Vrije Universiteit Brussel
Faculty of Science
Department of Computer Science

GENERATING VIRTUAL REALITY SHOPS FOR E-COMMERCE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF APPLIED COMPUTER SCIENCE
TO ACHIEVE THE DEGREE OF LICENTIATE

Vladimir Fomenko
May 2006

Promotor: Prof. Dr. Olga De Troyer
Supervisor: Dr. Frederic Kleinermann

Acknowledgments

I would like to thank all the people who took the time to answer my numerous questions, helping me define my problems and bringing me on the right path toward their solutions. More concretely, I would like to thank my supervisor, Dr. Frederic Kleinermann and my promoter, Prof. Dr. Olga De Troyer for being there when I needed them, for correcting my errors and most important, for making me "a better person". Also, I'd like to thank Bram Pellens, for his many programming tips and support.

Abstract

During the last years, E-Commerce, and online shopping in particular, has known a strong popularity and rapid growth. At the same time, computer hardware, software, and Internet broadband access has allowed to spread the use of Virtual Reality (VR) online. As innovation is an important aspect on the Web, Virtual Reality is a promising technology. By attracting the user both visually and in terms of intuitiveness, VR may seriously increase online retail sales. However, development cost and development time stays high. Moreover, the complexity of the development of VR application requires the need to involve VR experts and allows only partial involvement of domain experts.

In this thesis, we present SHOP-WISE, an approach to develop VR online shops that permits domain experts to be much more in control, and OntoShop, the supporting tool. Using high-level concepts modeling and by requiring only a minimum amount of technically "difficult" work, a complete VR shop can be (semi) automatically generated. The generated 3D environments are completely functional and ready for the customers to be used. The VR shop can be visited and products can have different behaviors (like turning). Moreover, shopping tasks like "buy" or "calculate shipping cost" can be executed.

Based on ethnographical studies, an intuitive graphical user interface for the OntoShop tool has been created for the shop designers. Typical shopping tasks, behaviors and infrastructure concepts (like walls and racks) have been pre-defined. In this way, the work done by the domain expert is reduced to a minimum.

List of Figures

List of Symbols

Symbol	Definition
VR	Virtual Reality: An artificial environment created with computer hardware and software and presented to the user in such a way that it appears and feels like a real environment.
VRML	Virtual Reality Modeling Language. An open, extensible, industry-standard scene description language for 3-D scenes, or worlds, on the Internet. With VRML and dedicated software tools, one can create and view distributed, interactive 3-D worlds that include text, images, animation, sound, music and video.
X3D	X3D is the ISO standard for real time 3D graphics, the successor of VRML. X3D features extensions to VRML (e.g. Humanoid Animation, Nurbs, GeoVRML etc.), the ability to encode the scene using an XML syntax as well as the Open Inventor-like syntax of VRML97, and enhanced application programmer interfaces (APIs).
WEB3D	Web3D is the concept of interactive 3D content as displayed via the World Wide Web with the overall goal of making the users experience better.
EFTPOS	Electronic Funds Transfer at Point of Sale: A system that allows funds to be moved automatically from a buyer's account to a seller's, the transfer taking place at the time of the transaction.
MPEG-7	A set of standardized tools to describe multimedia resources, including still images, moving images, audio, etc. Includes textual and non textual indexing and description. Defined as an XML schema.
XML	Extensible Markup Language. A W3C initiative that allows information and services to be encoded with meaningful structure and semantics that computers and humans can understand. XML is great for information exchange, and can easily be extended to include user-specified and industry-specified tags.

OWL	OWL is an acronym for Web Ontology Language, a markup language for publishing and sharing data using ontologies on the Internet. OWL is a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language. Together with RDF it is one of the pillars of the Semantic Web.
OO/OOP	In computer science, object-orientation/object-oriented programming is a computer programming paradigm. Key characteristics of object-oriented systems are encapsulation, inheritance and polymorphism.
VAT	Value added tax. Sales tax levied on the sale of goods and services.
CORBA	Common Object Request Broker Architecture is a language-independent object model and specification for a distributed applications development environment.
RMI	Remote Method Invocation provides a means for invoking the methods of remote Java objects.
DCOM	Distributed Component Object Model is a Microsoft proprietary technology for software components distributed across several networked computers.
UDDI	Universal Description Discovery and Integration is a directory model for web services. UDDI is a specification for maintaining standardized directories of information about web services, recording their capabilities, location and requirements in a universally recognized format. Seen (with SOAP and WSDL) as one of the three foundation standards of web services.
SOAP	Simple Object Access Protocol is a lightweight XML based protocol used for invoking web services and exchanging structured data and type information on the Web.
WSDL	The Web Services Description Language is an XML format published for describing Web services.
W3C	World Wide Web consortium, whose mission it is to create standards and specifications for the World Wide Web.

Contents

Chapter 1

Introduction

1.1 Thesis motivation

The topic of the thesis is inspired by the following observations: the growing success of E-Commerce and in particular of E-shops combined with a high competition for online customers, and the increasing interest for 3D applications (Virtual Reality) such as games. This has resulted in the idea to use Virtual Reality to develop online shops. This leads to online virtual shops that are more attractive and give better revenues than the traditional E-shops currently found on the Web.

1.1.1 E-commerce development

E-commerce activities have known, since many years already, a strong popularity. Online retail sales alone have grown with a double digit percentage every year and the expected growth till 2008 is expected to be as high as 19% [12].

While more than half of the US online population is buying at least once a year online [12], spending on the average US$ 784 [24], and with a total online retail market said to be representing US $329 billion in 2010 [11], no one should underestimate the need to attract customers online.

However, there is another important statistic fact we may not forget, namely: 65% of Web users won't patronize a poorly designed site, even not that of a favorite brand. 30% reported that Web site design is more important than a great product as even rock-bottom prices only persuaded 4% to shop on a poorly designed Web site. What's worse is that nearly 30% stop buying from their favorite offline store if their online experience is poor [12].

This actually means that whenever a potential online buyer goes through an improved online experience, there is (much) more chance he will buy something.

This leads us to think about new ways to attract customers. Some innovation is needed for improving online shopping's attractiveness and for helping to increase profit. One such possible way to attract customers is to build a Virtual Environment, more precisely a virtual E-shop [20], where customers can do online shopping in a three-dimensional world. This may offer an original, new and attractive experience to the buyer and may provide added value over traditional E-shops by making use of the opportunities offered by the technology of Virtual Reality (VR).

1.1.2 VR Development

Building an E-shop using VR is quite an undertaking: a virtual environment must be built containing 3D representations of the products; behaviors and user interactions must be provided for the products and/or environment; and different shopping tasks must be available. Similar as the design of Virtual Environments in general, it is a complex task that requires skilled people with considerable knowledge of VR technology and programming skills or knowledge about script languages. It also takes lots of time and has a high cost.

Given this situation, tools such as Sketchup [6] have already been created. Sketchup is a 3D modeling program that enables users to easily create and explore a 3D world. A screenshot of the Sketchup application and a model of the White House can be seen in figure 1. In the same manner, virtual E-shops are only feasible if a way can be found to simplify their development and make the development process also accessible to non VR-experts.

Moreover, we prefer a development process composed of a number of logical steps, i.e. we would like to provide a method for building virtual E-shops. Ideally, the steps in the method should correspond with how shops are built in real life. In this way it will be more intuitive for non-technical persons to create a virtual mall.

1.2 Thesis intent

The thesis' intent is to popularize virtual E-shops by providing an approach for building VR E-shops that is accessible for a broad public. The approach, called SHOP-WISE, is based on the VR-WISE approach developed at WISE to develop Virtual Environments in general. SHOP-WISE allows technically inexperienced people to

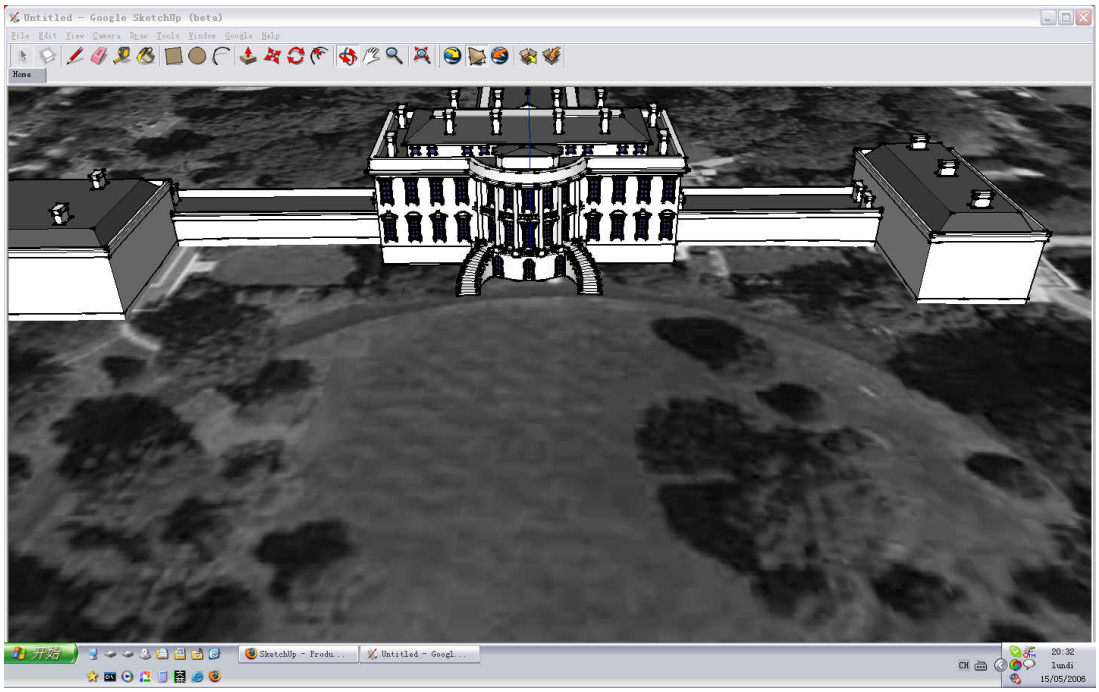

Figure 1.1: Easy 3D modeling with Sketchup

build a virtual shop and it will allow them to do so in an easy way. To support the approach a tool called OntoShop is developed.

Like the SHOP-WISE approach is based on the VR-WISE approach, the supporting tool OntoShop is based on OntoWorld, the tool that support VR-WISE and that is developed at the WISE Lab, in the Vrije Universiteit Brussel. OntoWorld's main idea is to let a person - this can be a domain expert, who is not necessarily technically skilled - describe what kind of objects he would like to have in the virtual world, where he would like to place them and what behaviors he would like them to have. Based on this description a virtual world can be generated (more details can be found in Chapter 2).

The work presented in this thesis is a customization and an extension of the VR-WISE approach. OntoShop is a customization of OntoWorld in the sense that the tool has been tailored especially for the design and generation of virtual shops. This means, that a lot of the functionality and features that are common in virtual shops are built-in and do not need to be specified anymore while developing a shop. This will result in a considerable gain of time. Also the development process has been adapted and customized towards the development of shops. Based on ethnographic

studies, we have built a tool that takes into account the way shop managers think, and how shops are built in reality.

To achieve this, it was also necessary to extend the functionality currently offered by OntoWorld. To support shopping activities, it was necessary to allow assigning non-VR tasks (such as buying a product) to VR-objects. This was not supported by OntoWorld.

1.3 Thesis structure

This document is structured as follows. In the next chapter (chapter 2), we will give more details about VR and E-Commerce, followed by a small historic overview of related works and backgrounds. An in-depth overview of OntoWorld and the VR-WISE approach, on which OntoShop and SHOP-WISE are based will be given and we will also explain some important concepts used further on in this thesis.

Chapter 3 will explain OntoShop's architecture from a conceptual point of view.

In chapter 4, we will explain how the customization and the extension of OntoWorld has been done to realize OntoShop, from a software engineering point of view.

A case study is given in chapter 5.

Finally, after a discussion about possible further research in chapter 6, we finish with the conclusion in chapter 7.

Chapter 2

Backgrounds

This thesis' work is a cross between two fields: E-Commerce and Virtual Reality. It is therefore important to give background on both disciplines.

2.1 E-Commerce

Apart from being very interested in sciences, the author of this thesis is also very interested in business, (E-)commerce and money in general. It is precisely the interest in online commercial activities that made him choose to work on this thesis' topic. For this reason, a somewhat more detailed introduction on E-Commerce will be given than is strictly needed. The author believes that this will make the reading of the remaining parts more interesting, providing insight in how important "virtual commerce" has become nowadays and just how useful - if not urgently needed - is the work made at the WISE Laboratory in the Vrije Universiteit Brussel.

2.1.1 Definition of E-Commerce

"E-Commerce is about the sale and purchase of goods or services by electronic means, particularly over the Internet" [22].

When talking about E-Commerce, most people primary think that this means shopping online. However, this activity actually represents only a small part of E-Commerce. The term of E-Commerce also refers to a wide range of electronically mediated commerce, which includes online marketing, brokering, auctions, EFTPOS, barcode verification, fax and e-mail. In addition, E-Commerce includes business-to-business transactions among big corporations over Electronic Data Interchange (EDI)

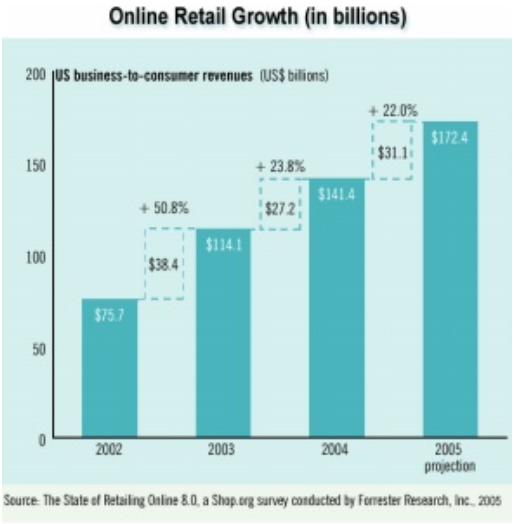

Figure 2.1: Online retail growth in the US (from [9]).

network. However, in this report, the term of E-Commerce refers to Internet commerce, and particularly to Web Commerce, online shopping and electronic business transactions over web sites.

2.1.2 Growth and popularity of E-Commerce

Today, it is obvious that the Web has become a significant marketplace for the buying and selling of products and services. Accordingly, E-Commerce and online retail have rapidly grown and increased in popularity. And due to a growing online population as well as to an increase in buyers' spending amount and a higher percentage of online shoppers, this trend will not stop in the near future.

Figure 2.1 shows some data about online retail revenues for the last years. It is estimated that approximately 142 million consumers or 65% of the online (US) population will have made a purchase online by 2007 [32] and that retail spending is to reach $329 billion by 2010 [29], accounting for 13% of all U.S. retail spending.

There are different reasons for the growth of E-Commerce. According to the survey of 1.200 online shoppers from Ernst & Young [28], the interviewees were asked the reasons for shopping online. Results are as follow:

- 64% of online shoppers said that they go shopping online because they want to save time.
- 56% of online shoppers said that malls or stores are too crowded.

- 52% of online shoppers find shopping online attractive because they can do it on more convenient hours.
- 37% of online shoppers said that is requires less driving and 16% decided to shop online because items cost less.

2.1.3 Benefits of VR E-Commerce

Chittaro and Ranon [26] indicate what benefits VR brings to E-Commerce. A VR store has some relevant advantages (if properly implemented):

- It is closer to the real world shopping experience, and thus more familiar to the buyer.
- It supports buyer's natural shopping actions such as walking, and looking around the store.
- It can satisfy emotional needs of buyers by providing a more immersive, interactive, and visually attractive experience.
- It can satisfy social needs of buyers by allowing them to meet and interact with people.

There are also case studies proving that interactive 3D graphics have the ability to increase the productivity of online businesses. Companies such as *Sharper Image Inc.* have seen their online number of visitor augment by 300% after puting three dimensional models for some of their products on their website. Visitors stayed 50% more time in the 3D area. This increase in visitors generated also a very significant increase in revenues, as in just one year, the profit went up to $30 million, compared to $4.9 million the year before.

Another reason for using Virtual Reality for E-Commerce is the high attraction of interactive 3D environment. Experiments concluded that interactive 3D graphics have attracted more customers on the Web. Haubl and Figueroa [20] at University of Alberta Edmonton conducted several experiments to examine the effects of interactive 3D product presentations on buyer behaviour.

The results show that with the availability of 3D product presentation, instead of still images, buyers tend to spend a greater amount of time viewing the products, and that there is a higher likelihood of purchase. Interestingly, interactive 3D product

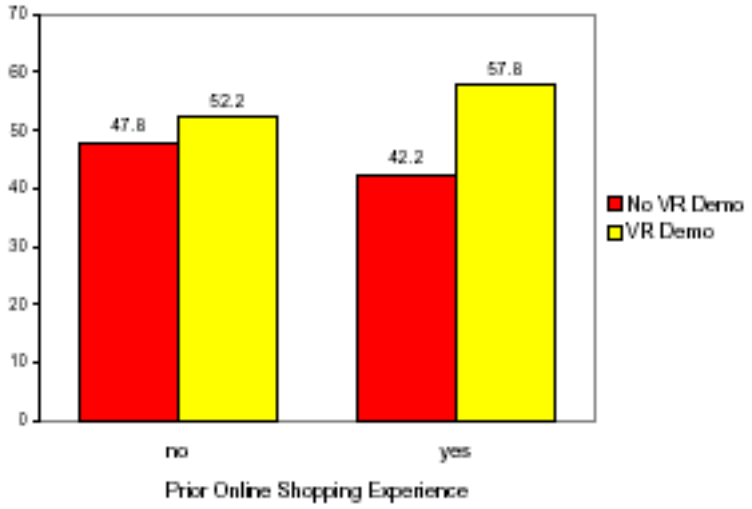

Figure 2.2: Prior online shopping experience and product choice share (from [20]).

presentation has a better effect on individuals with prior online shopping experience. The results of this experiment can be seen in Figure 2.2.

