instances was given a value for the *height* of 300 cm, for *width,* 1300 cm and for *thickness,* 20 cm. Also a Floor-instance was created. Then, the walls and the floor need to be positioned. Next to the traditional way of positioning instances by means of coordinates and to orientate them by means of angles, we also provide a more intuitive way in which objects are positioned and oriented with respect to other objects. At this moment, two kinds of high-level relations are provided namely *spatial relations* and *orientation relations*. A spatial relation uses one or more high-level concepts (***LeftOf***, ***RightOf***, ***InFrontOf***, ***BackOf***, ***Above***, ***Under***, ***Middle*** and ***OnTopOf***[1]) to position an object with respect to another object. Orientation relations are used to orientate an object with respect to another by specifying which part of which side of an object is oriented to which part of which side of the other object. More on these high-level positioning and orientation relations can be found in [17]. Using these relations for our example, it is easy to position the walls of our shop. E.g., we can position the *BackWall* ***Above*** the *WorldFloor* and the *LeftWall* ***LeftOf*** the *BackWall*. No distance is specified here for the spatial relations as the walls are touching each other. The

[1] Products have a left, a top and a front by default.

orientation can be specified by saying e.g., that the **front** side of the *BackWall* is directed towards the **right** side of the *LeftWall*. Although, this way of positioning and orienting may be less powerful and less precise than the use of coordinates and angles, it is more appropriate for novice users, not experienced with VR.

Using the same principle, furniture can be created and placed inside the shop. Instances are given unique names for ease of reference. If only one instance per product type is created, it may be convenient to use the name of the product type as name of the instance too. In our example shop, an instance of *DeskGustav* was created and positioned against an internal wall created inside the shop room (*DeskGustav* ***Against*** *InternalLeftWall*). Also an instance of *LampExpressivo* was created. This lamp can be positioned with respect to the *DeskGustav* instance by using a spatial relation to express that the lamp is placed on top of the desk (*LampExpressivo* ***OnTopOf*** *DeskGustav*). These high-level relations also provide some kind of semantic relationships between the instances. The semantics of these relations can be used later on by the end-user of the shop to ask questions like ‘what is left of DeskGustav?’. In the same way, the other furniture is placed in the furniture shop.

Now that we have created the shop and placed all the products, it is possible to attach behaviors and tasks to instances. Remember that these were pre-defined in the VR-Shop Ontology and the Shopping Task Ontology. In this way, it is possible to specify different behaviors and tasks for different product instances e.g., some products can only be ordered while others can be bought directly. In principle, it is also possible to specify behavior that has not been pre-defined, however this is not very common for a VR-shop. More on this can be found in

[18][19]. When attaching behaviors and tasks to instances it must also be specified how it could be invoked. Invocation of a behavior or a task is usually done by means of user interaction. For example, the designer can specify that when the end-user *points to* a product-instance its product information should be displayed or when the end-user *clicks* on a *RollingChair* instance it will rotate. As most common user interactions for desktop VR are provided as primitive concepts (e.g., *clicking*, *pointing to*), the designer only has to select one (or several) and attach it together with the desired behavior or task to the instance.

3.3 The Mapping Step

In the Mapping Step, the second step in the development process, the mapping from the conceptual level to the implementation level must be specified. This means that we must specify how objects defined in the World Specification should be visualized using VR-implementation primitives and how the behaviors and tasks should be realized.

How objects should be visualized is, in principle, specified by mapping each object to implementation primitives. However, to avoid having to define similar mappings for similar objects over and over, the mapping is defined at two levels. First, default mappings are specified for the object types (concepts). Next, these default mappings can be adapted or completely overwritten for particular instances (objects) of the object type. This principle is applied to the different ontologies specified in section 3.2.

To ease the work of the designer, a lot of default mappings are pre-defined. For all concepts defined in the VR-Shop Ontology, a default mapping is given. For example, the default mapping for *Wall* is a box. The attributes of the *Wall* concept are mapped on the attributes of the VR-primitive Box: *height* onto the attribute ‘”height” of the VR-primitive Box, *length* onto the attribute “width” of the Box, and *thickness* is mapped onto the attribute “depth” of the Box. In this way, each instantiation in the World Specification of the *Wall* concept will by default be represented as a box. Note that we have tried to abstract from a specific VR-implementation technology by providing VR-primitives that are available in (almost) all VR-implementation environments. It may be possible that a single concept cannot be mapped on a single VR-primitive. In this case, complex objects need to be used. More on this can be found in [17].

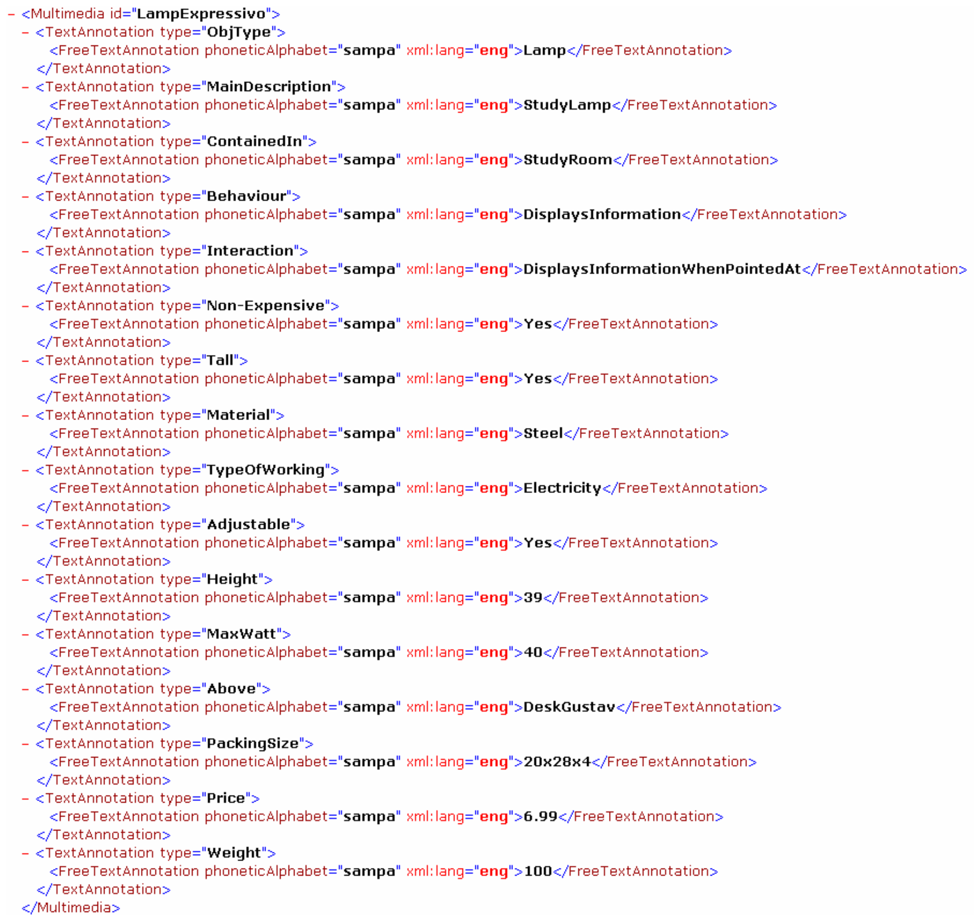
What is left to the designer is the specification of the mappings for the Product Ontology, the Task Ontology and the World Specification. The mapping for specifying the visualization of the products is kept simple. The designer can add 3D visualizations of its products in a library. Using this library, the designer can use the *visualization* property of a *Product* (see section 3.2.1) to point to the corresponding 3D object in the library, possibly together with a number of parameters to influence the visualization of the object (e.g., scale or color).

