Vrije Universiteit Brussel
Faculteit Wetenschappen
Departement Informatica en Toegepaste Informatica

The use of Semantic Mappings and Ontologies to enrich Virtual Environments

Thesis submitted in partial forfilment of the requirements of the degree of Master in Applied Computer Scienes.

By: Barry Nauta
Promotor: Prof. Dr. Olga De Troyer
Supervisor: Dr. Frederic Kleinermann
2009 - 2010

Patience is a virtue: possess it if you can, seldom found in women, never found in men.

A mes trois femmes préférées:
Geneviève, Elizan et Rosalie

Merci pour toute votre patience!

Abstract

The web is continuously evolving, web 3.0 is on our doorstep. While the predictions differ, most definitions agree about the next evolution in the web. It will include the semantic web. Where the current web is for human consumption only, the next evolution of the web will allow machine-to-machine communication, enabling better processing of the enormous amount of data that is available. Web 3D is also gaining momentum, with native support (removing the need for plugins) in webpages on the way, leading to virtual worlds, either realistic or game oriented. This thesis looks at the possibilities to combine these two technologies.

We describe how we use the semantic web, via the use of ontologies, as base to construct the virtual world, which means that the information on virtual objects which we would like to display come from our ontology. Via a semantic mapping, we link this information to a 3-dimensional representation, in other words: we *know* what we are displaying.

Afterwards we show how we can modify this virtual world, in real-time, with external information. External information is often available in legacy formats, so we use translators. This way we can access the external information in the same way as we access our primary ontology.

Finally, we will not only use the ontology to provide us with information on virtual objects to display in the virtual world, we will also use it as a dedicated search engine and let it guide us from starting point to our destination, using a graph and graph search-algorithms to guide us from starting point to our destination, based on the query towards the ontology.

Samenvatting (in Dutch)

Het web evolueert constant, web 3.0 staat voor de deur. Alhoewel veel definities verschillen zijn de meeste defitities het erover eens dat het semantische web onderdeel is van de volgende evolutie van het web. Het huidige web is gemaakt voor menselijke consumptie, de volgende evolutie van het web zal machine-naar-machine communicatie mogelijk maken en hiermee ook betere verwerking van de enorme hoeveelheid data dat momenteel beschikbaar is. Web 3D wint ook momentum met in-browser ondersteuning zodat er geen plugins meer nodig zijn. Dit eindwerk bekijkt de mogelijkheden om deze technologiën te combineren.

We beschrijven hoe we, door middel van ontologiën, het semantisch web gebruiken als basis om onze virtuele wereld te bouwen. Dit houdt in dat de informatie over de virtuele objecten die we tonen van onze ontologie komt. Gebruik makend van een semantische mapping linken we de betreffende informatie aan zijn 3-dimensionale representatie. Ofwel, we *weten* wat we aan de gebruiker tonen.

Daarna tonen we hoe we deze virtuele wereld in real-time aan kunnen passen met externe informatie. Deze informatie is vaak beschikbaar in een oud formaat, zodat deze eerst vertaald moet worden. Op deze manier kunnen we deze informatie op dezelfde manier gebruiken als onze ontologie.

Als laatste gebruiken we de ontologie niet alleen om informatie te verkrijgen over de virtuele objecten die we tonen, we gebruiken dezelfde ontologie om ons de weg te tonen in de virtuele wereld. Gebasseerd op het antwoord van de vraag aan de ontologie maken we gebruik van een zoek-algoritme dat ons van het begin-punt naar het eind-punt leidt.

Acknowledgements

A while ago, I decided to pick up studies again, to correct opportunities I missed in the past. I have often asked myself the question why, but the choice was made, the path was taken.

Combining a full-time job, family and studies have been very challenging, and I could not have done this without the support of my wife, Geneviève and my two daughters, Elizan and Rosalie. The countless hours I spent in books and behind my computer has weighed on them. Finishing my studies could not have been possible without their continuous support and love.

I would like to take the opportunity to thank Prof. Dr. Olga De Troyer, for giving me the opportunity to realize this thesis. Furthermore, my gratitude goes to Dr. Frederic Kleinermann who has supported, advised me and proof-read my work several times. It is thanks to his guidance that I have been able to write this document.

The end of this path is near, I'm coming home.

Glossary

This chapter contains a glossary of terms and abbreviations that are used throughout this document

Agent
An Agent is just something that acts. But computer agents are expected to have other attributes that distinguish them from mere "programs", such as operating under autonomous control, perceiving their environment, persisting over a prolonged time period, adapting to change and being capable of taking on another's goal. [45]

AJAX
AJAX stands for "Asynchronous Javascript And XML". It is a group of related technologies that enable interactive webapplications. Using AJAX, web-applications can retrieve data from a server in an asynchronous, non-blocking manner, and update only a part of the webpage afterwards.

API
API stands for "Application Programming Interface", an API is an interface implemented by an application, enabling other applications to interact with it.

AR
AR stands for "Augmented Reality" and describes the technology to view a real-world environment whose elements are augmented by virtual computer-generated imagery. It is used to enhance a users perception of reality.

Avatar
An avatar is digital representation, often in the form of a one-dimensional username, a 2-dimensional image or 3-dimensional model.

Collada
Collada stands for "COLLAborative Design Activity" and establishes an interchange file format for interactive 3-dimensional applications.

CSS
CSS stands for "Cascading StyleSheet" and is a stylesheet language primarily used to style web-pages that are written in HTML.

CTT
CTT stands for "Concurrent Task Trees", a notation used for task modeling. It is designed to overcome limitations of other notations that are used to design interactive applications.