With the above facts and forecasts, it is clear that E-Commerce promises many beneficial opportunities. Therefore, the consideration to use new concepts to further enhance the possibilities of E-Commerce on the Web, e.g., Virtual Reality and rich-media content, needs to be researched.

2.2 VR-WISE

As stated earlier, our research is mainly based on previous work done by the WISE group which has developed an approach called the VR-WISE approach. This approach has been developed to deal with the following three main problems:

- The design phase in the development process of a VR application is usually an informal activity. Few formal techniques exist to support the VR design phase effectively. A systematic approach that uses the output of the design phase as input for the implementation does not exist.
- Developing VR is difficult and expensive.
- It is important to ensure that the domain expert stays involved in the different stages of the development of a VR application in order to provide the necessary

input and feedback.

Using a step-by-step method, VR-WISE allows developing virtual environments from explicit semantic descriptions such as knowledge bases or ontologies. Using this approach, it becomes unnecessary to have knowledge of specific VR programming languages or authoring tools to design 3D-worlds. Finally, it also allows the domain expert to model the world from his point of view, on a much higher abstract level.

We will explain VR-WISE briefly. Details can be found in [30].

2.2.1 Ontologies

VR-WISE relies heavily on ontologies. Simply stated, an ontology can be seen as an abstraction of a computer-based lexicon, thesaurus, glossary or some other type of structured vocabulary, suitably extended with knowledge about a given domain [29], [34]. A domain ontology can be considered to be a representation of a domain conceptualization describing possible concepts and relationships between these concepts.

2.2.2 Generating Virtual Reality from Ontologies

For developing Virtual Environments, the VR-WISE approach uses conceptual specifications (also called conceptual models). In VR-WISE, a conceptual specification is a high-level representation of the objects in the Virtual Environment, how they are related to each other, how they will behave and interact with both each other and with the user. Such a conceptual specification must be free from any implementation details and not influenced by the current technical limitations of the VR technology. The use of a conceptual model will improve the reusability, extensibility and modularity of the VR application [15].

As underlying representation formalism for the conceptual specifications, VR-WISE uses ontologies. As stated in [36], in this context, ontologies have the following advantages:

- *Grasping the knowledge of a domain*
  The domain knowledge can be captured by a domain ontology. Such a domain ontology can be an existing one or a new one can be created explicitly.

- *Expressing the virtual world much more in terms of the end-user domain*
  By using a domain ontology, the design of the virtual world can be expressed

in terms of the end-user's domain and therefore, it will be more intuitive for a domain expert to communicate about the design. For instance, the fact that an object is in front of another can be presented to him as: object A is "in front of" object B (using the proper domain terminology names for the object A and B) rather than giving him low level specifications such as coordinates for the primitive VR objects.

- *Generating the virtual world more easily*
  From the knowledge given in the domain ontology, it is possible to derive a number of properties for an object and therefore, it becomes easier to generate (semi-)automatically the VR application. For instance, a cube in a language like the Virtual Reality Modeling Language (VRML) will need to have a width, a height, a depth and a position reference in the virtual world. All these parameters need to be specified explicitly. Using an ontology, it may be possible to derive that a particular object should be modeled as a cube with the appropriate values for its parameters and be positioned at the center of the virtual world.

In VR-WISE, ontologies are used for two different purposes [15].

- Externally, during the design process for representing knowledge about the domain under consideration.
- Internally, as a general information representation formalism.

This means that the modeling concepts developed are described by means of an ontology and that all information collected during the design phase are maintained in ontologies.

The design process in the VR-WISE approach is divided into three (mainly) sequential steps, namely the *specification step, the mapping step* and the *generation step*. These steps are discussed in detail in [36]. A general overview of the approach is given in Figure 2.3. We will provide here a short description.

In the specification stage, the VR application is specified at a conceptual level using the *Domain Ontology*. Different concepts are created and added to this ontology. Later, actual instances of these concepts are created, given attribute values and added to the World Specification, so called because these instances will be used to populate the virtual world. The *World Specification* describes how these instances are related, where they are placed in the world, how to interact with them and what behaviour they have.

The mapping step is used to specify how the domain concepts and instances will be represented in the virtual world. This is done by mapping the concepts defined in the Domain Ontology and the instances defined in the World Specification onto VR primitives (e.g., 3D spheres and boxes) defined in the *Virtual Reality Language Ontology*.

Finally, in the generation stage, the virtual world defined by the previous two stages will be generated in a standard VR format such as X3D [8] or VRML [25].

2.2.3 OntoWorld

Considering the fact that the theoretical work done during this thesis is based on VR-WISE, the practical part - the supporting tool - is based on OntoWorld, a tool developed by the WISE group to support the VR-WISE approach.

We now give an overview of the architecture of the OntoWorld tool. More details can be found in Appendix A in the form of different class diagrams. Figure 2.4 gives a general overview.

The user starts by first loading an ontology containing the description of the objects that will populate the 3D virtual environment (e.g., car, window, book,) or by creating one from scratch using the OntoWorld tool. For specifying and adding behavior, a Visio plug-in is provided, where the user can model behavior (e.g., a car can ride, a window can be opened) in a graphical way. Once all the input has been collected, the code generator can generate both VRML or X3D formats, as discussed previously. Using one of these 3D world descriptions, the user may be able to view the generated virtual environment. In addition a file in the MPEG-7 format will be generated containing the different properties and annotations for all the items defined in the world specification step (e.g., the car can be a BMW, the book's owner can be a specific person, etc). The user has also the possibility to export everything back to an ontology in an OWL format and reuse it later for regenerating the world.

Core

The Core Module is the heart of the tool. The rest of the code is based on it. It will thus be extensively used in our own work. The most important parts are explained below:

- *Graph Module* For an ontology, being basically a dictionary of concepts and relations between these concepts, the logical data structure to represent them

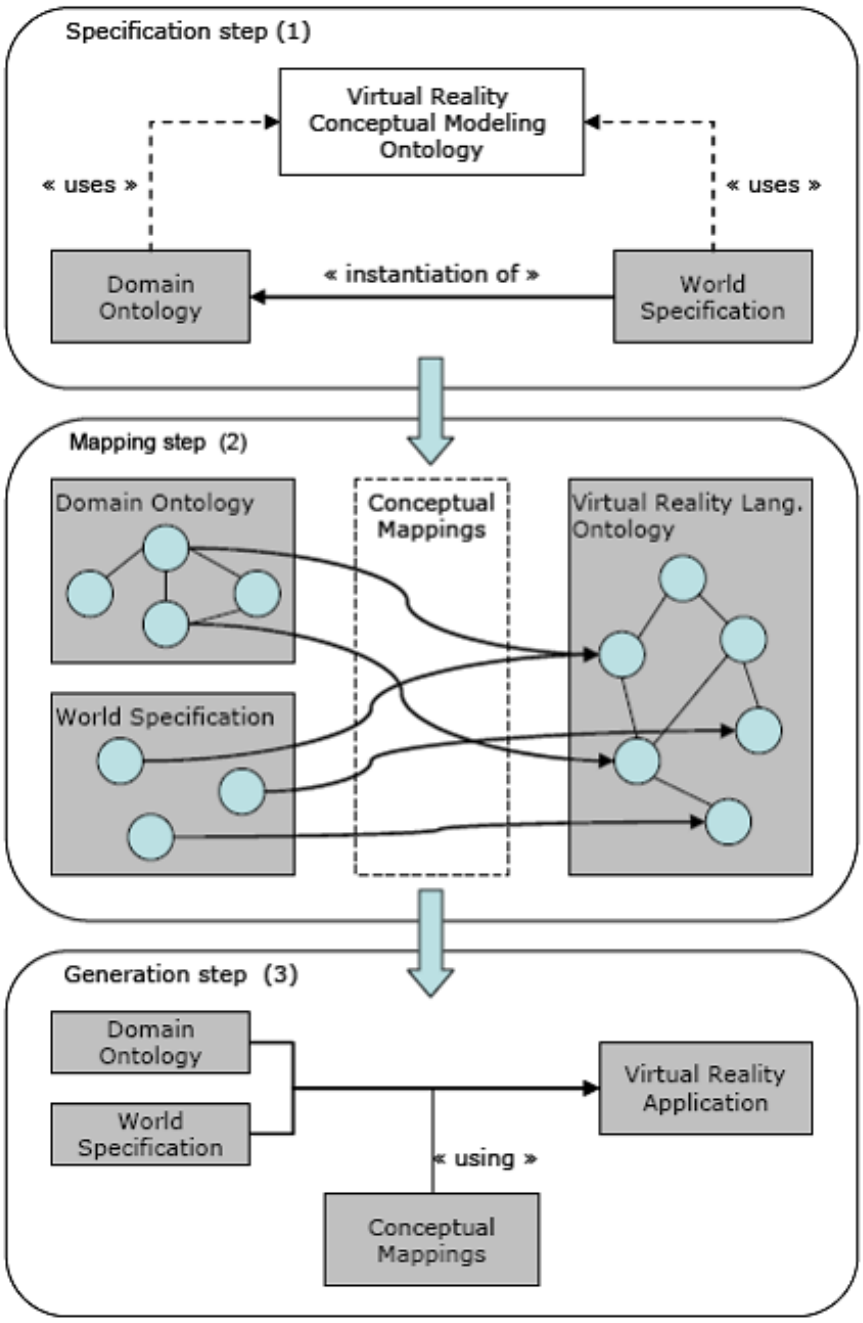

Figure 2.3: VR-WISE approach

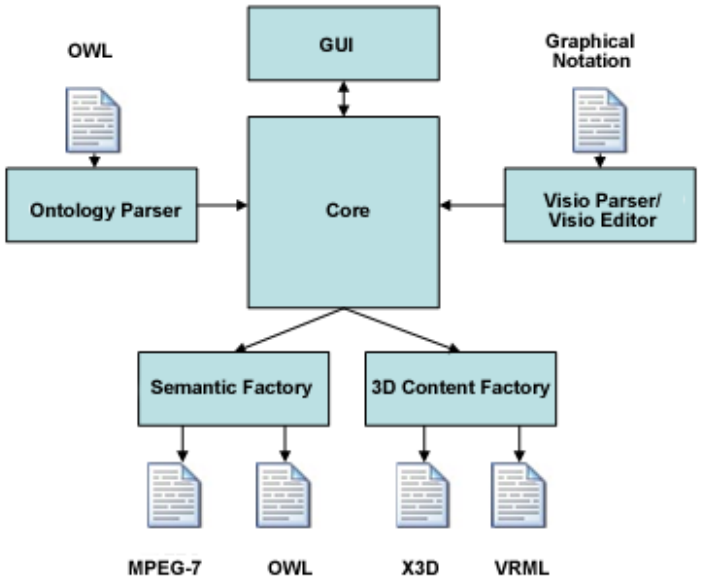

Figure 2.4: OntoWorld overview

internally is a graph, with nodes playing the role of concepts and edges representing the relationships between them. Classes from this module actually only implement an abstract data type, representing internally the information provided by the loaded or created ontologies.

- *Statics Module* This module contains the representations of what we called a concept, instance and relation. It implements the classes whose instances will be used to build the graph discussed in the previous bullet.

- *Dynamics Module* This module implements the attachment of behavior to the different nodes in the graph.

- *Collection Module* This module contains a number of data types where different kind of information is stored. The *MappingsPool* will contain a collection of all the existing mappings. Mappings have both a source and a target (e.g., we map a concept "ball" (the source) to a "sphere", being a VR primitive (the target)). The *NodesPool* type will contain all the concepts and instances we created. It actually represents a list of all possible sources. The *TargetsPool* will contain a

list of all targets.

- *Model Module* This is the main core class. It basically contains two graphs. One for the concepts and one for the instances. It is also responsible for creating these concepts and instances, for creating the relations between them and for the mappings.

Code Generator

The code generator's task is to generate code from the conceptual descriptions and mappings defined in earlier steps. At this moment, OntoWorld can generate code in VRML or X3D formats. The code generating class diagram can be found in the Appendix.

The design is simple, yet flexible. The core has a VRMLCodingStrategy class responsible for generating code for the VRML and it has an X3DCodingStrategy for generating code for the X3D format. Both classes derive from an abstract class CodingStrategy which is responsible to make sure that all the derived classes will define a number of functions members.

Each strategy contains a number of *VRObjects* stored in a hashtable and a *VRCollection* containing *VRPrimitives*. Using those data members and functions such as *CreateBox*, *CreateCylinder* or CreateViewPoint, each specific coding strategy is able to "translate" a conceptual description into a real 3D format and a real 3D world, which can then be viewed and manipulated with such tools as Cortona [2], (a VRML Client) and Web3D viewer.

GUI

In figure 2.5, a view of OntoWorld's GUI is given. As it can be seen in the left side of figure 2.5, this GUI reflects the three steps of the VR-WISE approach explained shortly in section 2.2.2 with a specification, mapping and generation step. As behaviors may be too complex for modeling through a standard form-based GUI, a graphical behavior editor has been developed. This graphical behavior editor is a Microsoft Visio plug-in. A screenshot is given in figure 2.6. More on the graphical behavior editor is described in [15].

We will not elaborate on this GUI as we will not use it in our work.

OntoWorld's GUI was designed to allow users creating virtual environment in general, not directly oriented towards a specific type, like a shop. For this reason,

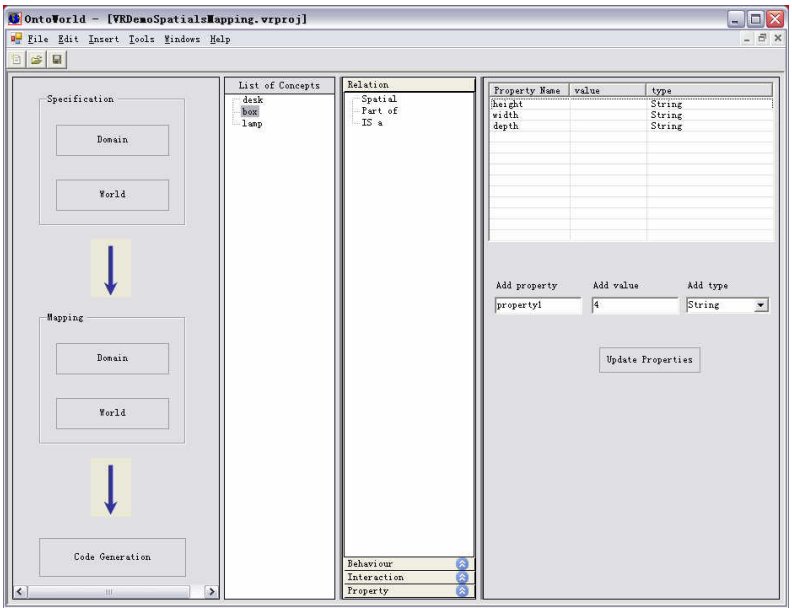

Figure 2.5: OntoWorld GUI

it uses a generic terminology, letting the user creating "concepts" and "instances", instead of "products" for example.

One of OntoShop's main priorities was to specifically customize OntoWorld's GUI to support the development of VR-shops and to be used for a specific group of users, like those working in the E-Commerce business.

This chapter has given a brief overview of the VR-WISE approach and the tool "OntoWorld" which implements such approach. More details will be provided in the next chapters.

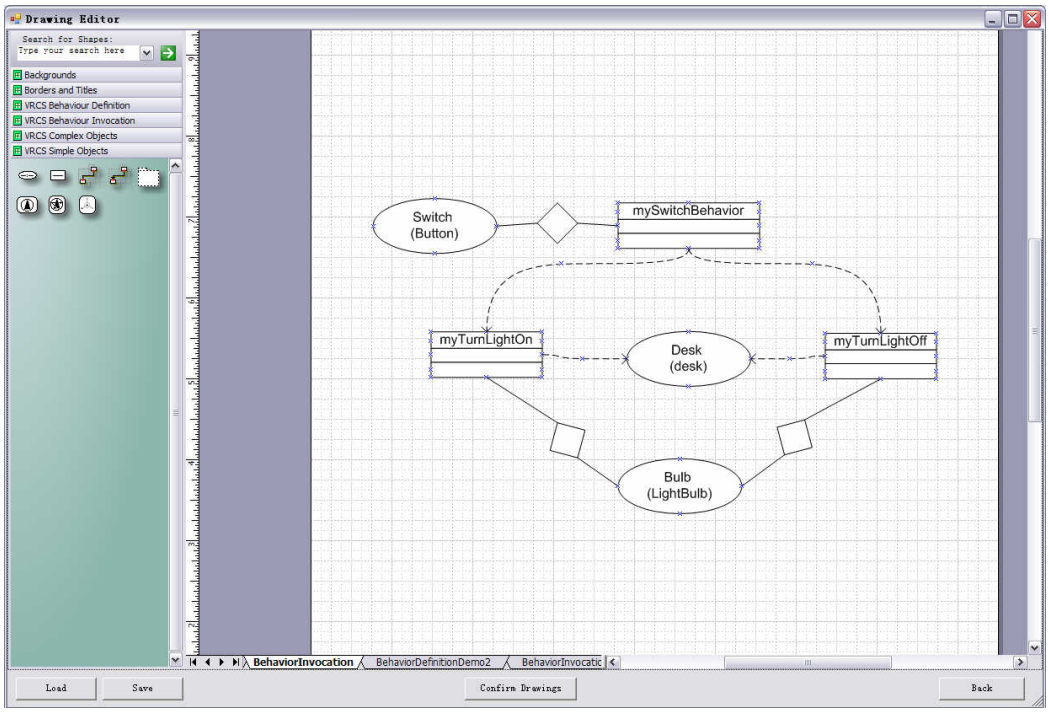

Figure 2.6: Creation of a behavior in OntoWorld

Chapter 3

SHOP-WISE approach

As stated in the first chapter, a more natural and interesting way to shop online would be to display the products and the environment in which they are displayed using Virtual Reality. In such a Virtual Reality shop, the products would be visualized as 3D objects in a virtual world where the user can walk around, inspect the products or pick them up [13].