The World Specification mainly contains two kinds of objects: the objects specifying the infrastructure of the shop and the products for sale. For the mapping of the infrastructure objects the designer can largely rely on the default mappings specified in the VR-Shop mapping. However, as already explained, such a default mapping can be overwritten. For example, the concept *Wall* has been mapped by default onto the VR-primitive Box. If the designer likes to have one of the walls in the shop as a curved wall, he can map this wall-instance onto the VR-Primitive “2D Curved Surface”. Here, some help from a VR-expert may be needed if the designer wants to make non-trivial modifications. For product instances, the mappings specified for product types are used as default mapping.

This leaves us with the mapping of the shopping tasks. Although, conceptually most shopping tasks available in an e-shop are the same, the way they should be realized and how they are implemented will be different. For example, the task *Buy* may be implemented differently in different VR-Shops. In some shops, it is only possible to buy goods by means of a credit card while in other shops other payment methods are also possible. To solve this issue and to abstract from implementation issues, tasks are mapped on web services [27]. We suppose that for each task that needs to be supported, a web service exists or can be built. The use of web services makes the system flexible (one service can easily be replaced by an equivalent one) and independently of how the underlying procedures (e.g., using a database system) are implemented. It also allows for personalization as e.g., the *GetPrice* task can be mapped on a web service that calculates the price depending on the discount assigned to the customer and on current promotions. Also publicly available web services can be used, e.g., to check credit card information, to calculate the shipping cost if shipping is handled by an external company, or to convert prices into different currencies.

3.4 Code Generation

The Generation Step generates the actual code for the virtual world specified in the Specification Step and using the mappings defined in the Mapping Step i.e., the conceptual specifications are converted into a working application by means of the mappings. This is done by the tool OntoWorld, developed to support the VR-WISE approach. More on OntoWorld can be found in [17] and [18]. Actually, OntoWorld generates two files. A first file contains the 3D scene structure with its objects. This has an X3D format [28] or VRML format [29]. The second file contains the semantic annotations and is generated in an MPEG-7 format [30]. MPEG-7 is used, as it is an ISO/IEC standard supported by the industry [30] and it is readable by human and machine [31]. It also has a tree representation and therefore, it can easily be used to make a one-to-one mapping between the 3D scene structure generated by our tool and its semantic information. A fragment of such an MPEG-7 file for the virtual furniture shops is given below in figure 2.



```
- <Multimedia id="LampExpressivo">
  - <TextAnnotation type="ObjType">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">Lamp</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="MainDescription">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">StudyLamp</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="ContainedIn">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">StudyRoom</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Behaviour">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">DisplaysInformation</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Interaction">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">DisplaysInformationWhenPointedAt</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Non-Expensive">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">Yes</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Tall">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">Yes</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Material">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">Steel</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="TypeOfWorking">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">Electricity</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Adjustable">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">Yes</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Height">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">39</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="MaxWatt">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">40</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Above">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">DeskGustav</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="PackingSize">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">20x28x4</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Price">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">6.99</FreeTextAnnotation>
    </TextAnnotation>
  - <TextAnnotation type="Weight">
      <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="eng">100</FreeTextAnnotation>
    </TextAnnotation>
  </Multimedia>
```

Figure 2. Example of MPEG-7 Description

In the MPEG-7 file, each annotated object is represented as a 'MultimediaContent' object type with a unique identifier ('id'). This identifier corresponds to the object's VRML node name. This id can be used to identify the different objects in the description. Each annotation is composed of a keyword and a value. In this example, we can also see that spatial information is also automatically added (in this case, our ‘LampExpressivo’ is ‘Above’ the object ‘DeskGustav’) as well as the information derived from the Product Type hierarchy (LampExpressivo is a Lamp). The result of validating the rules has also been added to the description. We can see for instance, that the label ‘Non-expensive’ has been added. Moreover, all the properties, such as ‘MaxWatt’, ‘Price’, defined for the object (by means of its concept type) are included.

We have chosen to generate two files (one containing the 3D structure of the virtual world and one containing the semantic annotations) as it provides some advantages for the maintenance of the VR-Shops. For instance, if the (fixed) prices of the products for the VR-Shop have changed only the MPEG-7 needs to be updated. Finally, it is interesting to note that thanks to the MPEG-7 standard, such descriptions need not be confined to a single language. It is perfectly feasible to include different descriptions for the same keyword in order to support different languages. In this case, it suffices to change the language parameter of the ”FreeTextAnnotation” tag to differentiate between the different languages and choose the correct one at run-time.

The complete architecture of a generated VR-shop is given in figure 3. To make the VR shop available through a web browser, an applet has been created. It is embedded inside an html page along with a plug-in player like Cortona [32]. The applet communicates with the plug-in via EAI [29] and is also capable to communicate with a search engine (see section 3) developed for using the semantic information stored in the MPEG-7. It also communicates with the web services, which take care of the shopping tasks.

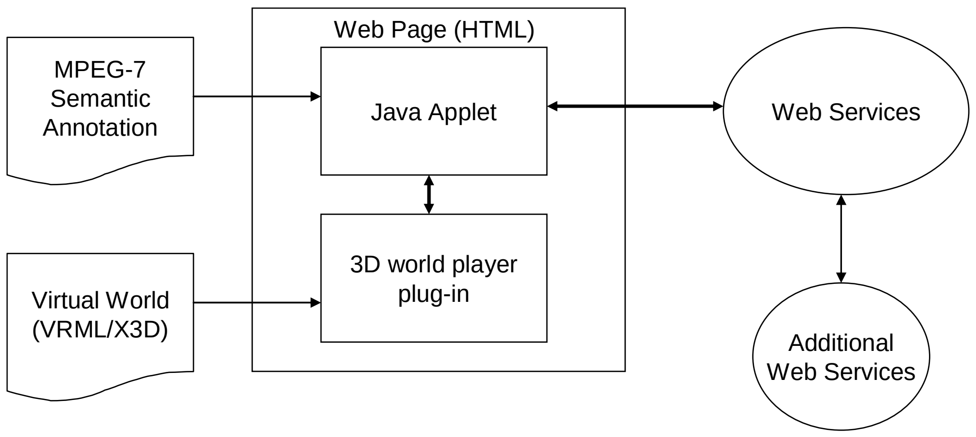


Figure 3. Architecture of a VR-Shop

4 Searching in the VR-Shop

Because usability will be crucial for the success of a VR-shop, it is important to provide alternatives for exploring the shop or looking for products through walking. One way to enhance usability is by providing a search engine. Because, all specifications were made at a conceptual level and using domain knowledge, we can exploit this semantic information by offering a **semantic** search engine. Indeed, we can allow the end-user to formulate semantic queries (based e.g., on the product information provided during design) in order to locate products quickly and this without the need to rely on any kind of Artificial Intelligence (as done in some VR search engines). This is possible because the semantic information explicitly provided during design as well as semantic information derived (from the product type hierarchy and the rules, as well as from positioning and lay-out information) is stored as semantic annotations in the MPEG-7 file. This means that the properties specified for the different products, the spatial information about where products are located, and the annotations provided by the rules can be combined with logical operators to formulate queries such as ‘find a non-expensive sofa for 3 or 4 persons, made of leather, and located in the living room area’ (see section 4 for the formulation of such a query). The search engine can also be used to jump to a specific area (defined as such in the shop) like the kitchen area. In addition, because all the product information is available for each product, it can be obtained easily (e.g., by pointing to or clicking on the product[2]). Also the behavior and tasks associated with the product can be provided, e.g., the end-user can be informed that the Rolling Chair will rotate when clicking on it.