In this chapter, we will explain the specific approach developed at the WISE Lab which allows the creation (and semi-automatic generation) of such Virtual Reality shops using ontologies.

3.1 Introduction

3.1.1 Why SHOP-WISE?

As E-Commerce is starting to use Virtual Reality technology and as this technology is difficult and expensive, it is important to develop an approach which allows E-commerce to be created with VR more easily for a non VR-Expert. It is for this reason that the "VR-WISE" approach has been developed. However, this approach was made to be generic enough in order to address a broad range of domains In this thesis, the approach has been extended so that it is only oriented towards VR Shops.

Although VR-shops do not require sophisticated VR technology, they have to 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 change issues like prices, available stock or delivery time. In addition, it should also be easy to add, adapt or remove functionality, e.g., adding the functionality of

reserving products, seeing their prices in a chosen foreign currency and changing or extending the payment method. Finally, it should be easy to adapt the virtual shop itself, i.e. its infrastructure. Like real shops, it should be brushed-up regularly and products should be rearranged.

For all of these reasons, an appropriate approach for the development of VR-shops is needed. Therefore, the aim of this thesis was to address this problem by introducing a new approach called SHOP-WISE approach which is based on the VR-WISE approach and oriented only toward VR Shops.

3.1.2 Benefits of the SHOP-WISE approach

SHOP-WISE foremost advantage is that practically anyone with some (intermediate) computer experience is able to develop a fully 3D virtual shop which he can put online and start exploiting it for making money! This reason on itself may not be enough for the academic world, but the author believes it would be more than attractive for big companies in search of a good business opportunity. If marketed in the right way and with some extra features added to it (cfr. Chapter 6 about further research), the theory explained here and in [23] could represent a multi-million deal in practice.

A more detailed explanation follows:

- The SHOP-WISE approach allows developing VR-shops using high-level, conceptual specifications in terms of concepts from the shop-domain. It means that this approach can be used by a non-VR expert.
- As a consequence of the first bullet, the development of VR-shops is facilitated and the cost is considerably reduced.
- A facilitated development also implies shorter development times.
- Results are easy to maintain, update or adapt. In order to do so, the needed modifications can be made at the conceptual specification level and the adapted shop can be easily re-generated.
- Semantics are captured by means of ontologies, which implies that existing product information can be re-used.
- A number of primitive concepts, behaviors and functionality, which we believe should be used - or at least exist - in every virtual shop has been built-in. For example, we may assume that every shop will have at least some walls

and racks (these are primitive concepts), and that available products should at least be buyable (this is a functionality). Default libraries should make a shop developer's work easier and more understandable.

3.1.3 Literature study

Before starting to create the approach, we had to understand how people who are going to use it work and think in daily life and how shops are built in reality. For achieving this, we first have made an ethnographic study of real stores.

Ethnography provides interesting ways to gain insight into existing processes, focusing the attention on the task and actions performed by different people. As a first task of the development process, an ethnographic study of a shopping mall had to be fulfilled to gather the information of the domain of the problem. A study of a real store had to be made first in order to gather the requirements of store, and to identify the main processes that it involves. Such a study has been conducted in [13]. The ethnographic study of a real store included the observation of the behavior of customers when shopping, and interviews with different employees of the store.

As it was observed and later confirmed by those employees, the reason that leads a customer to buy a product is her or his interest on it, which derives from necessity or simply fondness. For example, a customer that is fond of computers may decide to visit the computer accessories section in a first place, and if a product grasps her or his attention, she or he may decide to buy the product even though its purchase was not in mind before finding it.

The interviews of different employees has revealed some of the reasons that guide the allocation of products in the store. Among the different reasons, it was observed that the interest of the store manager is to sell certain products. For this, the store manager places them in areas seen by most of the customers such as the corners of shelves or the central part of pathways, since these places are the most seen by customers, which are then attracted by the products located in them. Sometimes, these products are sales, which the shop sells at less than the normal price. Sometimes, the reason is that there are too many items in stock.

Marketing studies also guides the allocation of products in the store, as the employees recognized in the interviews. These studies analyze the behaviour and preferences of customers according to several factors, such as age, genre, purchasing power, country, etc. As the employees recognized, the studies show that it is better to place products near related consumer products, as there is much chance that when a customer buys a product, she or he will buy a related one. For example, if the customer

looks for a TV set, she or he will find other related products near the TV sets, e.g., a DVD-ROM, that the customer may think of buying too.

A list of requirements and characteristics that a store should exhibit has also been composed in [13]:

- The customer is free to enter in the shop, no ID is required
- The customer is free to walk around in the shop
- The customer is free to look and examine the products
- The customer is free to purchase products or leave the shop without buying
- The products must be well organized, grouping related items
- The information of the products must be clear and specific
- The interests and preferences of customers must be taken into account

In [16] it has also been suggested to think about substitutable products (e.g., coca-cola & pepsi-cola) and complementary products to make customers buy (e.g., cheese & wine).

Finally, in [27], the importance of showing the customers where products can be found in a virtual supermarket has been demonstrated.

During our work on this thesis, we have taken into account all of these characteristics of real stores for modeling an approach (SHOP-WISE) and building a tool (Onto-Shop) allowing the users to create a virtual shop and browse it, in a way that is as natural as possible, i.e. like in real life.

3.2 Developing VR-Shops using the SHOP-WISE approach

Figure 3.1 illustrates the development process of SHOP-WISE. A detailed explanation follows

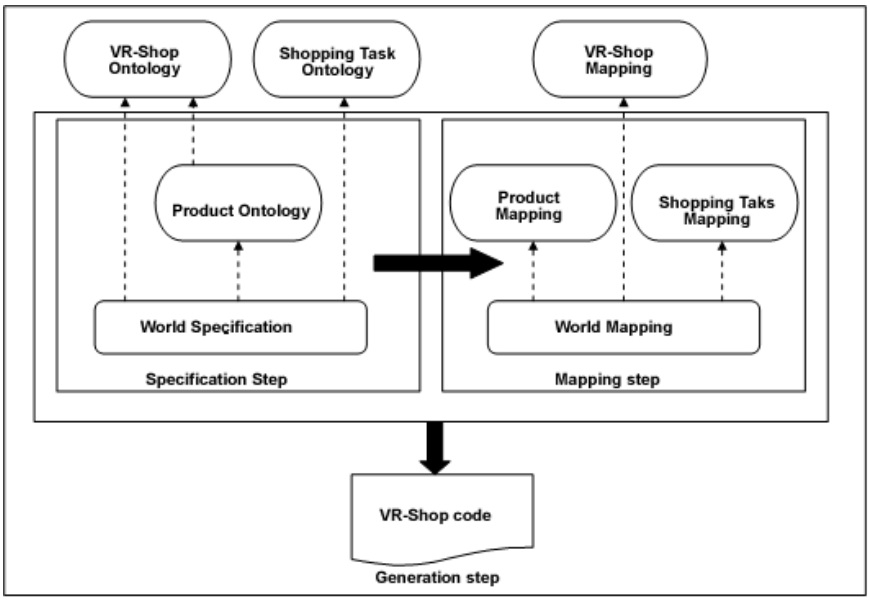

Figure 3.1: Development process of SHOP-WISE

3.2.1 Specification step

Specifying a VR-shop is done at two levels: the type level and the instance level. These two levels can be compared to what is called "class" (or "type") and "instance" in OOP languages. The type level allows specifying the different types of products and other objects that will be available for use in the shop. We also call these types "concepts". At the instance level, the actual shop is composed: first, a virtual world in which the products can be placed is specified. Therefore the layout and infrastructure of the shop is created. Later, the actual products (instances) are specified and placed in the world. Finally, we need to specify what tasks should be available for what products. For this, it necessary to provide shopping functionality to these products.

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*.

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 the generic concepts useful for each VR-shop.

Infrastructure concepts A first group of concepts in the VR-Shop Ontology are the concepts used for specifying the infrastructure of the shop. After some research "on the field" in several shops, it has been found that this group can be divided in three main categories:

- *Universal shop infrastructure* These concepts will probably all be instantiated, no matter what kind of shop is modeled.
  - Walls, floors and ceilings: these are the first infrastructure objects that come to mind. They are the basic building blocks for any kind of (3D) building.
  - Racks: this concept is not only applicable for typical supermarket racks. It can also be used for creating instances of refrigerators, snack vendors/vending machines or beverage holders. Anything that is meant to show a product can be considered to be a rack.
  - Areas: this concept can be used to define areas in the shop that have a specific purpose. Some areas are pre-defined like the Entrance.
  - Informative/Promotional panels/sheets: these sheets can be used for pricing a product or give its description. It could also be an advertising panel, showing special promotions for products or newly added articles. Lastly, informative panels can be useful for easily allowing customer to find their way to a product they need (e.g., with an arrow pointing in the direction of a specific product category, like "electronics" or "furniture").
  - Light sources: light is not only a way to make products visible. It is also a way to emphasize them and attract customer's attention.
  - Shopping cart: collect the items of a customer.
- *Specialized shop infrastructure* Some of the concepts enumerated below will be useful in certain shops, while not on others.

  - Windows, doors, columns: not every shop will use these concepts (supermarkets do not), but they may be useful for smaller virtual environments.
  - Information desk: this is a variation on the informative panels stated in the first bullet. The difference is that by taking extra room, it will have more chance to catch a customer's eye. For this reason, important information can be collected here. There should not necessarily be someone behind this desk, written text is an easier solution.
  - Cashiers' desk, check-out facilities: though a cashier's desk is not strictly necessary for buying product in the generated virtual shop. But it can be used to provide the illusion of reality and it may also be the place for providing information to the customer about how buy the chosen products.

- *General infrastructure* These concepts are so called "nice to have": they may make a shop look more attractive and real, but in no case are they required.
  - Ventilation tunnels, speakers, electric cables/wires, steps, elevators, video cameras

One of the *Universal Infrastructure* items - promotion of products - may require some more explanations. In [27], Ranon and Chittaro have introduced the concepts of *Walking Products*. Those are products that walk in a virtual shop and show customers the location of a specific group of products. For example, a walking television may regularly pass through the electronics section or a walking chair may walk around the furniture area. This approach's main benefit is the time customer need to find what they are looking for. From 182 seconds needed to a customer to complete a search task in a virtual shop without walking products, he only needs 77 seconds in a shop where such products walk. In the same paper, the authors have emphasized the importance for customers to know where to find what they need for improving the usability of the virtual shop. It is for this reason, that we have thought that something specially dedicated to the promotion of products needed to be available.

All the items on the above list represent *concepts*. If needed, new shop concepts can be added to this VR-Shop Ontology. For actually placing these concepts in the virtual shop, they need to be instantiated. For instance, 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.

With some imagination, a lot of different virtual shops can be created. For example, if the concept floor is instantiated with a texture representing grass, the concept

wall is instantiated with a texture representing trees and the concept ceiling is instantiated with a texture representing a blue sky, virtual visitors will be able to buy their products in open air!

Abstract concept Product Each VR-shop will also contain products. These are the products for sale. Therefore, it was decided to provide the abstract concept *Product*. This concept is pre-defined with some properties that are usually applicable to all products, such as *name, description, price, packing size* and *weight* (some of these properties being optional). As the products need to be visualized, they also have a property *visualization* that allows specifying the visual 3D representation of individual products. All product types that will be available in the shop should be defined as subclasses of *Product* (cfr. Section 3.3.1.3). In this way, they will inherit the common properties defined for the concept *Product*.

Behaviors 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, also behaviors like turn, *changeColour*, *addToCart* and *removeFromCart* are pre-defined. These behaviors can then be associated with specific products. *showContent* and *makeEmpty* are 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.

Shopping Task Ontology

A 3D e-shop is of no use for the customer if it doesn't provide some basic shopping functionality like the possibility to "buy" products. Together with buying, most E-shops provide similar functionalities. We will call them "tasks". These tasks are grouped into the Shopping Task Ontology. This ontology contains the following tasks:

- *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 and show it.

- *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.
- *Compare same products.* This task lets the user comparing one product between different E-shops.
- *Post product review.* Allows a user to write a small review about the product in question and/or his experience with it.
- *Review product reviewers*
- *Calculate shopping cart total.* Return the total amount to pay for all items in the shopping cart.
- *Calculate product shipping price.* Calculate the shipping fee for one specific product.
- *Calculate total shipping.* Calculate the total shipping 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, reserve it. The user should be logged in first.
- *Register and Login.* At first sight, it may seem more logical to force the user to register and/or login before entering the shop. However, this process may bore (or scare) potential buyers before they even take a look at the virtual shop. Also, as stated in the ethnographic study from section 3.1.3, customers are allowed to enter real shops without even being forced to buy anything. For this reason, we decided to let the user freely browse the shop first. After that, if he is ready to buy, he may register and/or login.
- *Subscribe for promotions or email notifications.* This task should give the user the possibility to subscribe for special promotions or ask for notifications about, for instances, new added products or newly created features in the virtual shop.
- *Compute tax.* If product's prices do not contain the VAT, shop user may calculate it with this task.

- Bookmark product. In big virtual supermarkets, bookmarking an often visited/bought product can be an added value.
- *Email URL to a friend.* There's no smarter, cheaper, or more effective way to market than by word-of-mouth. When customers or peers discuss the benefits of a certain business, the conversation generates excitement and instant trust for this business' products or services. This simple task may actually be an important profit generator.
- *Print out a product's details*
- *Order history.* View the order history of the user profile a virtual customer is currently using.
- *Customize Product.* This functionality is for shoppers who more and more want to be their own broker, and not only select but interactively customize products to fit their individual needs.
- *Bargaining.* Though maybe a somewhat unusual task from a Western point of view, this functionality will be very appreciated by, for instance, Asian online shoppers, who still have a high preference for face-to-face buying and live bargaining.
- *Find related products.* Find products in the same category. For example, "computer" and "electronics", "car" and "vehicles", "books" and "educational resources".
- *Find add-ons.* Find products that can be used with this product. For example, "paper" to a "printer" or "cheese" to "wine".
- *Finding new products.*

The last three tasks are actually part of a much broader functionality, which is common to all E-Commerce websites: searching. Searching is absolutely a compulsory task. According to Forrester Research [33], convoluted e-commerce sites can lose up to half of their potential sales and 40% of their return visitors if customers can't find merchandise. Searching improves usability, increases customer satisfaction and leads to better sales.

As virtual worlds may become very large and incorporate many different virtual objects, it is becoming very important to be able to query the virtual world in order to find the objects needed or to get quickly to a desired location in the virtual world.

At WISE, a mechanism has been created to attach a vast amount of semantic information to virtual worlds according to the domain for which a virtual application has been developed. Later on, this information is used by the search engine to allow the user to efficiently query the virtual world for objects populating it. It is possible to solve queries expressing vagueness such as "I am looking for a cheap table". This is an important topic in virtual environments, which however we will not be discussed in this thesis. More information on this can be found in [23].

For each task, a number of parameters will need to be defined. For instance, the task for converting currency needs three parameters: the price, the starting currency and the currency to convert to. Furthermore, a complex task can be made by combining a number of simple tasks. The task "buy" for instance, may require first to execute the task "check stock". Like for behaviors, it is not specified how these tasks are implemented.

Product ontology

Each type of product that may be for sale in the shop should be defined as a concept in the Product Ontology. 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 properties defined for *Product*. For each product type, additional relevant semantic properties and information can be specified.

World Specification Ontology

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).

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 where 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*) 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 [35].

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 who are not experienced with VR.

Once the shop and the products have been defined and positioned, behaviors and tasks can then be added 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 will not happen very often. More on this can be found in [17] and [16]. When attaching behaviors and tasks to instances, it must also be specified how the behaviors and tasks 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.

3.2.2 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 it must be specified how the objects defined in the World Specification should be visualized using VR-implementation primitives and how the behaviors and tasks should be realized.

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 be represented as a box by default. Furthermore, our approach tries 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 since it may not exist or complex objects need to be used. More on this can be found in [35].

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 map the *visualization* property of a *Product* 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 to be a curved wall, he can map this wall-instance onto the VR-Primitive "2D Curved Surface". Here, some help of a VR-expert may be needed if the designer wants to make non-trivial modifications. For the product instances, the mappings specified for the 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 [7].

A web service is a collection of protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer [14].

As stated in [19], Web Services have several advantages:

- Web services provide interoperability between various software applications running on disparate platforms.
- Web services use open standards and protocols. Protocols and data formats are text-based where possible, making it easy for developers to comprehend.
- By utilizing HTTP, web services can work through many common firewall security measures without requiring changes to the firewall filtering rules.
- Web services easily allow software and services from different companies and locations to be combined for providing an integrated service.
- Web services allow the reuse of services and components within an infrastructure.

There are some disadvantages as well:

- Web services standards for features such as transactions are currently nonexistent or still in their infancy compared to more mature distributed computing open standards such as CORBA.
- Web services may suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off when choosing text-based formats. XML explicitly does not count among its design goals efficiency of parsing.

As web services are nowadays well accepted, it was decided to use them for implementing tasks in our approach. Web services have the following advantages:

- First, it allows us to add already existing functionality to VR-shops by using publicly available Web Services such as those that can be found in one of the Universal Description Discovery and Integration (UDDI) directory [10]. For instance, we can use public web services 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.
- It also allows us to build a more flexible system, where different programming languages could be used, and different services can be easily replaced by an equivalent one.

- Web services make the system independent of how the underlying procedures (e.g., the use of a database system) are implemented.
- It also allows for personalization as e.g., the task of getting the price can be mapped on a web service that calculates the price depending on the discount assigned to the customer and on current promotions.
- Finally, by using Web Services, we separate functionality from the rest, which does not only make it easier to understand, but also promotes reusability.

Our approach assumes that each task that needs to be supported, a web service exist or can be built.

3.2.3 Generation step

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 OntoShop, developed to support the VR-WISE approach (cfr. Chapter 4). Actually, OntoShop generates two files. A first file contains the 3D scene structure with its objects. It is an X3D format [8] or VRML format [25]. The second file contains the semantic annotations and is generated in an MPEG-7 format [31]. MPEG-7 is used, as it is an ISO/IEC standard supported by the industry [31] and it is readable by human and machine [21]. 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. More on MPEG-7 can be found in [23].

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 thinkable 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.

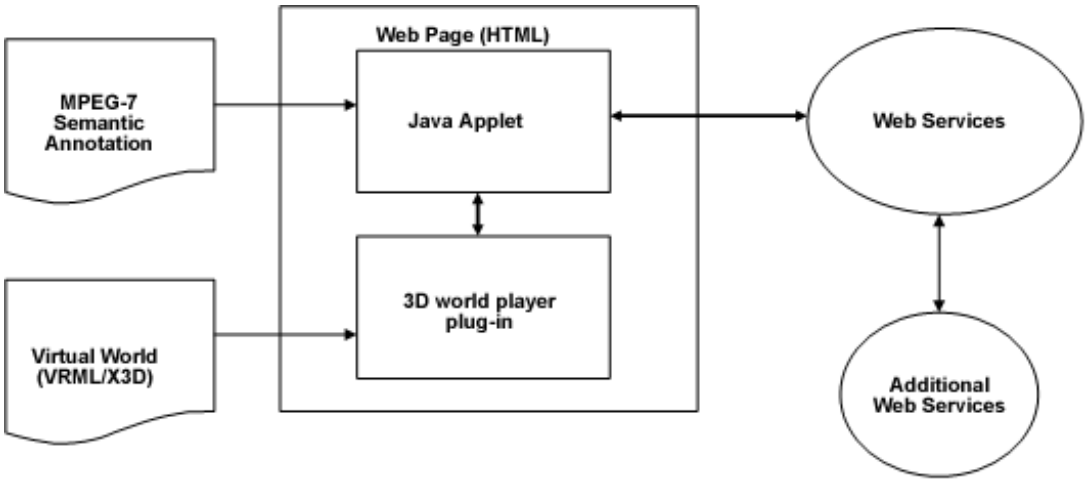

Figure 3.2: Architecture of a VR-Shop

The complete architecture of a generated VR-shop is given in figure 3.2. 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 [2]. The applet communicates with the plug-in via EAI [25] and is also capable to communicate with the search engine (see 3.2.1 and [23]) 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.

Chapter 4

OntoShop

As stated before, OntoShop is a customization of OntoWorld. It serves as a proof of concept of the feasibility of the SHOP-WISE approach. In this chapter, we will give some clarifications about how this tool has been implemented and on what grounds it was decided to do so. For the implementation, Visual Studio has been used, with its .Net framework. The chosen programming languages is c#.

4.1 Needed customizations and extensions

Let's take a look at what was needed to be customized and what extensions were needed to OntoWorld.

4.1.1 Creation of dedicated ontologies

First, it was necessary to create a number of ontologies: the VR-Shop Ontology and the VR-shopping Task Ontology. For this, we have used VR-shopping Task Ontology. For this, we use "Protege" (an open source ontology editor and knowledge-base framework [4]). Using Protege, all the shopping tasks described earlier were added to the Shopping Task Ontology (cfr. Section 3.2.1) and all the generic VR shop concepts were added to the VR-Shop Ontology (cfr. Section 3.2.1).

4.1.2 Creation of a new GUI

While creating the SHOP-WISE approach, we needed to consider what typical tasks, behaviours, and objects should be available in virtual E-shops. In the same way, for implementing OntoShop, we have made a study on how shop managers think

about creating a shop, maintain it and control it. A new GUI has been created for facilitating a shop manager's work during the development process of the virtual shop. This has been done by allowing him to follow the same strategy as when creating a real shop.

4.1.3 Integratation of OntoWorld's Core

For implementing OntoShop, the already existing Core of OntoWorld was used. However, some small changes were needed. For this reason, we have written an interface for OntoWorld's Core. An interface refers to an abstraction that an entity provides of itself to the outside world. This separates the methods of external communication from internal operation, and allows it to be internally modified without affecting the way outside entities interact with it. In our case, we have used an interface between OntoWorld's Core and the other newly built modules in OntoShop.

4.1.4 Adding task functionality

The VR-WISE approach does not support non-VR tasks, nor does OntoWorld. As explained earlier, one of this thesis' intent was to create the possibility for users to add (shopping) functionality to their generated virtual worlds. Shopping tasks are implemented through the use of web services as explained in section 3.2.2.

4.2 Creation of ontologies

As explained earlier, two ontologies were needed: the Shopping Task Ontology and the VR-Shop Ontology. Ontologies were created with Protege [4]. Figure 4.1 depicts a part of the created VR-Shop Ontology.

The format used is OWL, which is itself based on RDF, which in turn uses XML. There are three major kinds of tags in the Ontology. First, classes (at the top of figure 4.1), represent the different pre-defined infrastructure concepts (e.g., Window), behaviors (e.g., Change_Colour) and generic objects (e.g., Shopping_Cart) available in our shop (cfr. Section 3.2.1). Second, ObjectProperties (in the center of figure 4.1), represent relations between two different instances of a class (called "individuals" in OWL). For instance, the relation Add-on is a relation between two Products classes. Lastly, DatatypeProperties represent a relation between an individual and a datatype (e.g., string, boolean). For instance, a product will have a Condition, which can be Normal, Bad, Very Bad, Good and Very Good (at the bottom of figure 4.1).

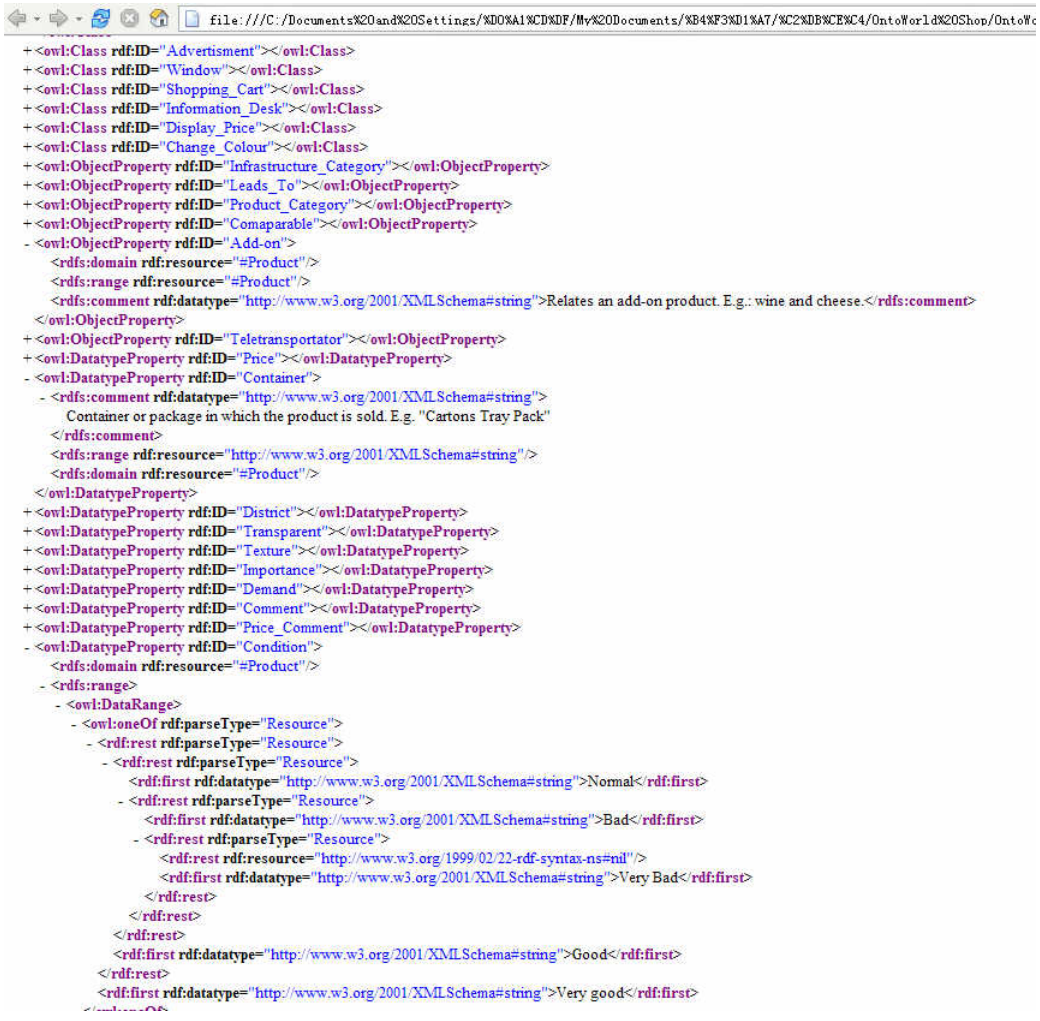

Figure 4.1: A part of the VR-Shop Ontology

An important class of the VR-Shop Ontology is the Product class. It is the abstract concept Product introduced in section 3.2.1. This concept is pre-defined and has been given some properties that may usually be applicable to all products in the VR shop. These properties are:

- *Name*. Name of the product.
- *District*. Describes the region where the product originated from. This may be a region within a country, or a port of entry for a foreign commodity or a general area covering two or more countries.
- *Category*. Describes the category of this product. E.g.,: "electronics".
- *Sub_Category*. A second category for a product. E.g., "Computers", inside "Electronics".
- *Color*. Color of this product.
- *Comparable*. Indicates to what other product this product can be compared.
- *Add-on*. Indicates to what other product this product can be related. E.g.: wine and cheese.
- *Condition*. Stage of maturity and other factors affecting a product.
- *Container*. Container or package in which the product is sold. E.g., "Cartons Tray Pack".
- *Demand*. Represents the immediate or current desire for a product and the ability and willingness of the buyer to buy it. This can be useful for statistics.
- *Import_Export*. Indicates whether the sale is for domestic consumption or for exporting only.
- *Material*. Indicates from which materials this product is made.
- *Only_for_Adults*. Indicates if this product should only be available for adults. E.g., alcohol, cigarettes, ...
- *Price_Comment*. Additional factors specifically affecting the price.
- *Quality*. Physical properties of a product.

- *Sales_Term*. Conditions that determine the type of sale. E.g., "Delivered sales", "F.O.B" and so on.
- *Season*. The crop-year of the commodity, based on the harvest start date.
- *Storage*. Storage or other external factors affecting the product. E.g., "Controlled Atmosphere Storage," "Regular Storage" and "Unwashed".
- *Visualization*. Complete path to the file that contains the texture representation of this object.
- *Unit_of_Sale*. The unit at which the product is sold. E.g., "per kilogram", "per chair".
- *Weight*. Weight of this product.

All these properties will be available for all concepts of product created in the *Product Ontology*. Notice that obviously, not all the properties are mandatory.

The *Shopping Task Ontology* is less complex. A class *Task* is created and all predefined tasks summed in section 3.2.1 are subclasses of this class. The most important property here - common to all tasks - is *Web_Service_Address*. It describes to which web service a task is being mapped on. More explanation will be given about this in section 4.5.

4.3 GUI creation

As stated in section 3.1.2, one of OntoShop's benefits is the shorter development time that it provides for modeling and generating virtual shops. Though the main reason for this gain of time is the approach that lies behind it, a correctly designed Graphical User Interface may further enhance effectiveness, efficiency and user satisfaction. Properly customizing the GUI part was therefore crucial.

Our primary goal when creating the GUI was to develop a specific GUI for the creation of Virtual shops using OntoShop. It was therefore important to identify users and what they will need to do. This study is partly based on the discoveries made during the ethnographic study from section 3.1.3. The next subsections will address these issues.

4.3.1 Potential users

A first group of potential users are the persons who are responsible for the marketing and development of products. They are also responsible for positioning products, assessing the competition and thinking about the future. Furthermore, they must know their company's product to the perfection and track what customers want, think and respond to their needs of new products. For this reason, a product manager is the person who will be responsible for the Product Ontology. This user will know in details what products exactly can be placed (or not placed) in the shop, i.e. what products can be sold. He will input the information related to available product and he will decide for each of them what Visualization it has (see also further on for this last issue).

The second type of potential user is the user who is going to be responsible for the defining what infrastructure is used in the shop and for instantiating it. This user will both create infrastructure concepts with their properties as well as creating the shop's layout.

The third potential user is the shop manager. As stated during the ethnographic study, the shop manager is the person who decides where products should be placed. He will be responsible for placing products in the virtual shop and rearranging their placement depending on the available stock, some temporary promotion, or other factors. The shop manager will also attach the behaviors and tasks to all the instantiated products.

A final category of user for OntoShop will be the technically experienced IT people. Though with some training, any person with intermediate computer knowledge will be able to develop a virtual shop using OntoShop, a computer scientist may help. This should not be a VR expert (although it may be the case). However, it should be someone who has enough computer knowledge to execute the following tasks:

- Find or create the right textures or 3D models (with the right textures) for all the available concepts and instances described in the different ontologies.
- Help other users during the second step of the SHOP-WISE approach, namely: the mapping. For example, if some exotic (oval) windows need to be created. They need to be mapped on a non-standard VR primitive. For this, the IT expert may help.
- Find (or create) the needed web services that will be used in the virtual shop. Creating web services is practically the only time that a computer engineer

is required. In contrast with the above two bullets, where shop and product managers should be able to cope with the technical difficulty after a short training (if no textures and 3D models need to be created, but existing ones can be reused), creating web services requires programming skills.

Nevertheless, OntoShop also provides a library of VR objects which can help the design of VR-shops.

4.3.2 Creation of user profiles

In the beginning of the development of this thesis, it has been imagined to create four user profiles with a different GUI for each: one for the products creator, a second for the layout designer, a third for the shop manager and a last for the IT expert. However, it was soon clear that such a clumsy interface would not be intuitive. On the contrary: it will even be less usable.

It is for example practically impossible to create a GUI only for the IT expert. His primary job is to help other user when those have problems, not to develop the shop. Secondly, a product manager will not always exist in smaller enterprises, nor will a shop manager be available in enterprises that have no physical shops.

Because of all these issues, it has been decided to create a GUI that does not focus on who are the users, but rather on what to do. Concretely, two user profiles have been created: one for the *shop designer* and one for the *shop manager*.

The shop designer is the one who is going to *design* new items. Using the SHOP-WISE terminology, it is the person who is going to create the concepts, both the items from the *Product Ontology* and items in the *VR-Shop Ontology*. The shop designer will thus add new product concepts (e.g. "chair" or "car") and infrastructure concepts that are not yet available in the VR-Shop Ontology, but are needed for his specific shop. The shop designer will add all the needed default properties and attach all the default behaviors and tasks. In other words, he will create the building blocks which could later be used to create the virtual shop. The shop manager, in his turn, will use the created building blocks to "assemble" an E-shop. This means that he will instantiate the concepts created by the shop designer to generate the virtual shop.

4.3.3 Created GUI

The created GUI directly incorporates two user profiles: one for the shop designer and one for the shop manager. As explained earlier, the shop designer will create

all the objects (concepts) that will be available for use in the shop, while the shop manager will instantiate them and place them into the shop.

Many screenshots and detailed explanation can be found in the next chapter, where a case study will be given. Here, we only provide a short overview of how the GUI has been designed.

Shop manager GUI

Using the ethnologic study in section 3.1.3 and the created user profile in section 4.3.2, we first summarize a shop manager's responsibilities:

- *Create/manage the layout. Instantiate the infrastructure.* Obviously, virtual wall and racks are easier to move than real ones. Depending on a products' place - among other factors - customers may feel more attracted to buy it. Layout management will probably be an important part of a shop manager's responsibilities.

- *Place the products and rearrange them when needed.* It is up to the shop manager to decide where products should be placed, what behaviors and tasks should be attached to them, whether to use massification [36], and so on.

- *Promote these products; let the customer know about them.* In real shops, many products are promoted with some text or picture on (sometimes large) promotion panels. This approach targets two goals: increase sales for some specific product and inform customers about where they can find the products they need. The shop manager's responsibility is to achieve both goals by using and placing correctly the informative and promotional panels/sheets described in the Universal Infrastructure list in section 3.2.1.

- *Generate the shop* This is more than just pressing a button, as it will be explained in the use case of chapter 5.

- *Control customers' behavior, needs and wants in the shop.* Controlling is one of the main activities for any kind of manager. Specifically, a shop manager should be able to get statistics like products' popularity, sales volumes, but also most visited or used paths in the virtual shop, products' visibility and others.

Based on what has been described above, a simple GUI was created for the shop manager. It is a step-by-step logical way of developing a shop that fits - we believe -

with a shop manager's way of working. The first step consists of adding new instances to the shop. This step is divided in three sub steps: adding infrastructure, products and promotional objects instances. Step number two is the generation of the virtual shop. The last step will allow a shop manager to "control" the virtual shop by getting statistics and other important data (cfr. Chapter 6).

Design manager GUI

Although logically, OntoShop's first user will be the designer manager - the one creating all the concepts for the virtual shop - his GUI was actually developed based on the GUI of the second user: the shop manager.

This GUI also follows three steps. Step number one allows the shop designer to add all the needed infrastructure, product and promotional object concepts. This is completely similar to the shop manager's GUI. However, a designer will need to be a little more technically experienced (or get some help from the IT expert) as the second and third steps are respectively the creation of new behaviors through use of the visual behavior editor and the mapping of tasks to web services. This last may be more elaborated than it looks at first sight. Earlier, it was explained that each required web service needs to exist (or to be created) before it can be used. Some programming will be required. We will see how this is done in chapter 5, when we describe the use case scenario.