5 An Example: a VR Furniture Shop

This section presents a VR-shop elaborated with the prototype tool OntoWorld. It concerns a fictitious furniture shop. It should be taking into consideration that it concerns a prototype, built to show the feasibility of the approach and that the user interface provided is basic and can be enhanced a lot (see also further research). For this same reason, the applet for the regular user interface and the applet for the semantic search engine are not yet integrated. In figure 4, a view on our VR furniture shop is given. As it can be seen, the VR-shop contains walls and has products such as tables, lamps, chairs, beds, and so on. At the top of this figure, the GUI provided by means of the applet is visible. Currently, the window of this applet is horizontally divided into two panes. The top pane contains four columns. The first column provides the name of all products in the VR furniture Shop[3]. The second column provides the behaviors associated with each product. The third column gives the different shopping tasks that can be performed and the

[2] Specified by the designer when defining behavior for products

[3] Currently, also the shopping cart is listed here. From a user point of view this is strange. It will be adapted in the next version (see also further work).

fourth column is only relevant when a behavior or a shopping task is selected in the second or third column. The bottom pane provides feedback to the user.

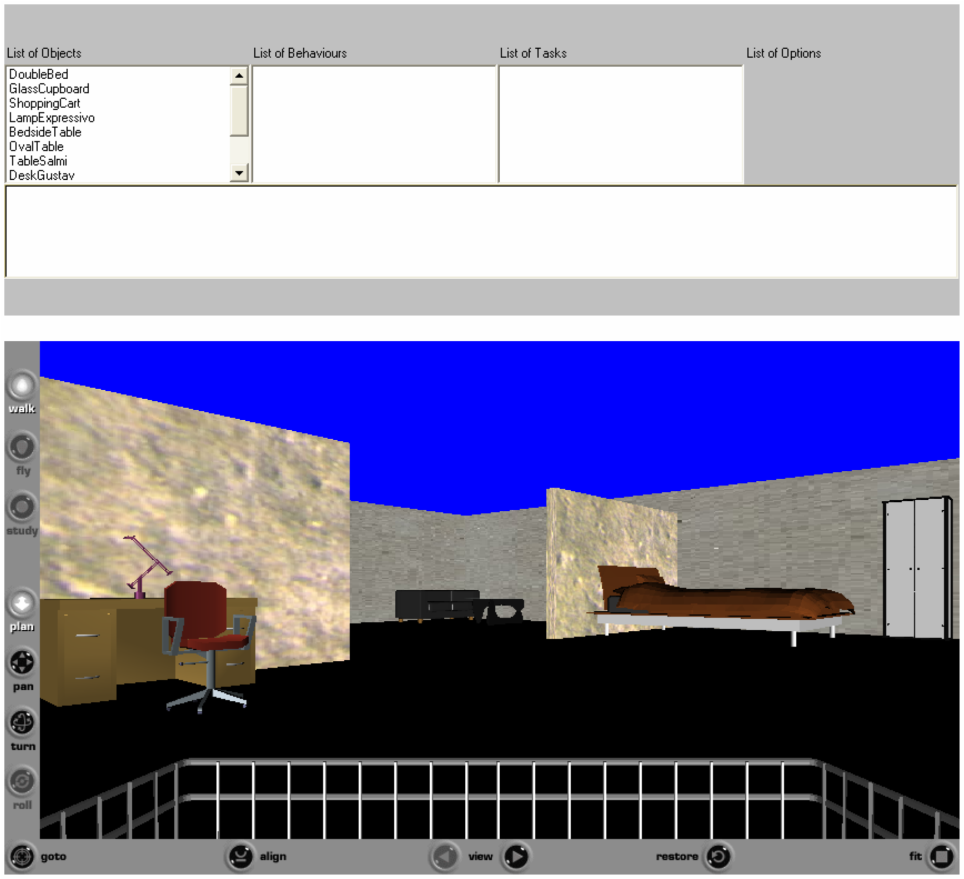


Figure 4. View of the VR furniture shop

5.1 Walking in the shop and obtaining information

The user can walk around in the shop (using the mouse or the interface provided by the VR-player). In this shop, the designer has opted for displaying the information associated with the product (and specified by means of the Product Ontology) when the user points to a product, e.g., in figure 5 the information related to the *RollingChair* is displayed inside the virtual world because the user has pointed to the *RollingChair*. At the same time, the product is highlighted in the first column of the GUI, the behaviors available for this product are shown in the second column, and its tasks in the third column.