4.4 Core integration

The Core is the central part of the complete program. It is largely based on the Core Module of OntoWorld (cfr. Section 2.2.3). Extra needed functionality has been added. However, it has been decided from the beginning that all access to the Core will go through interfaces. The main reason behind this choice is of course the flexibility that it procures. If for some reason a new Core is created (or the old Core is changed), the rest of the code does not need to be re-written.

The result of the policy explained above was that a large number of interfaces were written. For an easier control on those interfaces, the Facade design pattern was implemented. The Facade pattern provides a unified interface to a set of interfaces in a subsystem. Facade thus defines itself a higher-level interface that makes the subsystem easier to use [18]. Figure 4.2 depicts the idea behind the facade pattern.

We will review now the different parts of the Core and explain how it was integrated and extended in OntoShop.

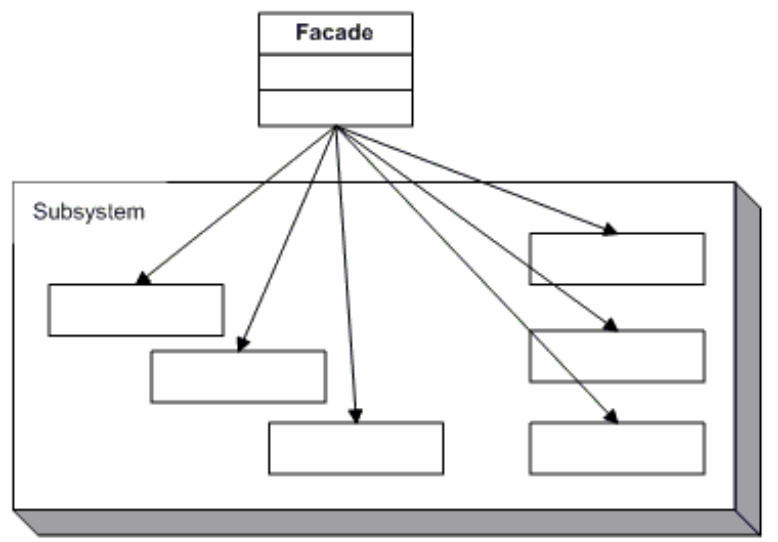

Figure 4.2: Facade pattern

4.4.1 Nodes

In chapter 3, it was stated that the SHOP-WISE approach uses ontologies not only for representing knowledge about the domain under consideration, but also as general information representation formalism. It has also been stated that the best way (in terms of speed, and extensibility) to internally represent an ontological structure with entities and relations between those entities in an abstract data type is to use a graph with nodes and edges.

In this case, nodes will represent any entity that is related to something else by means of any relation. In OntoShop, *Nodes* can either be the representation of a concept, an instance or different kinds of behaviors. This is shown in figure 4.3. Note that there are different kinds of concept and instance nodes, namely: "infrastructure", "products" and "promotion" nodes. For example, every time the shop designer is going to create a new infrastructure concept, the *InfrastructureConceptNode* type will be instantiated, while every time the shop manager wants to add some product into the shop, a *ProductInstanceNode* will be instantiated internally.

Furthermore, the abstract class *Node* has itself been derived from the interface *INode*. The reason for that (extra flexibility) has already been explained in section 4.4. Figure 4.3 contains a last group of classes derived from *Node*. These are the classes used for modeling behavior. Nothing has been added there. The whole has been taken from the Core of OntoWorld. These classes translate the graphical behavior

diagrams created in Visio by the user into Nodes. More on this can be found in the next chapters.

4.4.2 Edges

The edges between the nodes in the graph represent relations between concepts and instances. Just as for *Node*, there can be different kinds of *Edges*. Figure 4.4 gives an overview of the two most important classes derived from *Edge*: *SpatialRelationEdge* and *OrientationRelationEdge*. As explained in section 3.2.1, a spatial relation uses one or more high-level concepts (such as *LeftOf* or *Above*) to position an object with respect to another object, while 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.

Notice also that every *Node* contains a collection of *sourceEdges* and *targetEdges*, specifying respectively what relations this specific node has with other nodes and which relation other nodes have with this specific node. These edges are stored in an *ArrayList* (a vector) and indexed in *Hashtables* for fast retrieval. In the same way, every edge does itself contain a pointer to both the *targetNode* and *sourceNode*. Furthermore, *Edge* and *EdgesCollection* will never be accessed directly, but through interfaces. Such an approach allows the Core to be independent of the external modules (for instance, the GUI). If for some reasons, the Core needs to be updated, the only place where we will need to apply changes (if needed at all) would be in those interfaces and nowhere else.

This is the main core class. It contains the concept's graph as well as the instance's graph. The *Model* class has been taken from OntoWorld's Core. Its class diagram is given in figure 4.5. It is responsible for creating concepts, instances and behaviors, together with their relations and to add them into a *ConceptualGraph*. This is the main graph class which is used as a data structure to hold all the data in the internal model. OntoShop instantiates two graphs one for the concepts, called *domainSpecification*, and one for the instances, called *worldSpecification*.

The *Model* class also contains two other important fields (data members): the *MappingPool* and the *NodesPool*. The *MappingPool* is a collection of mappings for both the concepts graph as well as the instances graph. The *NodesPool* class collects all the nodes of the graphs (concepts and instances) and contains additional information for the nodes which are not in the graphs themselves.

Nevertheless, the *Model* from OntoWorld needs to be adapted for OntoShop. Indeed, OntoWorld, being usable for generating very generic virtual worlds, has only

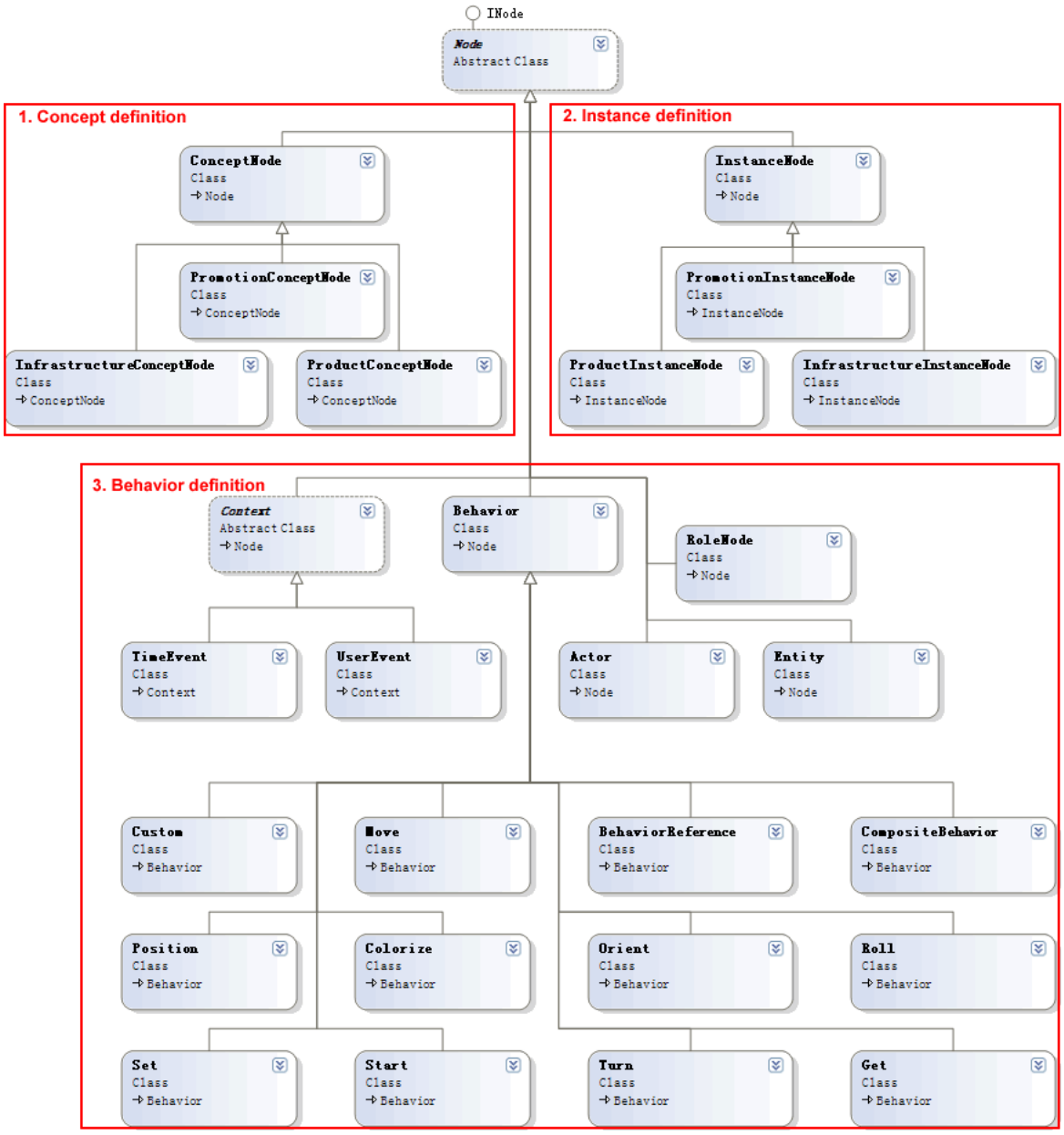

Figure 4.3: Classes derived from Node

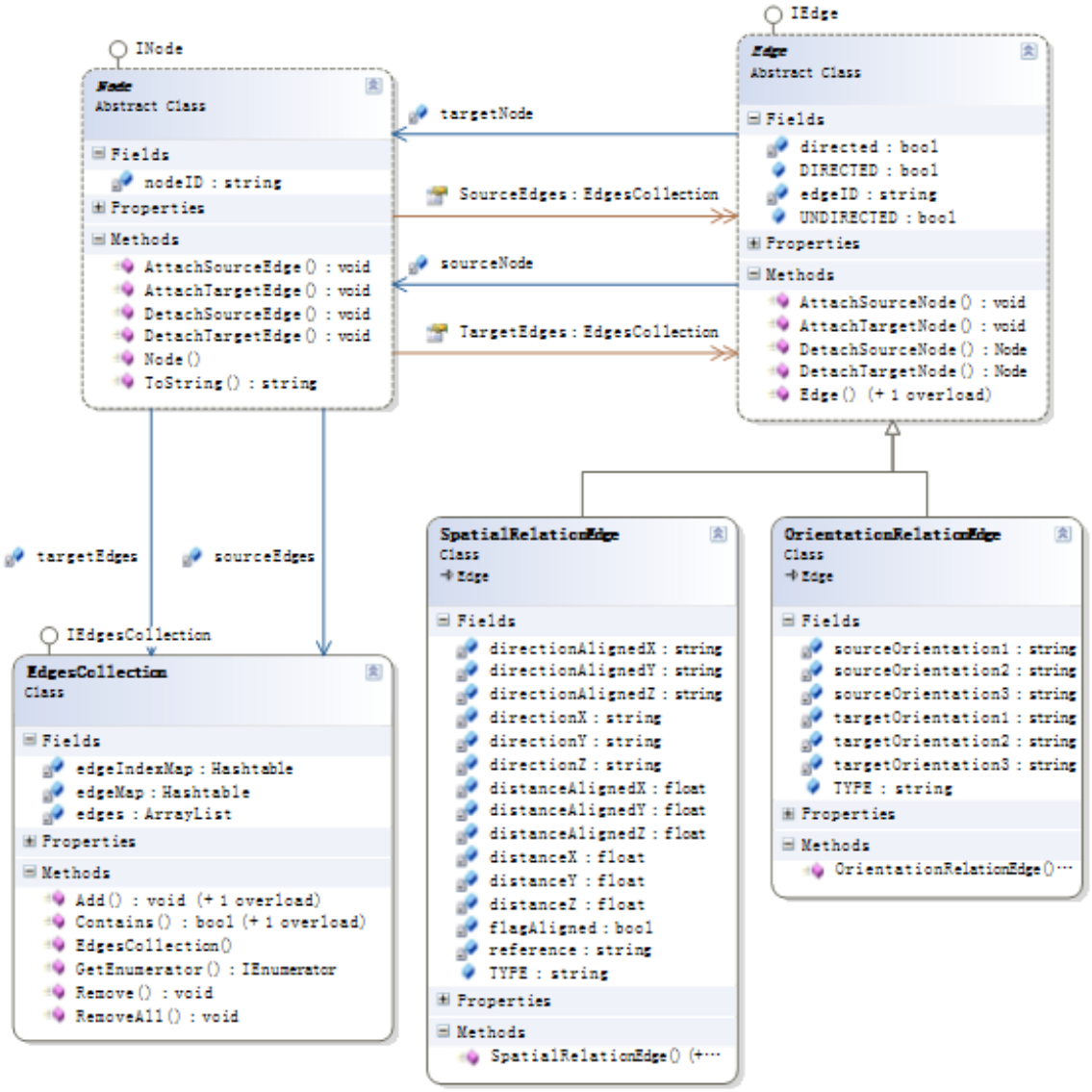

Figure 4.4: Classes derived from Edge and their relations

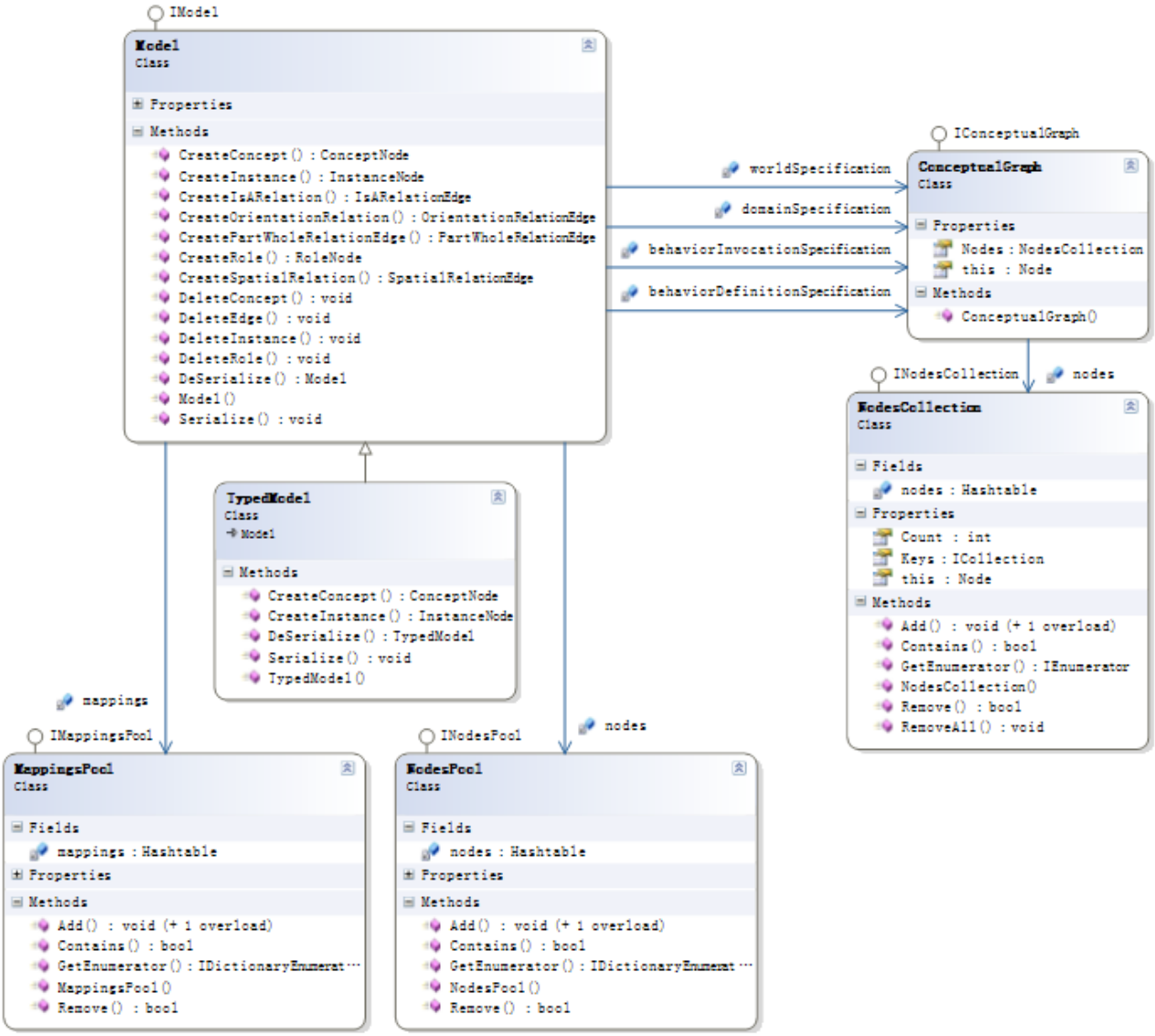

Figure 4.5: The Model and TypedModel classes

one kind of concept and instance nodes, while OntoShop has three of them: one for infrastructure, one for products and one for promotion objects. This *Model* can thus not be used in OntoShop. The solution was to derive a new class from *Model*, *Typed-Model* which overrides certain crucial functions like *CreateConcept* or *CreateInstance*, but uses the rest of the *Models* functionality.

4.5 Task integration

Integrating tasks is quite elaborated (see section 5.1.3). Task integration happens on two levels. The first level is rather intuitive and it is used by the domain expert. As stated earlier, all possible tasks are grouped in the *Shopping Task Ontology* and for every task, a web service can be found or created. The domain expert will thus only need to link tasks to web services. As a web service can be represented by a standard web address, all we need to do is to save in an external file the information about what web service is used by what task.

However, when an online customer is visiting a virtual shop and he, for instance decides to check-out, several processes will need to be activated in the background. It is needed to check whether the shopping cart is not empty and if not, whether the bought products are still in stock, along with other things. This part needs to be programmed. We will see by whom and how in section 5.1.3.

Chapter 5

Case study

In this chapter, a real case scenario is described where a shop with its products is built using our approach and tool.