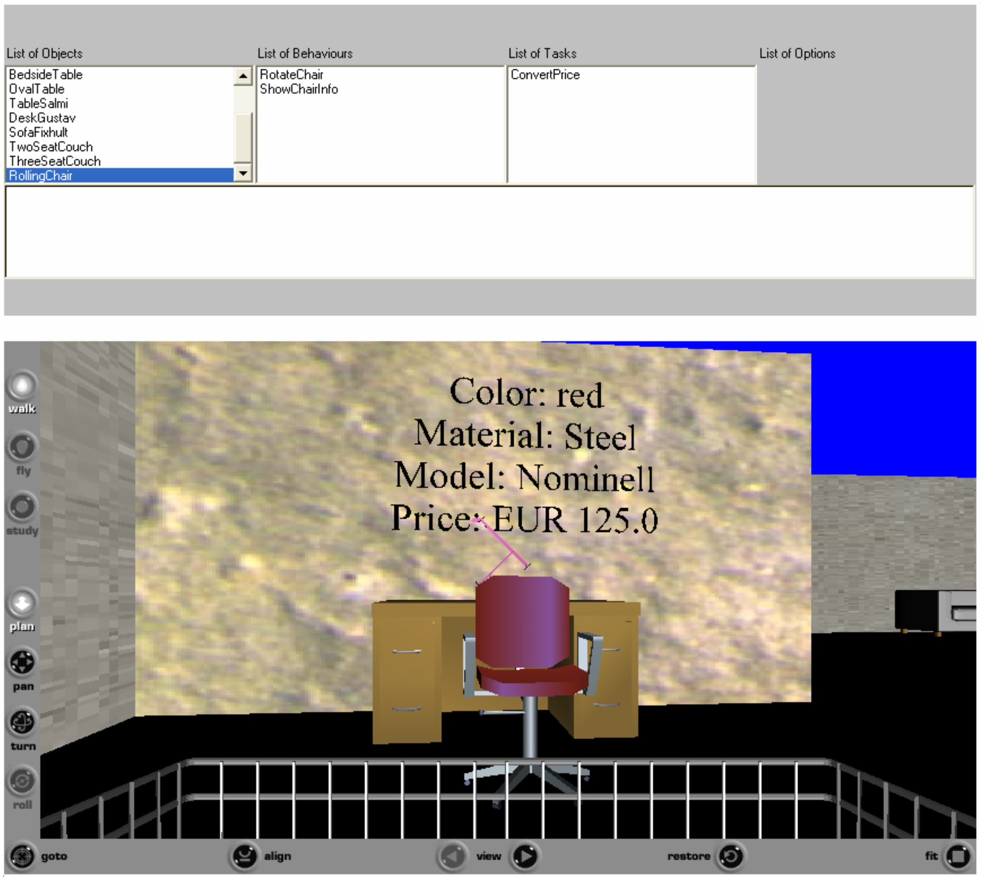
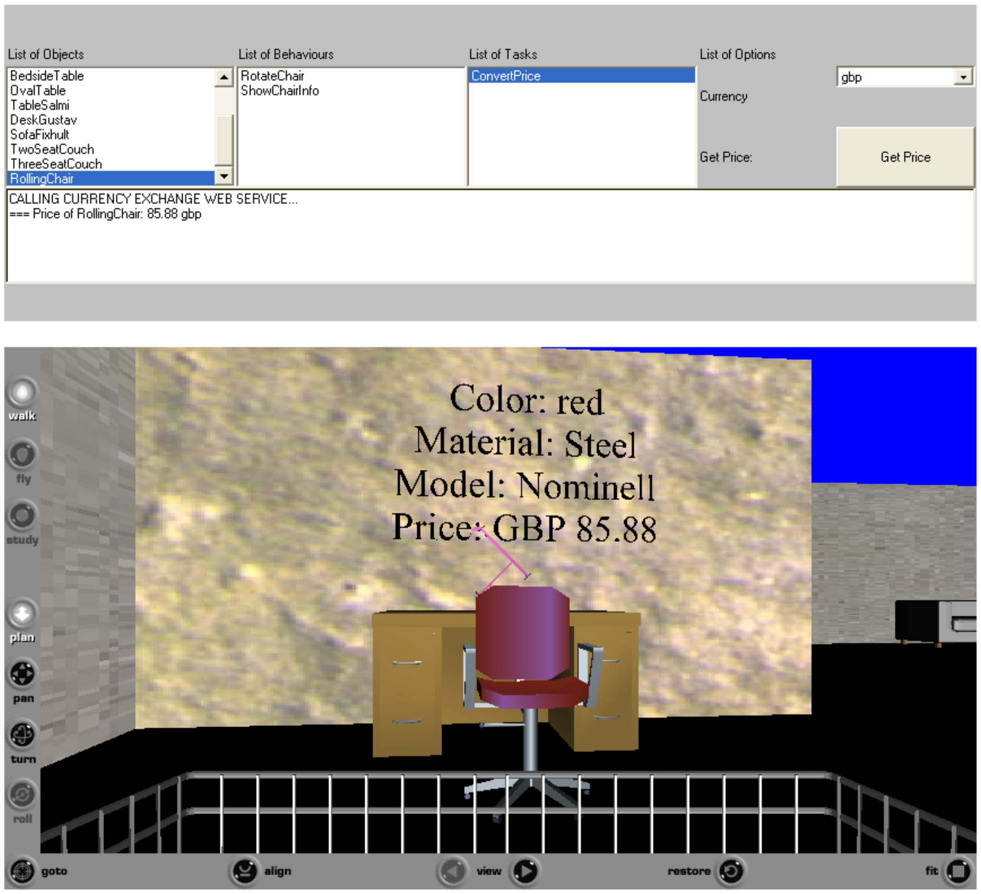


Figure 5. Information related to the RollingChair is displayed

At this moment, a task is available to convert the price into a different currency (ConvertPrice). When the user selects this task, a web service for converting the price into different currencies (see [33] for this web service) is called. Using this web service, the price of a product can be converted in any currency and in real-time.


List of Objects
BedsideTable
OvalTable
TableSalmi
DeskGustav
SofaFixhult
TwoSeatCouch
ThreeSeatCouch
RollingChair

List of Behaviours
RotateChair
ShowChairInfo

List of Tasks
ConvertPrice

List of Options
Currency: gbp
Get Price: Get Price

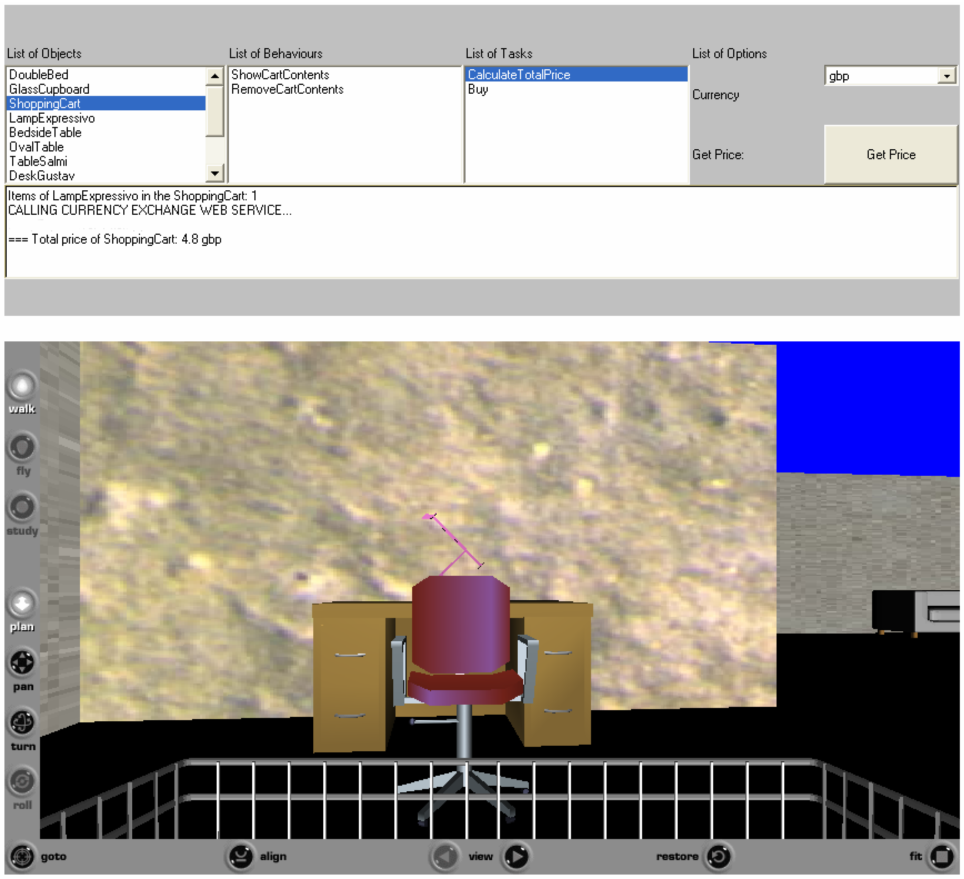
```
CALLING CURRENCY EXCHANGE WEB SERVICE...
=== Price of RollingChair: 85.88 gbp
```

Figure 6. Calling the task "ConvertPrice" for converting the price of the RollingChair in British Pounds

In Figure 6, you can see the result of converting the price of the *RollingChair* to British Pounds. As can be seen, also the price displayed inside the virtual world is updated.

5.2 Buying products

To buy products, the user first needs to select those products and add them to the shopping cart (one by one). At any moment, the user can inspect the content of his shopping cart. Currently, this has to be done using the GUI. To do this, the user should select the *ShoppingCart* object in the GUI (first column). In the feedback pane, the products in the shopping cart will be listed. If the user wants to know the total amount for these products, he should select the task CalculateTotalAmount, specify the currency in the fourth column and the result will be displayed in the feedback pane (see figure 7).


List of Objects

DoubleBed
GlassCupboard
ShoppingCart
LampExpressivo
BedsideTable
OvalTable
TableSalmi
DeskGustav

List of Behaviours

ShowCartContents
RemoveCartContents

List of Tasks

CalculateTotalPrice
Buy

List of Options

Currency: gbp

Get Price: Get Price

```
Items of LampExpressivo in the ShoppingCart: 1
CALLING CURRENCY EXCHANGE WEB SERVICE...

=== Total price of ShoppingCart: 4.8 gbp
```

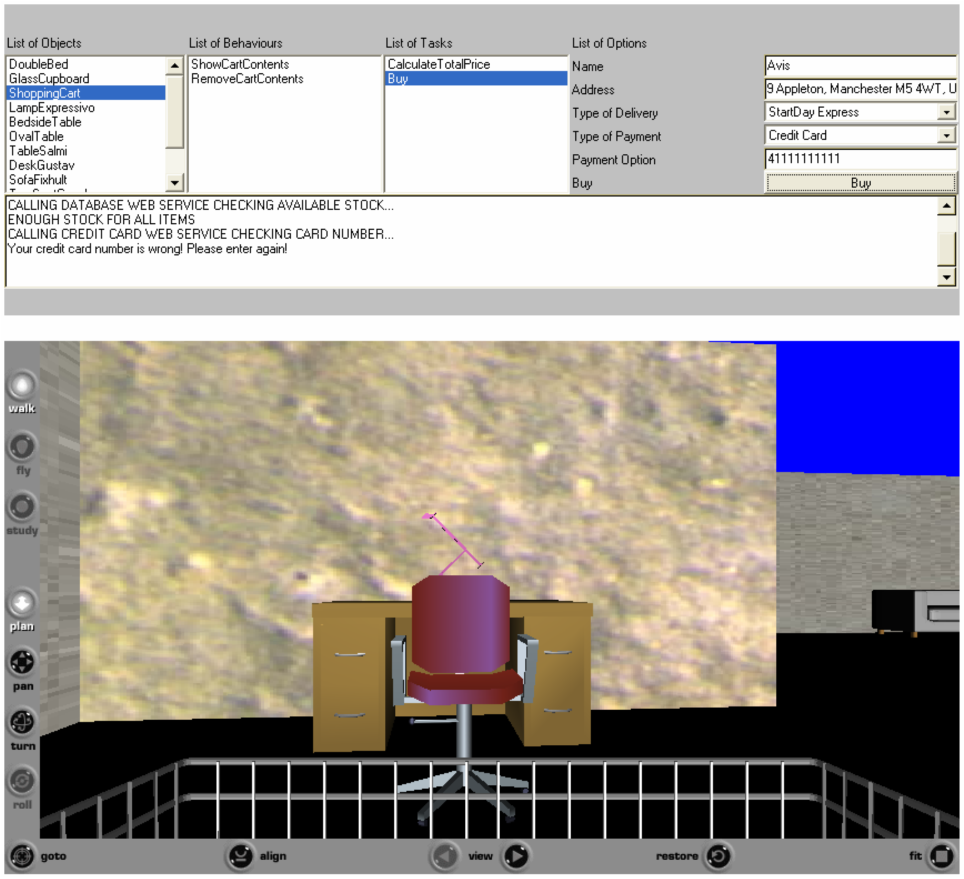
Figure 7.Calling the task “CalculateTotalAmount” to know the total amount for the products in the ShoppingCart

Once the user has decided to buy the products in his shopping cart, he can invoke the shopping task ‘Buy’ by selecting this task in the GUI. The user will enter his name and address. Furthermore, in this shop, this task allows for a number of options:

(1) The user can select the type of delivery: ‘Start Day Express’ or ‘Mid Day Daily Express’;

(2) The user can select the payment method: credit card or bank transfer. When a credit card is used, a web service (see [34] for this web service) is called to

check the validity of the credit card. In figure 8, we can see that the web service detected that the credit card number entered was invalid.

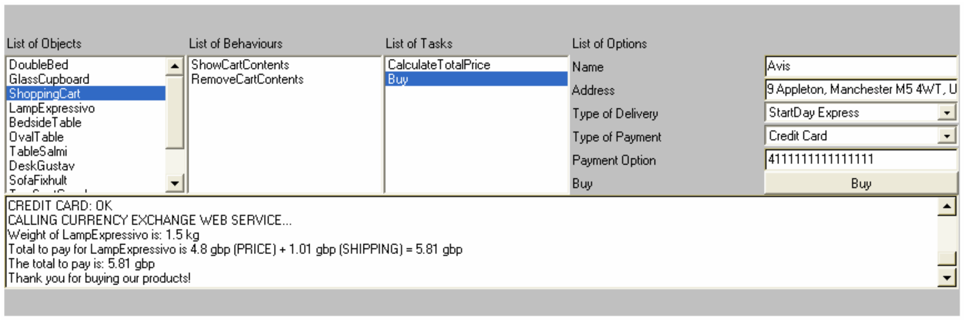


Name	Avis
Address	9 Appleton, Manchester M5 4WT, U
Type of Delivery	StartDay Express
Type of Payment	Credit Card
Payment Option	41111111111
Buy	Buy

```
CALLING DATABASE WEB SERVICE CHECKING AVAILABLE STOCK...
ENOUGH STOCK FOR ALL ITEMS
CALLING CREDIT CARD WEB SERVICE CHECKING CARD NUMBER...
Your credit card number is wrong! Please enter again!
```

Figure 8. During the buying task operation, the credit card number is being checked. Here, the credit card number is invalid.

In figure 9, a valid credit number was entered and therefore, the buying task could be performed (see the feedback pane). Note that the buying task also checks the stock to verify if all (and enough of) the products are available. This is also done by means of a web service. If (some) of the products are not available the user can order them. The registration of a sale or an order, and the update of the stock are also done by means of web services.