5.1 Designing the shop

As described earlier, the shop designer will be responsible for starting a new project, adding all possible concepts, creating behaviors and mapping tasks to web services. We will explain how this happens.

5.1.1 Step 1: adding new designs

Figure 5.1 is a screenshot of OntoShop in *Designer View*. In this view, the shop designer is provided with a three step method to create the needed concepts, behaviors and tasks for the virtual shop. This method is outlined on the left part of the window, below the label *Designer View*. In the central panel, all the design objects are listed categorized by type: infrastructure, product and promotional objects. Finally, in the right part of the window, the different properties, mappings, attached behaviors and tasks are displayed.

To avoid being too abstract, a "concept" was renamed into a "design". In real life, a design is the engineering process that occurs before a project can be built. Something created on paper (or computer), but not really existing yet. In the same manner, by *Adding new designs*, the shop designer should intuitively understand that he is creating something that is not "really virtual" yet.

This first design step contains three sub-steps. In each of them, the shop designer

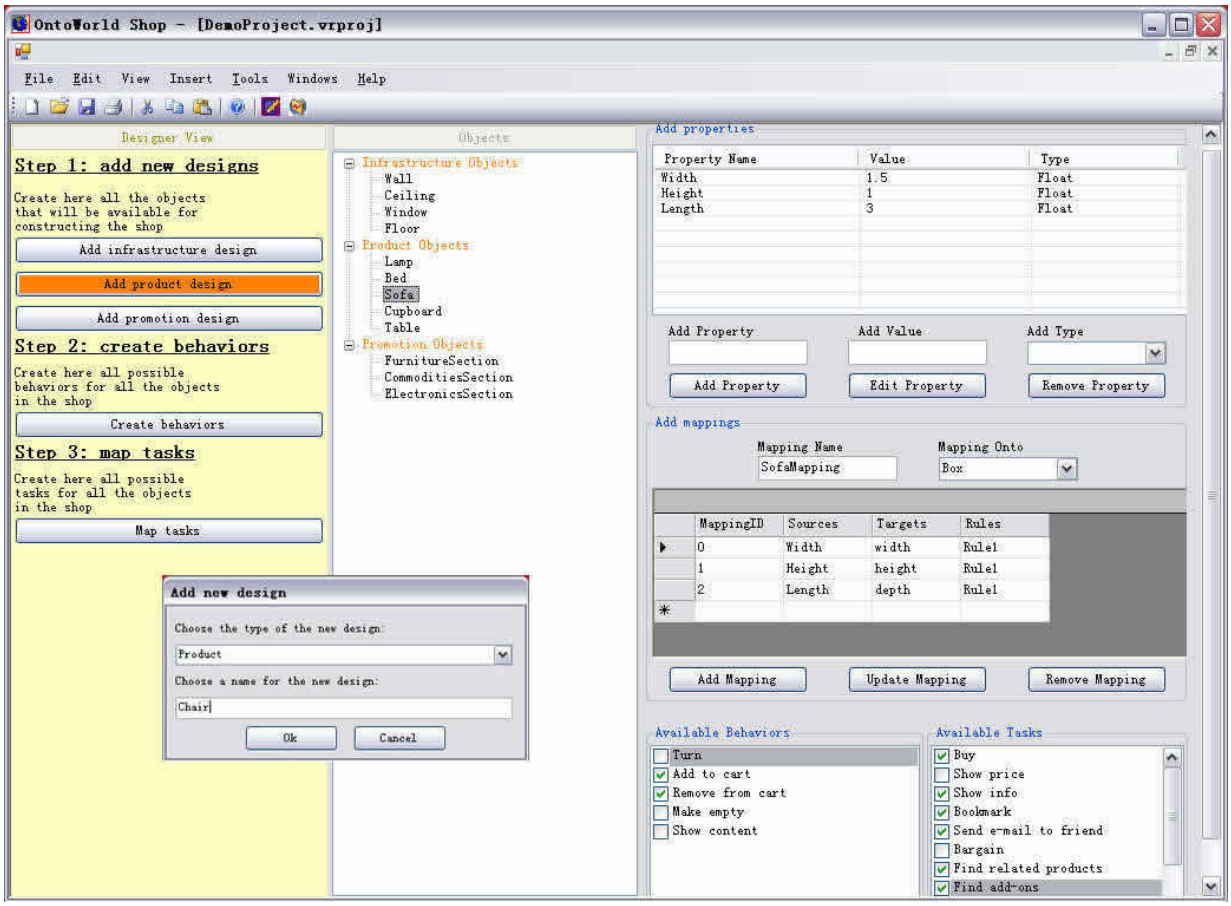

Figure 5.1: A shop designer's first step: create new designs

will add infrastructure, product and promotional object designs. Notice that these sub-steps are themselves correctly ordered: promotion and information panels or sheets can only be created (3rd sub-step) after some product designs have been added (2nd sub-step). Furthermore, some infrastructure concepts should be created (1st sub-step) before anything else. Nevertheless, this ordering is not required.

In figure 5.1, a new product design is being added, namely a chair. Inserting a new design can either be done by clicking the *Insert -¿ New Design* buttons or by directly clicking on one of the three buttons under *Step 1.* A light color (orange) has been used to emphasize visually the shop designer's position in the provided three step method.

In figure 5.1, the *Sofa* concept was selected in the list of available products. Its properties and other information are displayed in the right part of the window. A

Sofa has the following properties: a *Width, Height* and *Length* (floats) with default values of 1.5, 1 and 3. At the moment, each value we specify contains no units. This is because we made the assumptions that all the units in OntoShop are the SI Units.

To visualize the instances inside the virtual world, mappings need to be created. In the screenshot of figure 5.1, a mapping named *SofaMapping* is created. Based on the description given in section 3.2.2, the *Sofa* is mapped onto the VR primitive Box. The source property *Width* of the *Sofa* concept is mapped onto the target property width of the Box VR primitive. The same is done for *Height* and *Length*.

Finally, behaviors and tasks are attached to this specific product concept. Here for instance, a *Sofa* can either be added to the cart or removed from it, but it should not be able to turn (like a chair for example). This *Sofa* can also be bought or bookmarked. It can however not be bargained. For every concept he has created, the shop designer decides what functionality (i.e. tasks) should be attachable to these concepts. The decision about what tasks are actually attached to instances of these concepts is taken by the shop manager later on.

5.1.2 Step 2: creating new behaviors

In the previous paragraph, we talked about attaching existing behavior to specific instances. However, new behaviors can also be created. Behavior creation in OntoShop relies completely on the way behavior is created in OntoWorld. The GUI used is the same as well (cfr. Figure 2.6). more information can be found in [15].

5.1.3 Step 3: mapping tasks

The last important duty of a shop designer is to map tasks onto web services.

Figure 5.2 depicts how this happens. After choosing a specif concept (*Table* in this case), new tasks can be made available for it by clicking on *Add new...* in the *Available Tasks* panel. In that case a new window will prompt the user to choose one or more tasks between all the tasks available in the *Shopping Task Ontology*.

After that, these task will need to be mapped to some web service(s). In theory, it should be possible to build one web service that will handle all the tasks requests. However this is not very common nor flexible. Once tasks have been selected and mapped onto web services, some programming may need to be done by the IT expert so that those tasks are usable in the generated world.

When needed, this programming will be executed in the Java applet we talked about in section 3.2.3. Recall that to make the VR shop available through a web

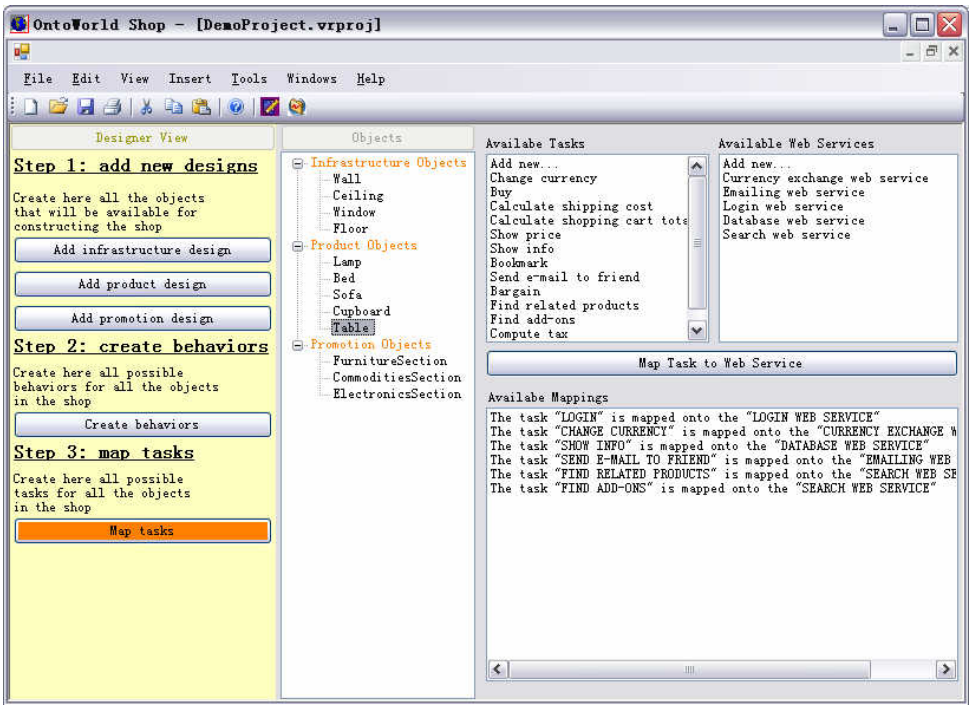

Figure 5.2: Third step of a shop designer: task mapping

browser, an applet embedded inside a html page has been created along with a plug-in player like Cortona [2]. The applet communicates with the plug-in via EAI [25] and the web services. Depending on the type of task to accomplish, the way to communicate between the applet and a web service may vary. In other words, a task can be implemented either in a *generic* or *specialized* way.

A *generic* implementation means that no matter what kind of Virtual E-Shop is being developed, the task will always be implemented in the same way. A good example is the *ConvertPrice* task. This task will always have three parameters: a price, a currency to convert from and a currency to convert to. For this reason, tasks requiring generic implementation are easily pre-built in OntoShop.

A *specialized* implementation is needed when there really is no possibility to decide in advance in what way a particular task should be performed. Therefore, there is no possibility to pre- implement it. This is where some programming is needed to be done by the IT expert. This technical part cannot be avoided. For instance, the task *Compute Tax* will be depending on the country where the shop owner is located. In Europe, this task would come to adding a well-known percentage of the price of the product. However, it may be that such a tax computation follows a completely

different formula in other countries.

In both cases (generic vs. specialized implementation), the code implementing a task will start to run as soon as a customer decides to execute it by using the applet. At that time, a connection will need to be established between the applet and one (or more) web service(s) and the needed parameters will need to be sent. To determine which web services to use for which task, it will be required to consult an external file where all mapping are stored. Recall that the shop designer did map tasks to web services in his third step (task mapping).

Actually, calling a web service may be seen as a remote function call. After parameters are (optionally) sent, the results can be fetched. At that time, it is again up to the applet to decide what needs to happen next. For example, when checking-out, the user may decide to pay for his products with a credit card. Before accepting, there is a need to check the credit card number. Therefore we connect the applet with a credit card checking web service, check the number and give some feedback to the user. Figure 5.3 shows the situation when a credit card number is wrong. The IT expert thus needs to program what feedback to give, what parameters to send and what payment options to allow (e.g., paypal, check, credit card,) among possibly other factors.

5.2 Managing the shop

Once all *designs* (i.e. concepts) are ready, it is up to the shop manager to decide which instances of products (called *items* in OntoShop) will actually be placed in the virtual mall. For this the three main steps in the approach need to be fulfilled by the shop manager.

5.2.1 Step 1: adding concrete items

In this step, the shop manager will need to create instances for all three kinds of concepts defined by the shop designer in his first step. For every newly created instance, mappings are set to the default concept properties (cfr. Section 3.2.1). Those mappings can be changed at the instance level using the Mappings panel located in the top right corner on the screenshot in figure 5.4.

In this figure, an instance *BestSofaInTheWorld* has been created. As can be seen in the *Relations* panel; it is an instance of the *Sofa* concept.

In the *Available Behaviors* and *Available Tasks* panel, tasks and behaviors can

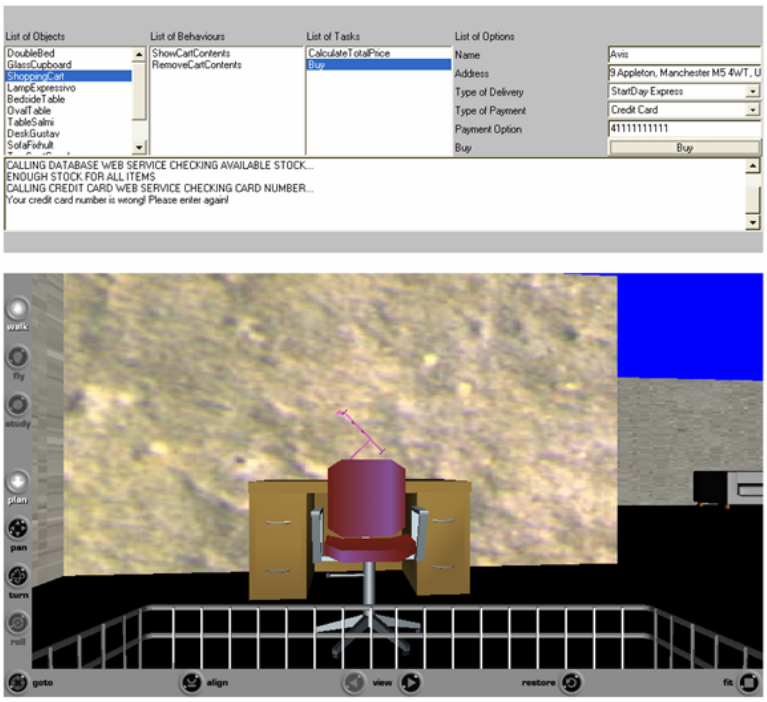

Figure 5.3: Checking the validity of a credit card by connecting to a web service

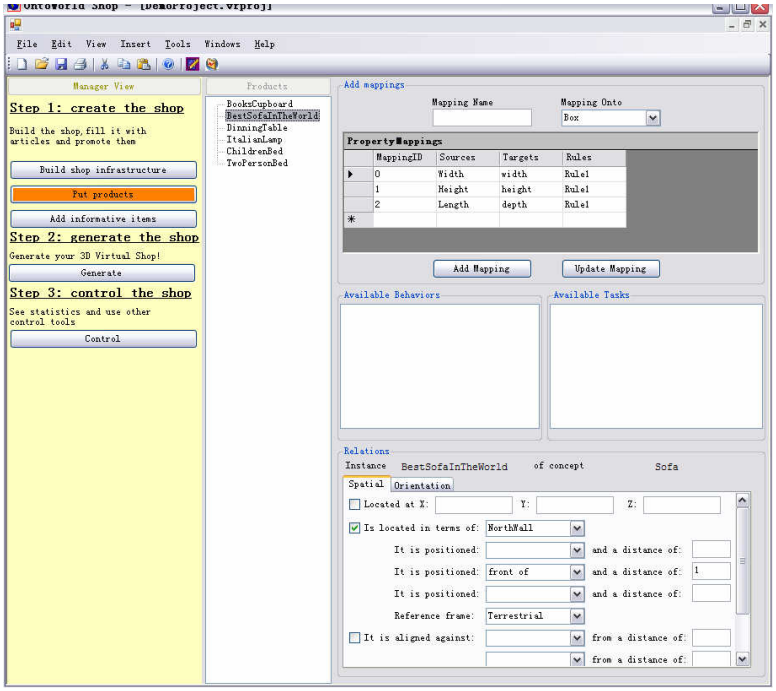

Figure 5.4: GUI of a shop manger

be attached to a specific item. Remember that the available tasks are chosen from at the previous level by the shop designer. A shop manager can decide whether to attach them or not. In this case, we attach all behaviors available for this product and all tasks except *Find related products* and *Find add-ons*. This is a decision made by the shop manager, as he may know that there are neither related products nor add-ons for this product and therefore those tasks are not needed here. Finally, in the bottom panel (*Relations*), instances are effectively placed into the shop, using high-level relations (cfr. section

5.2.2 Step 2: generating the shop

his step is actually merely a pressure on the button Generate. Section 3.2.3 explains what happens internally during the generation of the virtual shop and what file formats are exported.

5.2.3 Step 3: controlling the shop

An important duty of a shop manager is to control that shop is running as planned. In chapter 6 (further research), we will see what features could be helpful for increasing a virtual shop's profitability and volumes of sales. This controlling functionality has however not yet been incorporated in the present version of OntoShop.

5.3 Exploring the shop

This section presents the VR-shop elaborated with OntoShop. It concerns a fictive 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).

In figure 5.5, a view on our VR furniture shop is given. As it can be seen, the VR-shop contains walls and it 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. The second column provides the behaviors associated with each product. The third column gives the different shopping tasks that can be performed and the 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.