Name	Avis
Address	9 Appleton, Manchester M5 4WT, U
Type of Delivery	StartDay Express
Type of Payment	Credit Card
Payment Option	4111111111111111
Buy	Buy

```
CREDIT CARD: OK
CALLING CURRENCY EXCHANGE WEB SERVICE...
Weight of LampExpressivo is: 1.5 kg
Total to pay for LampExpressivo is 4.8 gbp (PRICE) + 1.01 gbp (SHIPPING) = 5.81 gbp
The total to pay is: 5.81 gbp
Thank you for buying our products!
```

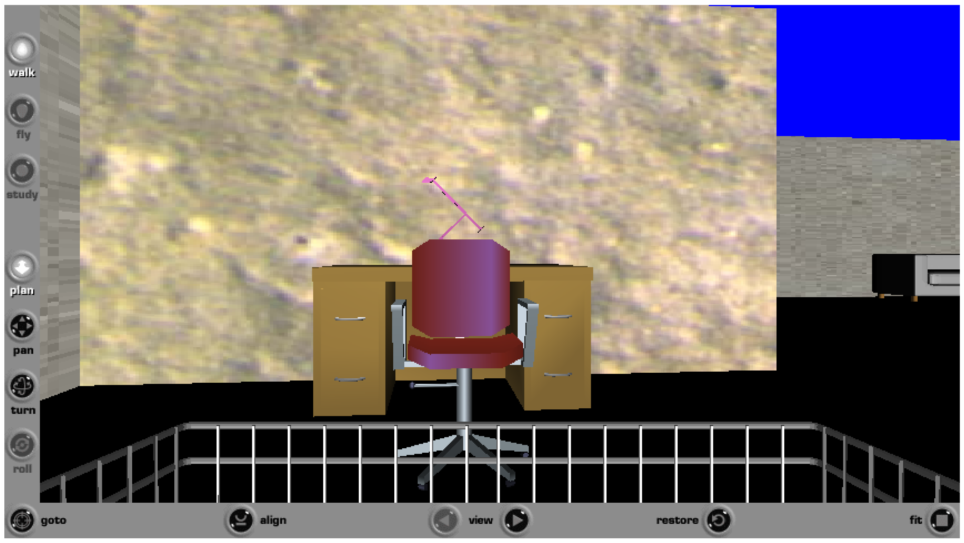
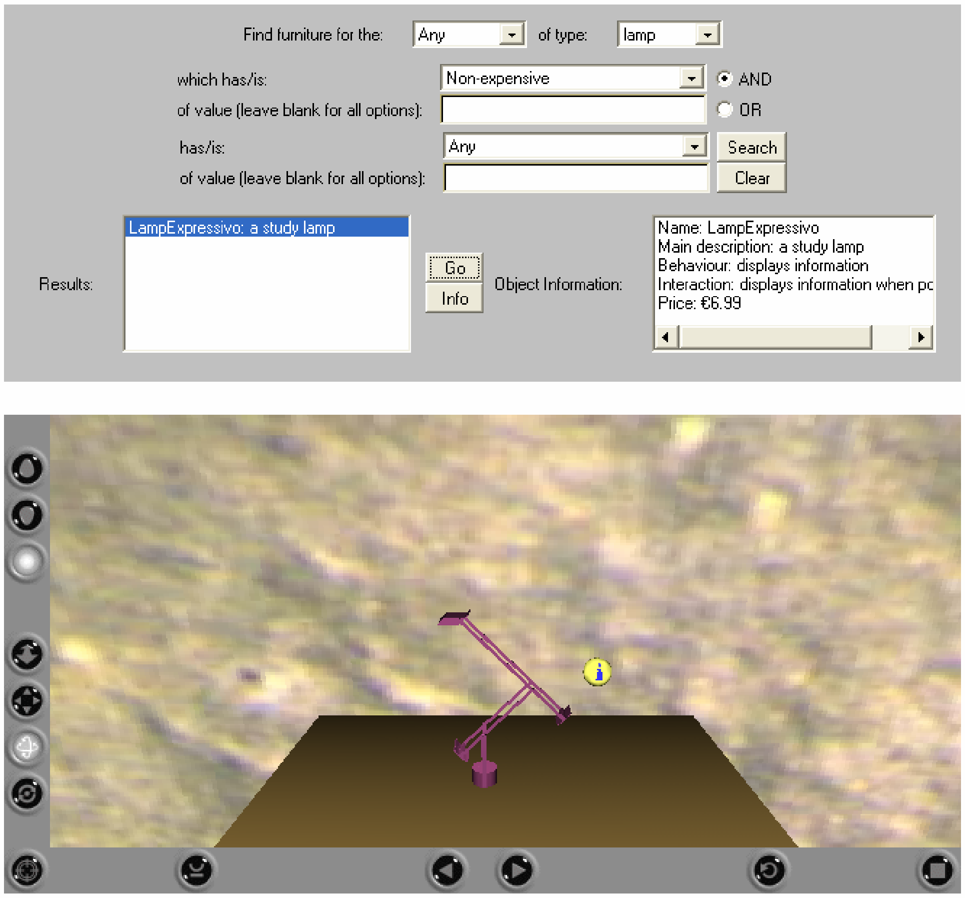


Figure 9. The credit card entered is valid; the buy task was successfully completed.

5.3 Using the semantic search engine

In figure 10, the GUI of the semantic search engine is shown. It allows the user to specify queries with the logic operators ‘AND’ and ‘OR’. The products that match the query are displayed in the ‘results’ list. The user can select a product from this list. Using the ‘Info’ button, the semantic information associated with the selected item will be displayed in the list on the right. Using the ‘Go’ button, the user will be transported inside the VR furniture shop to the selected product. Instead of using the ‘Info’ button, the user can also obtain the product information by pointing to the product once he was transported to the place in the shop where the product is displayed.


Find furniture for the: Any of type: lamp

which has/is: Non-expensive [✓ AND] OR

of value (leave blank for all options):

has/is: Any

of value (leave blank for all options):

Results: LampExpressivo: a study lamp

Object Information:

Name: LampExpressivo
Main description: a study lamp
Behaviour: displays information
Interaction: displays information when pc
Price: €6.99

Figure 10. Finding a non-expensive lamp

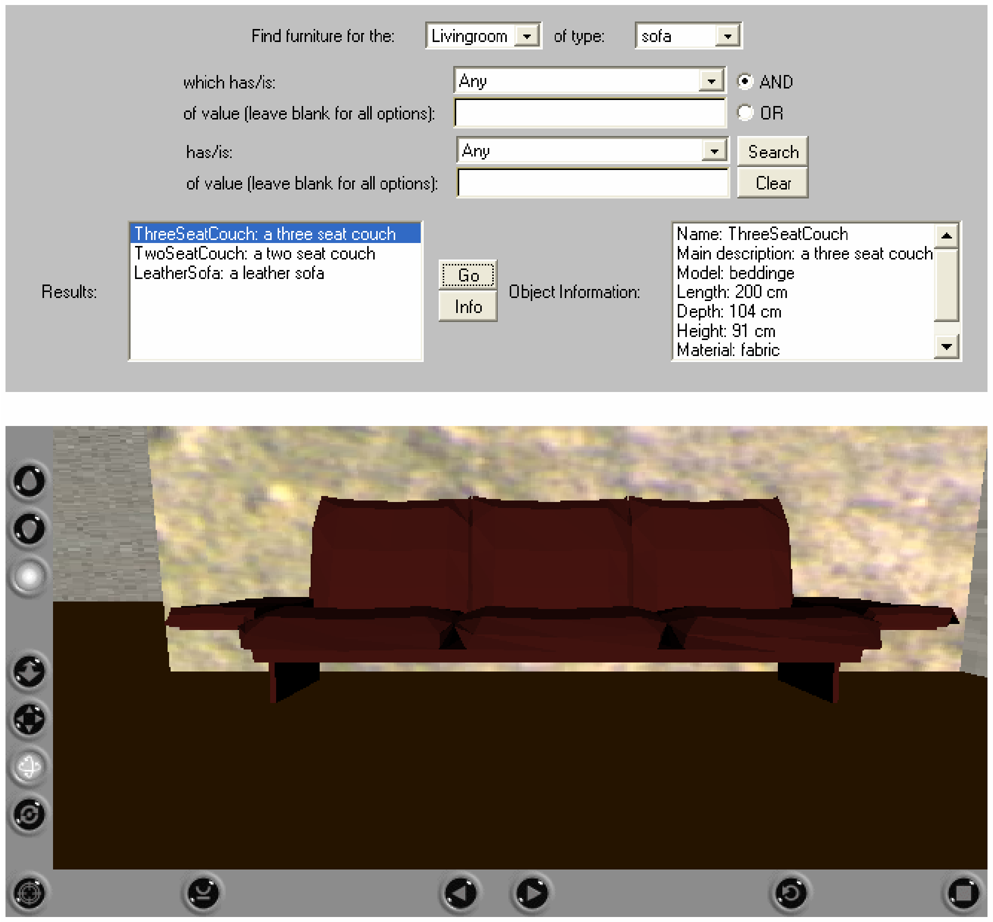
The search engine is illustrated with two examples:

- Finding a non-expensive lamp

  In figure 10, the user has specified a query to find a non-expensive lamp. Here, only one product 'LampExpressivo' is matching the user's query.

- Finding products in a particular area of the shop

  During design, our VR furniture shop was subdivided into a number of sections or area's (like in a real IKEA® shop): an area with furniture for the study room (like desks and rolling chairs), an area containing furniture for the living room (like sofas), an area with furniture for the bedroom, and so on. All of these areas have been stored in the MPEG-7 and therefore, the user can use the semantic search engine to be transported to such a specific area in the VR furniture shop. In figure 11, we can see that the user has formulated a query to see all the sofas in the Livingroom area. The result is shown in the Result-list of the GUI: 3 sofas were found. The user has selected the ThreeSeatCouch and is then transported (by using the ‘Go’ button) to this sofa.



Find furniture for the: Livingroom of type: sofa

which has/is: Any [✓ AND] OR
of value (leave blank for all options):

has/is: Any
of value (leave blank for all options):

Results:
ThreeSeatCouch: a three seat couch
TwoSeatCouch: a two seat couch
LeatherSofa: a leather sofa

Object Information:
Name: ThreeSeatCouch
Main description: a three seat couch
Model: beddinge
Length: 200 cm
Depth: 104 cm
Height: 91 cm
Material: fabric

Figure 11. Finding products by means of their location in the VR furniture shop

These two examples show that the semantic information can be exploited and used for enhancing the usability e.g., by means of a semantic search engine.

6 Discussion & Further Research

In the introduction we argued that an appropriate approach for designing VR-shops was needed that (1) is suitable for non-VR skilled people; (2) provides short development times; (3) results in systems that are easy to maintain and adapt; and (4) make provision for good usability.

In this section, we first discuss our approach with regard to these requirements:

(1) Suitable for non-VR skilled people

The use of pre-defined ontologies combined with high-level modeling concepts to compose the VR-shop and position products in it provides a development environment for which one does not need to be skilled in VR. The VR-shop can be specified using the terminology of a shop builder (given by means of the VR-Shop Ontology and the Shopping Task Ontology) and the product terminology used by the shop owner (specified by means of the Product Ontology) and without having to know much about VR-technology and about low-level implementation issues.

(2) Provides short development times

Next to the fact that a lot of concepts and functionality has been pre-defined by means of the VR-Shop Ontology (and associated mappings) and the Shopping Task Ontology, the actual code for the virtual world is generated from the specification. All this provides a considerable gain of time.

(3) Results in systems that are easy to maintain and adapt

Adding a new product to the VR-Shop, is quite simple: first a new product type must be specified in the Product Ontology and (at least one) instance of this product type should be positioned in the shop, next the file containing the 3D model of the product should be specified in the mapping and then the code should be re-generated. To remove a product, the product only has to be removed from the Product Ontology and the code can be re-generated. Similar, for updating product information, only the Product Ontology needs to be updated and code can be re-generated.

Adding, adapting or removing functionality is easy because it usually only involves attaching or detaching available tasks and/or adapting the task mapping, which means replacing one web service by another.

To rearrange products in the shop, only the spatial relations used to position the products should be adapted. For example, if the *Lamp* was placed ***OnTopOf*** the *Desk* and we would now like to place it on top of the table, this specification should be changed into '*Lamp* ***OnTopOf*** *Table*'. We do not have to deal with fixed coordinates anymore.

Changing the infrastructure of the shop should be done in the same way as defining it.

(4) Make provision for good usability

The approach allows exploiting the semantic information provided explicitly as well as implicitly during design to enhance the VR-application with semantic annotations. The semantic annotations are generated automatically from the specifications without additional cost. This means that semantic properties for concepts, instances and behaviors as well as the spatial positioning and orientation of the objects are automatically provided in the description by means of annotations and can be used to enhance the usability of the VR-shop. These semantic annotations can be exploited by many different applications or for many different purposes, e.g., a semantic search engine (as illustrated) or a Google-type of search engine to find VR-shops (on the Web) that satisfy certain semantic criteria; the semantic information may also be useful when different shops need to cooperate.

Although, our approach was developed to ease the development of VR-shops, it also offers some added value to the end-user. As indicated in (4), the usability of shops developed with this approach will be higher, mainly due to the incorporation of semantics. Firstly, different sorts of product information can be provided to the end-user. Secondly, a semantic search engine is incorporated that end-user can apply to search for products based on their semantic properties or to navigate quickly to certain locations (also identified by means of semantic properties). And thirdly, at any time and for each product, the end-user can ask what kind of interactions and tasks this object supports. Furthermore, the semantic information can be translated in different languages so that the end-user can express his queries (or see information related to the products) in his native language.

Although the current approach is already quite powerful, further research is still necessary. It may be interesting to investigate how the shop can be adapted without having to re-generate the complete shop. We should also investigate how to deal with large shops that would take too long to download completely over the Internet when the shop is requested. It would also be interesting to study the accessibility of the shop for software agents. Also localization (towards different markets) and personalization of the VR-shop (to the preference of the individual users) are interesting research issues. Furthermore, from a usability point of view at lot can be improved and added e.g., spotlights and sound could be added for instance, to draw attention to promotional products. We would also like to investigate the possibility to eliminate the need for a classical GUI (now provided by an applet) by integrating it completely in the virtual world, by preference using 3D interface elements. Also different types of shops can be studied. Different strategies for displaying and organizing products may be needed for different kind of shops e.g., in a furniture shop it is sufficient to show only one instance of each product, however in a food store it would not be a good idea to show only one can of cola or one orange.

At the time of writing this paper, the prototype is re-built. It has changed in several respects. First of all, a better and more appealing GUI is designed. Also the implementation architecture is changed. The X3D loader of Xj3D [35] is used to load the complete scene into Java3D [36]. All the objects in the scene are linked to OdeJava [37] objects in order to give the user the feeling of being in a real physical environment. The application is loaded in an ordinary Web browser via WebStart [38]. In such a way, no applications or plug-ins need to be installed by the user in order to visit the shop.

Furthermore, we are investigating how to adapt the system such that existing product catalogs can be used. The simplest solution here is to provide an import facility to convert the existing product catalog into the format required by the Product Ontology.

Finally, we are also searching for a way to include marketing strategies into the design process of the VR-shop. These strategies will then allow to automatically place the products according to some well-specified rules to give a more profitable result.