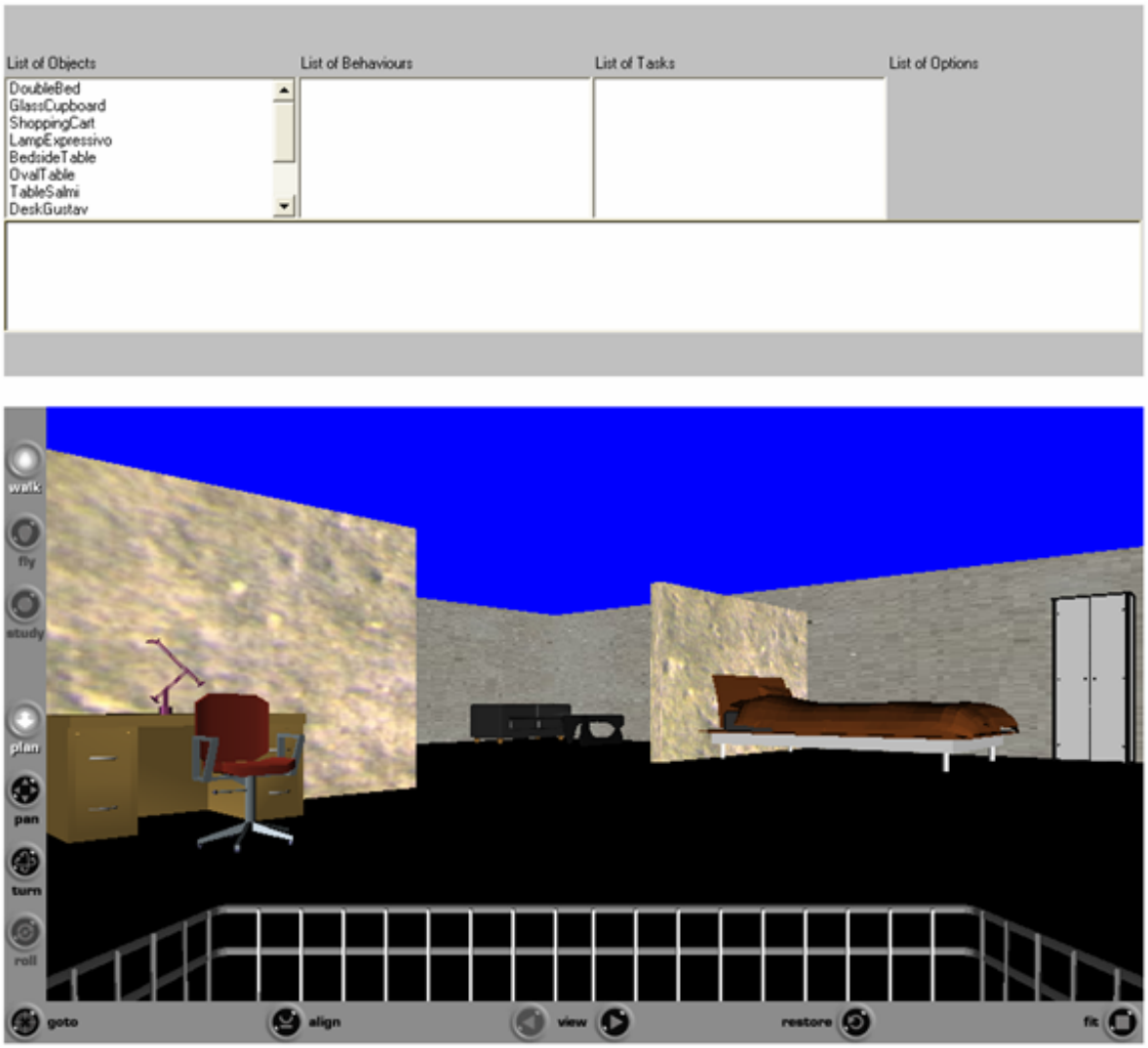

Figure 5.5: View of the VR furniture shop

The user can walk around in the shop by using the mouse or the interface provided by the VR-player. He can also obtain information by pointing to products seen in figure 5.6. If clicked, 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 task in the third column. In this specific case, the price of the chair can be converted into a different currency by choosing the *ConvertPrice* task. When the user selects this task, a web service for converting the price into different currencies [3] is called. Using this web service, the price of a product can be converted in any currency and in real-time. In figure 5.6, you can also see the result of converting the price of the chair to British Pounds. The price displayed inside the virtual world is updated too.

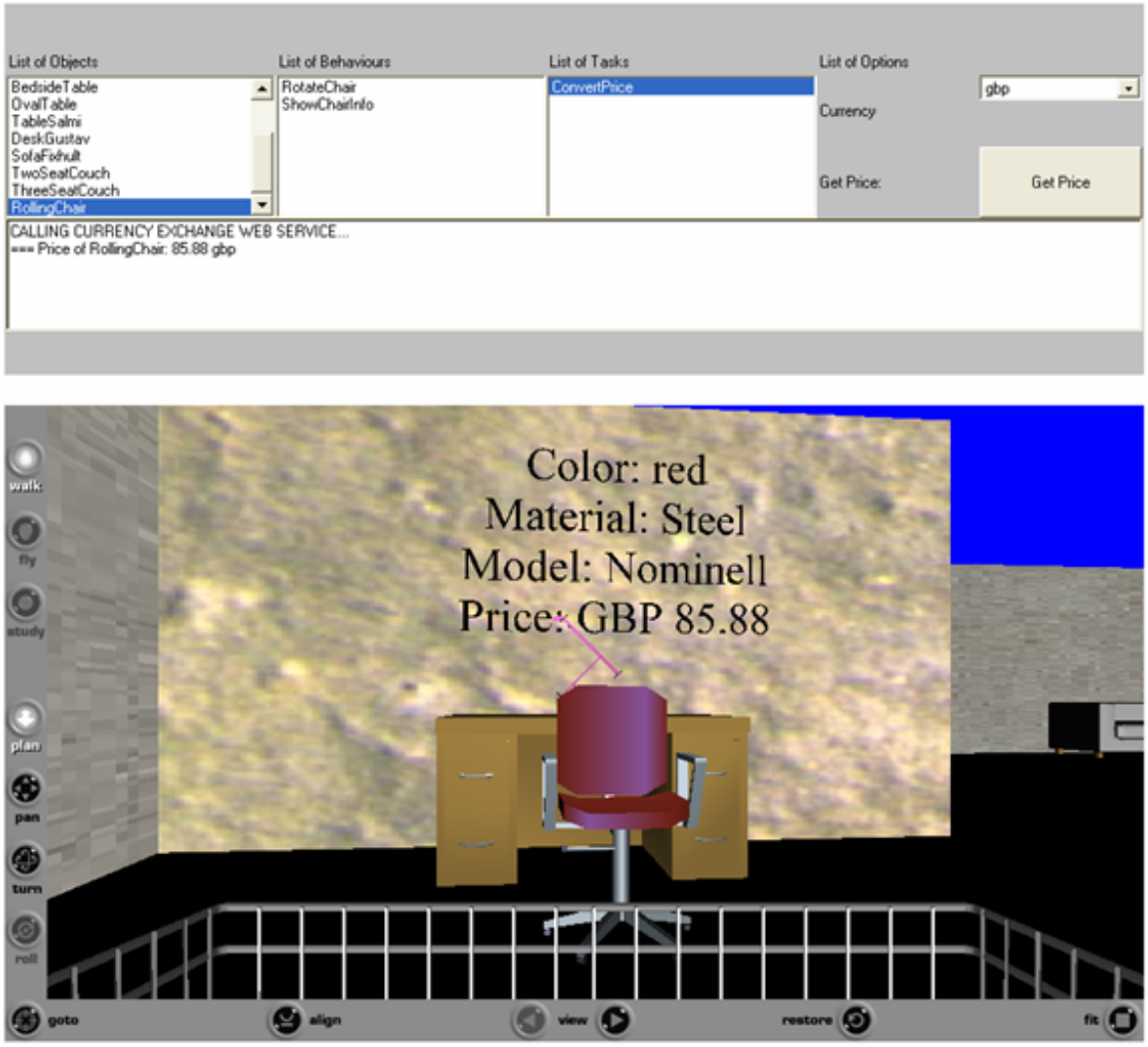

Figure 5.6: Displaying information and calling the task ConvertPrice for converting the price of the RollingChair in British Pounds

Chapter 6

Further Research

Research never stops. In this chapter, a number of further research topics are proposed to the interested reader.

6.1 Improving OntoShop

There are many ways to improve OntoShop and to let shop designers and managers generating virtual shops with even more ease, while in the same time extending their possibilities.

6.1.1 Visualize concepts and instances

It would be an interesting starting point, for instance, trying to make the definition of concepts and instances more visual. At present, OntoShop allows the generation of a small shop. If the user is perseverant, a middle sized shop can be created too. However, generating supermarkets will be very difficult. Such a shop would contain thousands kinds of products and thus tens of thousands of instances. In this situation, the whole application will not be controllable anymore. Shop manager will not remember the difference between all the different kinds of tables or other products available.

6.1.2 Create default instances

A second very useful research could be to not only define default concepts like those in the *VR Shop Ontology*, but to actually define default instances too. This would allow a shop manager to easily pick one available instance for the *Wall* or *Floor* concepts

Figure 6.1: Building a house in The Sims 2

without need to look for textures or edit the mappings. This situation could partly be compared with the process of building a house in the game *The Sims 2* [5]. Figure 6.1 depicts this process.

In this case, the user selects the kind of infrastructure (a concept thus) that he is going to add to the virtual world (a window here). After a number of different instances of windows appear (bottom right on the picture), the user then just needs to put the windows he likes on the walls. Of course, if this approach would to be integrated in OntoShop, we would still allow the user to create his own instances of concepts from scratch.

6.1.3 Back-up web services

Sometime, tasks are mapped to third-party web services. Third-party web services can go down. This dependency on others can actually be very dangerous for the prosperous grow of businesses using OntoShop to generate their virtual shop on the Web. It would be wise to allow users to map tasks not only to one web service, but also to some kind of back-up web services as well.

6.1.4 Enhance product creation

Shops with large numbers of products will need to be categorized in some way. Moreover, special information could apply to some products (e.g., containing allergen, ecologically clean, healthy product, low fat...). It will need to be stored systematically during the creation of the product in question.

Also, for easing the work of those businesses migrating from traditional E-shops to Virtual Reality E-shops, 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 (e.g., stored in a database or XML format) into the format required by the Product Ontology.

6.1.5 Other possible research to improve OntoShop

More research can be done to find out what other possible tasks and behaviors can be added to OntoShop. Maybe, it is also a good idea to allow users to add new tasks by directly editing the *Shopping Task Ontology* inside OntoShop. Of course, this would require more specialized people, as they would need to understand OWL.

A small issue is also to let users specify a unit for the entered values, when adding *Properties* to concepts, something that is not yet possible to do.

6.2 Improving virtual shops

6.2.1 Better GUI in virtual shops

The present GUI, which is as explained before a combination between control through the virtual shop and control through the java applet, is rather unintuitive. It makes the virtual world less real and attractive. Much better would be a built-in 3D control panel inside the virtual shop itself. In this way, it will be less disturbing, more

Figure 6.2: Chatting and meeting people in ActiveWorlds

appealing and the environment could even be browsed in full-screen, adding a great visual effect to the whole shopping experience!

6.2.2 Making virtual shops more natural and attractive

This could be achieved in a variety of ways: first, by adding avatars. Avatars refer to an interactive representation of humans in a virtual reality environment. These avatars can not only be seen as customers, but also as sellers or sales assistants. A second way to make the generated shops more real is to add background music and sounds. A third way is to allow virtual shops to become a place to meet people and not only a place to buy things. For instance, this can be done by allowing chatting. A very good example containing all these features is ActiveWorlds, (figure 6.2) a virtual world with hundreds of users and thousands of virtual places [1].

6.3 Improving Business

Finally, some very important research can be done for improving business activities of those using OntoShop.

6.3.1 Control generated shops

In the use case from the previous chapter, we have seen that our tool does not yet support any functionality to allow shop managers to control/manage their virtual shops, although a button *Control* has already been incorporated into the GUI.

OntoShop could be made more than a generator of virtual shops; it could be made a framework. A framework to build, control and improve E-shops or even all E-Commerce activities. Such *Control* functionality could be extremely interesting. Statistics will play an important role here. For instance, using statistics, a shop manager would be able to find out what places in the virtual shop are most visited, what paths are most used and what products are most popular. Undoubtedly, this would be a very important source of knowledge for every businessman.

6.3.2 Incorporate marketing strategies

We are also searching for a way to include marketing strategies and E-Commerce business models into the design process of VR-shops. These strategies will then allow to automatically place the products according to some well-specified rules to give a more profitable result. As an example, we can let shop managers decide whether a specific product should be massified or not, as explained in section 3.2.1 and [36].

Chapter 7

Conclusion

Today, E-commerce is everywhere. In particular, Online shopping has become a process that virtually all ages and demographics have come to embrace - for the very fact that it makes life simpler. Yet, many realize that customers want more.

The goal of this work was to create a way for building complex Virtual Shops in an intuitive way. To achieve this, we have extended the approach called VR-WISE approach which uses ontologies to help designers having no background in VR to build VR world using their own terminology. This has resulted in the implementation of a tool called "OntoShop". As the user may not be a VR expert, it was then necessary to make the GUIs very intuitive for these users. For this reason, the GUIs were really built around the shop manager and the shop builder. The tool allows the shop builder and the shop designer to build and to maintain a VR-shop from a high-level free from any implementation details. Using the conceptual specification made by either the shop manager or the shop builder, the actual "VR-shop" is then generated in a 3D file formats. At this moment, the tool supports X3D and VRML. Using these formats, the VR-shop can then be delivered through media like the internet.

The work in this thesis can be summarized into four main stages. The first stage was to customize the general VR-WISE approach to only the VR-Shop approach. The second stage was to implement a tool called "OntoShop" demonstrating the VR-Shop approach. The implementation did involve the creation of GUIs oriented towards the shop manager and the shop builder. Another challenge in the implementation was also to re-use and to extend the core of the "OntoWorld" tool which implements the "VR-WISE" approach. For this, a number of interfaces needed to be created. The third stage was to introduce the concept of task ontology which was not yet used in the approach and not yet implemented in the "OntoWorld" tool when the apprenticeship

started. OntoShop provides now a way to define tasks. To illustrate the tool, a scenario was built where a VR furniture shop has been created and in which the user can buy furniture online. Furthermore, we have predefined typical shop concepts used in almost any kind of virtual shops. For instance, the layout of the shop can easily be defined or changed because common VR-shop infrastructure concepts are predefined in the VR-SHOP Ontology. 3D virtual objects can be placed in the virtual shop by means of very intuitive high-level spatial and orientation relations. Typical behaviors have been created as well. Finally, more than 30 user tasks have been added and the idea to use web-services for the implementation of these tasks has been introduced. This allows functionalities to be added (or changed) to virtual shops. Finally, the fourth stage was about creating a VR-Shop using "OntoShop" and in which the user can buy goods and select different tasks such as to know the prices of products in different currencies or to be able to buy products.

This approach also provides another important advantage which is the maintenance of a VR-Shop. With our approach, the maintenance is very easy for a non VR expert. Products can be easily added, updated or removed because they are maintained separately and described by means of the Product Ontology. Resulting VR-shops are then easy to maintain and to adapt. This also means that adapted shops can be re-generated conveniently.

All these newly introduced concepts should allow an OntoShop's typical user to generate in a methodical way a complete electronic 3D store in reasonably short times. Practically, anyone can create attractive and original electronic virtual business and run those online. This leads to the obvious advantage of lowering costs and offering a competitive advantage compared to traditional E-Commerce.

We believe that our work can be a real benefit to the VR community looking at E-Commerce. We also believe that OntoShop can become a business success story on itself. If regularly updated with new features and if marketed in the right way, many buyers would be ready to acquire this technology.