7 Conclusion

This paper has presented the Shop-WISE approach, an approach to develop VR-shops for the Web. The approach specially targets developers not skilled in VR. The VR-shop is specified at a conceptual level and using the terminology of a shop builder and the shop owner. The actual VR-shop can be generated from the conceptual specifications. The approach provides the flexibility necessary for VR-shops. Products can be easily added, updated or removed because they are maintained separately and described by means of the Product Ontology. The layout of the shop can easily be defined or changed because common VR-shop infrastructure concepts are pre-defined and the virtual world itself can be specified by means of high-level and intuitive spatial and orientation relations. Also functionality is easily added and changed due to the use of web services. Furthermore, this paper has also explained how the usability of a VR-shop can be enhanced by allowing exploiting at run time the semantic information collected during design time. To illustrate the approach, a simple furniture shop has been built.

8 Acknowledgements

This research is carried out in the context of the Ontobasis and the VR-DeMo projects; both projects are funded by the 'Institute for the Promotion of Innovation by Science and Technology in Flanders' (IWT). It is also partially funded by the FWO (Fund of Scientific Research – Flanders).

9 References

1. --, Amazon: http://www.amazon.com. Accesses 10 January 2005
2. --, Dell: http://www.dell.com. Accessed 10 January 2005
3. --, IKEA: http://www.ikea.com//. Accessed 10 January 2005
4. Robles A, Molina, J P, López Jaquero V, and García A S (2005) Even Better Than Reality: The Development of a 3-D Online Store that Adapts to Every User and Every Platform. *HCI International 2005*, Las Vegas, Nevada, USA, July, 2005. Volume 7 - Universal Access in HCI: Exploring New Interaction Environments. 10-20
5. Haubl G and Figueroa P, (2002) Interactive 3D Presentations and Buyer Behaviors. In: Extended abstracts of the 2002 Conference on Human Factors in Computing Systems, CHI 2002, ACM, Minneapolis, Minnesota, USA, April 20-25, 744-745
6. Chittaro L and Ranon R (2002) New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites. In: AVI 2002: 5th International Conference on Advanced Visual Interfaces, ACM Press. 308-305
7. Domingue J, Stutt J, Martins M, Tan J, Petursson H and Motta E (2003) Supporting Online Shopping through a Combination of Ontologies and Interface Metaphors. International Journal of Human-Computer Studies. 59: 699-723
8. Sama A, Montrucchio B, Montushi P and Demartini C (2001) 3D-dvshop : a 3D dynamic virtual shop. In: The 6th Eurographics Workshop on Multimedia. 23-32
9. Mass Y, Herzberg A (1999) VRCommerce - Electronic Commerce in Virtual Reality. In: The 1st ACM Conference on Electronic Commerce, November 03-05, Denver, Colorado, United States. 103-109

10. Chittaro L, Ranon R (2000) Virtual Reality stores for 1-to-1 e-commerce. In: The CHI2000 Workshop on Designing Interactive Systems for 1-to-1 E-Commerce, The Hague, The Netherlands.
11. Chittaro L and Ranon R (2000) Adding adaptive features to Virtual Reality interfaces for e-commerce. In: the AH2000: International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Lecture Notes in Computer Science 1892, Springer-Verlag, Berlin. 85-96
12. Ballegooij A and Eliens A (2001) Navigation by query in virtual worlds, Virtual Reality Modeling Language Symposium. In:The sixth international conference on 3D Web technology, Paderbon (Germany), ACM Press. 77-83
13. Abaci T, Mortara M, Patane G, Spagnuolo M, Vexo F and Thalmann D (2005) Bridging Geometry and Semantics for Object manipulation and Grasping. In: The Workshop towards Semantic Virtual Environment (SVE), Switzerland. 110-119
14. Gutierrez M, Vexo F and Thalmann D (2005) Semantics-based representation of virtual environments. International Journal of Computer Applications in Technology. 23(2): 229-238
15. Bille W, Pellens B, Kleinermann F and De Troyer O (2004) Intelligent Modelling of Virtual Worlds Using Domain Ontologies. In: The Workshop of Intelligent Computing (WIC), Mexico City, Mexico. 272-279
16. Bille W, De Troyer O, Kleinermann F, Pellens B and Romero R (2004) Using Ontologies to build Virtual Worlds for the Web. In: The IADIS International WWW/Internet 2004 Conference, Madrid, Spain. 683-690
17. Bille W, De Troyer O, Pellens B and Kleinermann F (2005) Conceptual Modeling of Articulated Bodies in Virtual Environments. In: The 11th International Conference on Virtual Systems and Multimedia. Publ. Archaeolingua, Gent, Belgium. 17-26
18. Pellens B, De Troyer O, Bille W and Kleinermann F (2005) Conceptual Modeling of Object Behavior in a Virtual Environment. In: 3rd International Virtual Concept Conference, Biarritz, France. Springer-Verlag. 93-94 + CD-ROM
19. Pellens B, De Troyer O, Bille W, Kleinermann F and Romero R (2005) An Ontology-Driven Approach for Modeling Behavior in Virtual Environments. In:On the Move to Meaningful Internet Systems 2005: OTM Workshops: OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005, Agia Napa, Cyprus, October 31 - November 4. 1215-1224
20. Gruber T R (1993) A translation approach to portable ontologies. Elsevier. 5(2): 199-220
21. --, OWL: http://www.w3.org/2004/OWL. Accessed 10 June 2005
22. --, DAML+OIL: http://www.daml.org/2000/12/daml+oil. Accessed 10 June 2005
23. --, DAML Project: http://www.ksl.stanford.edu/projects/DAML. Accessed 10 June 2005
24. --, OIL language group: http://www.ontoknowledge.org/oil/. Accessed 16 January 2006
25. --, http://www.cs.man.ac.uk/~horrocks/DAML/Rules. Accessed 10 June 2005
26. --, http://www3.org/Submission/2004/SUBM-SWRL-20040521. Accessed 10 June 2005
27. --, Web services: http://www.w3.org/2002/ws/. Accessed 16 October 2005
28. Walsh A E and Sevenier M (2005) Core Web3D. Prentice Hall, Upper Saddle River: USA. ISBN: 0130857289
29. Hartman J and Wernecke J (1998) The VRML 2.0 Handbook. Addisson-Wesley Publishing. ISBN: 0-201-47944-3
30. Salembier P and Smith J R (2001) MPEG-7 Multimedia Description Schemes. IEEE transactions on circuits and systems for video technology. 11(6): 748-759
31. Ucelli G, De Amicis R, Conti G (2005) Shape Semantics and Content Management for industrial Design and virtual Styling. In: The Workshop towards Semantic Virtual Environment (SVE), Switzerland. 127-137
32. --, Cortona: http://www.parallelgraphics.com. Accessed 10 June 2005
33. --, Currency Exchange: http://www.web servicex.net. Accessed 4 January 2006
34. --, Credit Card Checking: http://www.cdyne.com/account. Accessed 4 January 2006
35. --, Xj3D: http://www.xj3d.org. Accessed 10th December 2005
36. --, Java3D: http://java.sun.com/products/java-media/3D. Accessed 10th December 2005
37. --, odejava: https://odejava.dev.java.net. Accessed 10th December 2005
38. --, WebStart: http://java.sun.com/products/javawebstart. Accessed 10th December 2005