Appendix A

ONTOWORLD SOFTWARE DESIGN

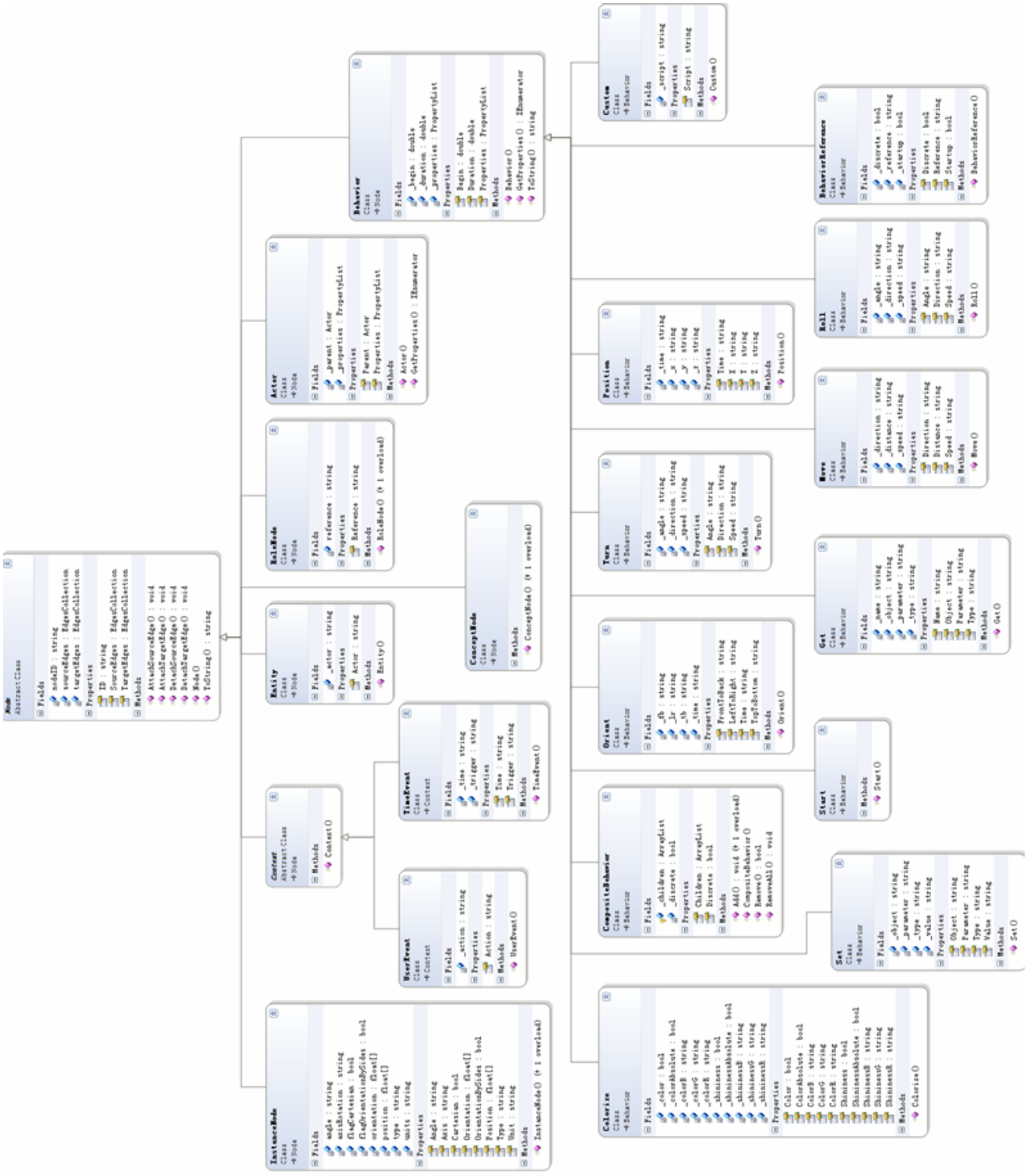

Figure A.1: Classes derived from Node

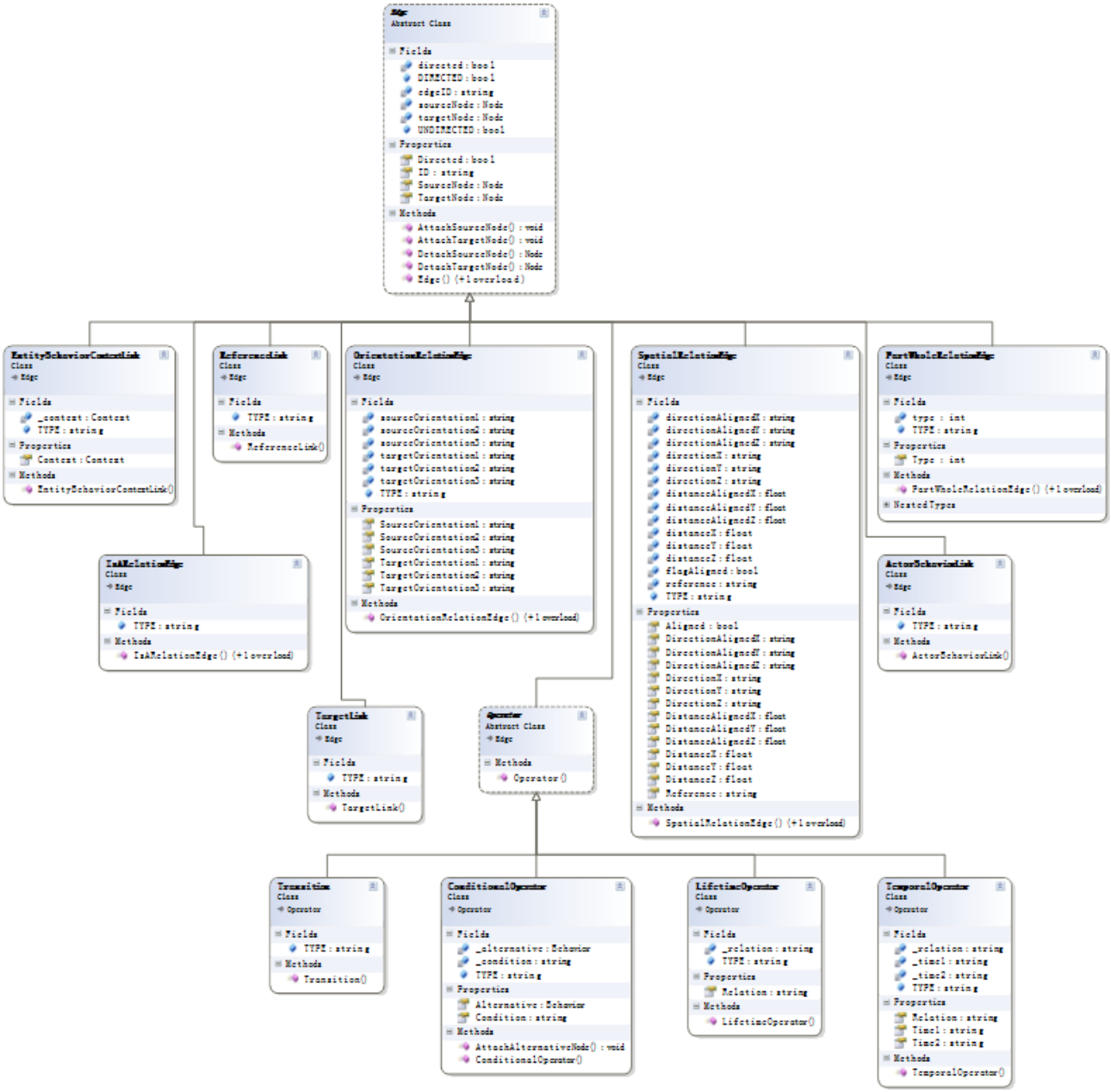

Figure A.2: Classes derived from Edge

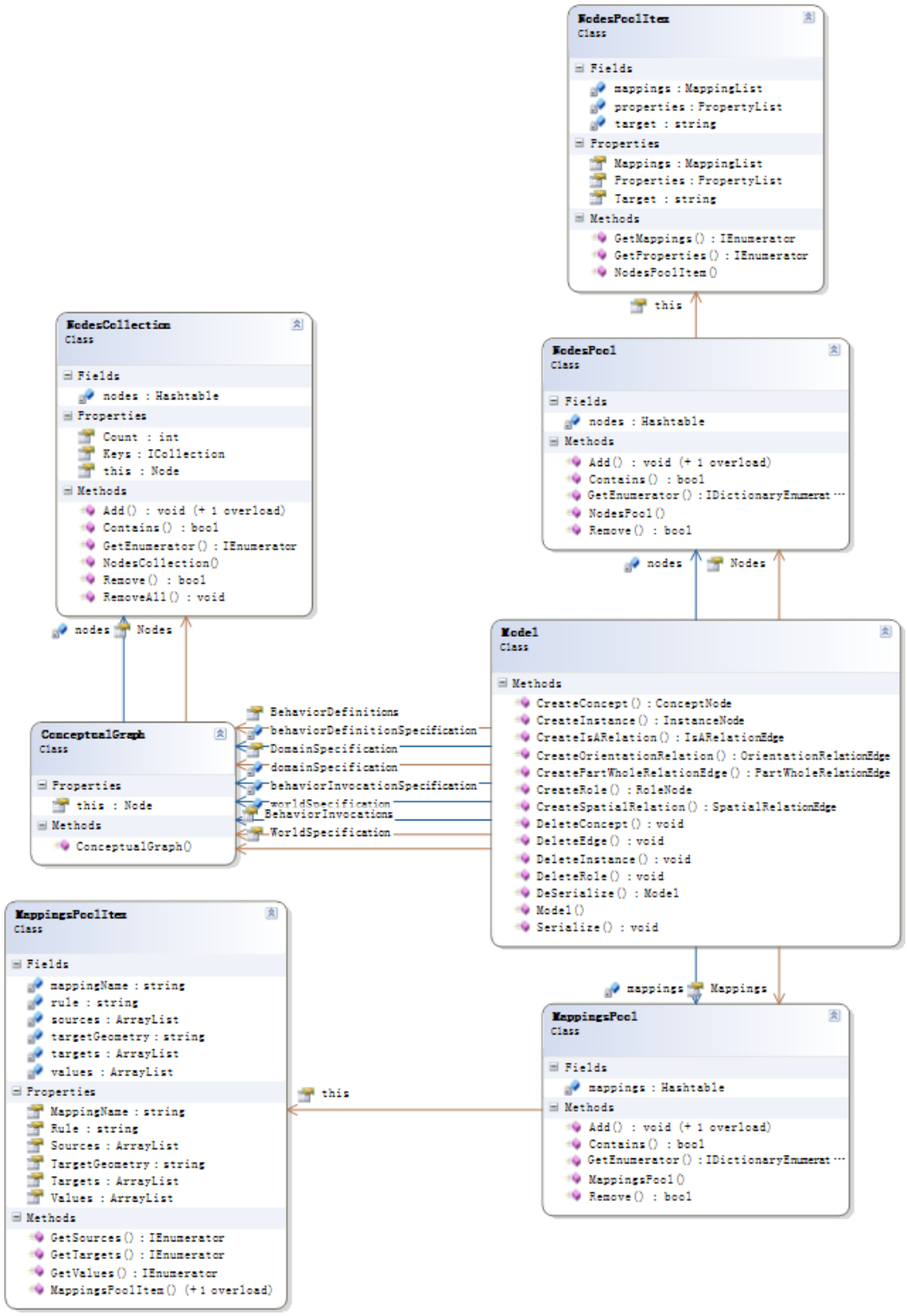

Figure A.3: Interaction with Model class

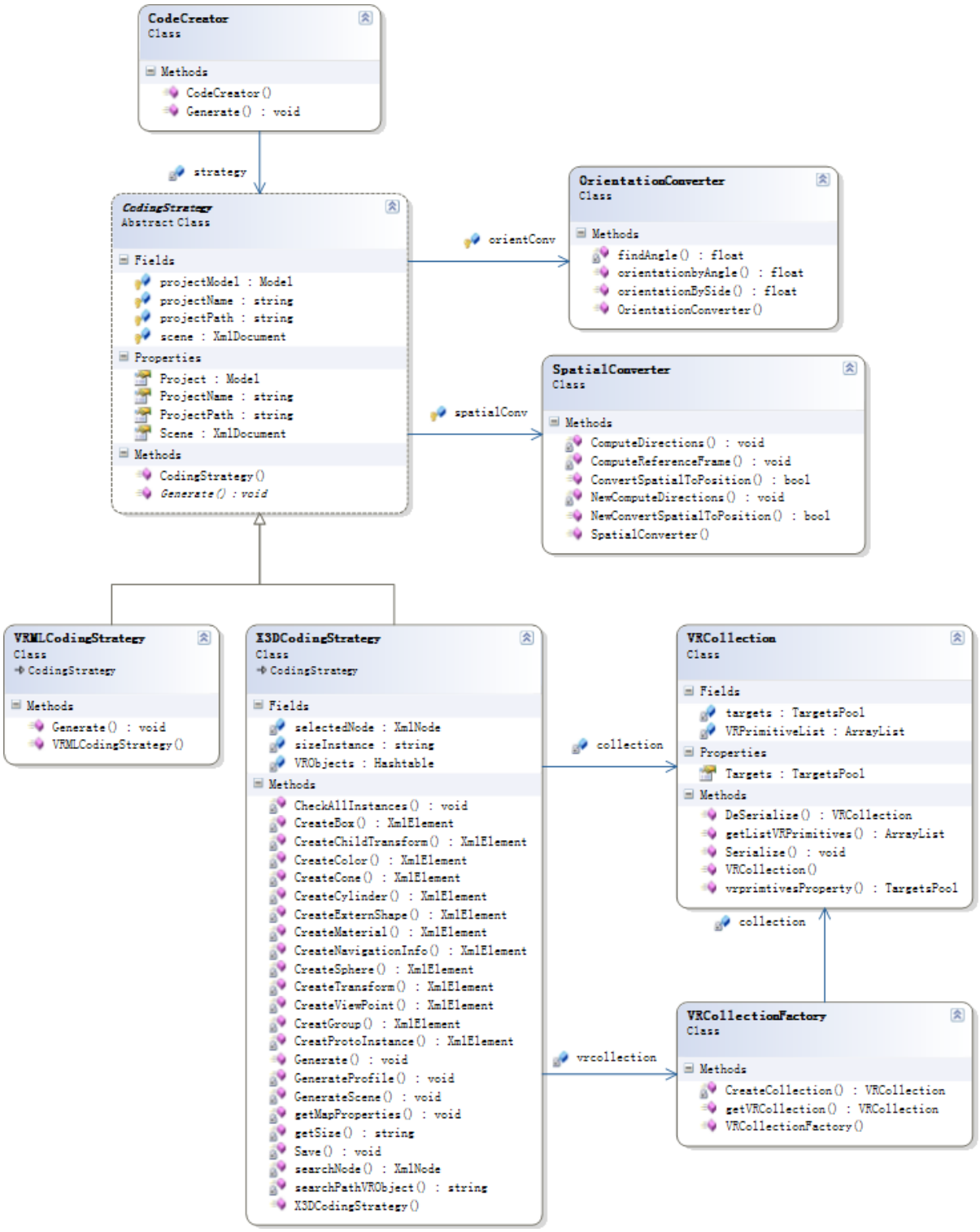

Figure A.4: Code generator class diagram

Bibliography

[1] Activeworlds. http://activeworlds.com/.

[2] Cortona. http://www.parallelgraphics.com/products/cortona/.

[3] Currency exchange web service. http://www.web servicex.net.

[4] Protege. http://protege.stanford.edu/.

[5] The sims 2. http://thesims2.ea.com/.

[6] Sketchup. http://sketchup.google.com/.

[7] Web services. http://www.w3.org/2002/ws/.

[8] Web3d. http://www.web3d.org/x3d/.

[9] E-commerce market size and trends. http://www.goecart.com/ecommerce_solutions_facts.asp, 2005.

[10] Uddi. http://www.uddi.org, 2005.

[11] Online retail sales to reach $329 billion by 2010. http://www.forrester.com/ER/Press/Release/0,1769,1033,00.html, 2006.

[12] Statistics: International online shopper. http://www.shop.org/learn/stats_intshop_general.asp, May 2006.

[13] Robles A., Molina J. P., Lpez Jaquero V., and Garca A. S. Even better than reality: The development of a 3-d online store that adapts to every user and every platform. In *HCI International 2005, Universal Access in HCI: Exploring New Interaction Environments*, volume 7, pages 10–20, Las Vegas, Nevada, USA, July 2005.

[14] Davide Booth, Hugo Haas, and Francis McCabe. Web services architecture. http://www.w3.org/TR/ws-arch/, 2005.

[15] Pellens Bram, De Troyer Olga, Bille Wesley, and Kleinermann Frederic. Conceptual modeling of object behavior in a virtual environment. In *In Proceedings of Virtual Concept 2005*, Biarritz, France, 2005.

[16] Pellens Bram, De Troyer Olga, Bille Wesley, Kleinermann Frederic, and Romero Raul. An ontology-driven approach for modeling behavior in virtual environment. 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*, pages 1215–1224, Agia Napa, Cyprus, October 2005.

[17] Pellens Bram, Bille Wesley, De Troyer Olga, and Kleinermann Frederic. Vr-wise: A conceptual modeling approach for virtual environments. In *In CD-ROM Proceedings of the Methods and Tools for Virtual Reality*, Ghent, Belgium, 2005.

[18] Gamma E., Helm R., Johnson R., and Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[19] Vladimir Fomenko. Virtual environments and web services, apprenticeship report, 2005. Vrije Universiteit Brussel.

[20] Haubl G. and Figueroa P. Interactive 3d presentations and buyer behaviors. Technical report, University of Alberta, Edmonton, 2002.

[21] Ucelli G., De Amicis R., and Conti G. Shape semantics and content management for industrial design and virtual styling. In *The Workshop towards Semantic Virtual Environment (SVE)*, pages 127–137, Switzerland, 2005.

[22] Chan H., Lee R., Dillon T., and Chang E. *E-Commerce - Fundamentals and Applications*. John Wiley & Sons, Ltd, Brisbane, 2002.

[23] Mansouri H. Using semantic descriptions for building and querying virtual environments. Licentiate's thesis, Vrije Universiteit Brussel, 2005.

[24] Cahners In-Stat. The changing market economy: Vertical industry and demographic profile of the small business market, July 2002.

[25] Hartman J. and Wernecke J. *The VRML 2.0 Handbook*. Addisson-Wesley, 1998.

[26] Chittaro L. and Ranon R. Virtual reality stores for 1-to-1 e-commerce. Technical report, Department of Mathematics and Computer Science, University of Udine, 2000.

[27] Chittaro L. and Ranon R. New directions for the design of virtual reality interfaces to e-commerce sites. In *In Proceeding of AVI 2002*, Trento, Italy, 2002.

[28] Pastore M. Online holiday shoppers to triple. http://cyberatlas.internet.com/markets/retailing/article/0,1323,6061_235331,00.html#table1, 2006.

[29] Guarino N. and Giaretta P. *Ontologies and knowledge bases: towards a terminological clarification*, chapter Towards Very Large Knowledge Bases: Knowledge Building Knowledge Sharing, pages 25–35. ION Press, 1995.

[30] De Troyer Olga, Bille Wesley, Romero Raul, and Stuer Peter. On generating virtual worlds from domain ontologies. In *In Proceedings of the 9th International Conference on Multimedia Modeling*, pages 279–294, Taipei, Taiwan, 2003.

[31] Salembier P. and Smith J. R. Mpeg-7 multimedia description schemes. In *IEEE transactions on circuits and systems for video technology*, volume 11, pages 748–789, 2001.

[32] Greenspan R. Another banner e-com year expected. http://www.clickz.com/stats/sectors/retailing/article.php/2196491, 2006.

[33] Forrester Research. How to measure what matters. http://www.forrester.com/ER/Research/Report/Summary/0,1338,10069,00.html, 2006.

[34] Gruber T.R. *A translation approach to portable ontologies*, chapter Knowledge Acquisition, pages 199–220. Elsevier, 1993.

[35] Bille Wesley, De Troyer Olga, Pellens Bram, and Kleinermann Frederic. Conceptual modeling of articulated bodies in virtual environments. In *The 11th International Conference on Virtual Systems and Multimedia*, pages 17–26, Gent, Belgium, 2005. Publ. Archaeolingua.

[36] Bille Wesley, De Troyer Olga, Kleinermann Frederic, Pellens Bram, and Romero Raul. Using ontologies to build virtual words for the web. In *The International Conference on WWW/Internet 2004*, pages 20–28, Madrid, Spain, October 2004.