DTD
DTD stands for "Data Type Definition", it defines the legal structure of an XML document, by listing its legal elements and attributes.

Flash
Adobe Flash is a multimedia platform used for adding interactivity, video and animations to web-pages. Adobe Flex, a platform to develop Rich Internet Applications is based on Adobe Flex.

Flex
Adobe Flex is a Software Development Kit (SDK), release by Adobe Systems for development of Rich Internet Applications (RIAs), based on the Adobe Flash platform.

Folksonomy
A Folksonomy is a collaborative classification, typically established by letting users add tags to resources. Another term for folksonomies is "collaborative tagging".

HTML
HTML stands for "HyperText Markup Language", it describes a markup language, a structured way of creating documents, for web-pages.

HTTP
HTTP stands for "HyperText Transfer Protocol", it is a protocol used to distribute information using the internet. It is typically used for, but not limited to, distributing web-pages, videos, images etc from a server to a client.

HUD
HUD stands for “Head-Up Display” HUDs provide a 2d component with data on a semi-transparent canvas so it doesn’t obstruct the users view.

IRI
IRI stands for “Internationalized Resource Identifier”. IRIs are a complement to URIs, using unicode.

Java
Java is a name for a number of products, it is the name of an object-oriented programming language, but it is also often used as name for the enterprise platform which is also known under the name JEE (Java Enterprise Edition)

Javascript
Javascript is a scripting language, often used in webpages (client-side Javascript), providing enhanced user-interfaces and dynamic websites. Javascript is unrelated to Java.

JavaFX
JavaFX is a Java platform for delivering Rich Internet Applications (RIAs) that are cross-platform.

Jena
Jena is a Java framework for building semantic web applications.

Joseki
Joseki is an HTTP engine, that supports SPARQL queries. It is based on Jena.

JSON
JSON stands for “JavaScript Object Notation”, it is a standard for data interchange, derived from the Javascript language. Although it is derived from Javascript, it is language independant.

Landmark
A Landmark is a position, typically in a Virtual Environment, that besides a position, also has an orientation, making them useful for navigation.

Mashups
A Mashup is basically a webpage or an application that combines information from mulitple sources to create a new service.

Metadata
Metadata is "data about data", often used to indicate *what* data it is.

O3D
O3D stands for "Open 3D", it is an opensource standard, created by Google, for interactive 3D applications.

Ontology
An ontology is a formal representation of knowledge, it is used to reason this knowledge.

OWL
OWL stands for "Web Ontology Language", it is a family of languages used to query ontologies.

POC
POC stands for "Proof of Concept", a short (and often incomplete) realization of a certain method of idea, used to demonstrate the principle.

POI
POI stands for "Point Of Interest", it indicates an interesting location, in 3d worlds, they are often attached to viewpoints.

RDF
RDF stands for "Resource Description Framework". It is a family of W3C specifications, designed as a metadata data-model.

RIA
RIA stands for "Rich Internet Application", the term describes webpages that have characteristics of desktop applications. Adobe Flex (Flash), JavaFX and Microsoft Silverlight are the biggest players in this area.

RIF
RIF stands for "Rule Interchange Format", an XML language for expressing rules which computers can execute

SAI
SAI stands for "Scene Authoring Interface", a Javascript API that allows the user to interact with the embedded x3d scene in a web-page.

Semantic Web
The semantic web represents an evolution of the web in which the semantics of data are defined, making it possible for machines to properly process it.

Silverlight
Microsoft Silverlight is a web application framework, enabling the creation of Rich Internet Applications (RIAs), ,similar to Adobe Flash and JavaFX.

SPARQL
SPARQL stands for "SPARQL Protocol and RDF Query Language" It is an RDF query language.

Taxonomy
A Taxonomy is a classification of objects/things in a hierarchical way (typically a tree-like structure).

URI
URI stands for "Uniform Resource Identifier" (RFC2396), a string of characters used for identifying a resource. A URI can either be a URL (Uniform Resource Locator) or a URN (Unified Resource Name).

URL
URL stands for "Uniform Resource Locator", a URL is, like a URN, a URI (Unified Resource Indicator). In this case, it is a URI that specifies where the resource can be located. It is typically used as locator of a web-address.

URN
URN stands for "Uniform Resource Name", and is, like URL, a URI (Unified Resource Indicator). It does not imply the availability of the identified resource. The functional requirements for URNs are described in RFC 1737.

VE
VE stands for "Virtual Environment", a computer simulated environment. The term is closely related to Virtual World (VW) and Virtual Reality (VR).

VR
VR stands for "Virtual Reality", it is a term that applies to computer simulated environments that mimic either real-life or imaginary places.

VRML
VRML stands for "Virtual Reality Modeling Language" and is a standard file format for representing 3D objects, especially designed for the World Wide Web. It has been superseded by X3D.

VW
VW stands for "Virtual World", it is a computer simulated environment, through which users can interact and create objects, the term has become a synonym for 3D Virtual Environments. Some, but not all VWs allow multiple users.

W3C
W3C stands for "World Wide Web Consortium", it is the principle international standards organization for the World Wide Web (WWW).

Web 2.0
Web 2.0 is a term that is often used to indicate web-applications that allow information sharing, user-centric design, collaboration and more.

Web 3.0
Web 3.0 represents the future of the web as we currently know it. Many definitions of web 3.0 currently exist, but the most common include the semantic web, some others also include web 3d.

WWW
WWW stands for "World Wide Web", commonly known as "the web". It represents a system of interlinked hypertext documents, accessible via the Internet.

X3D
X3D stands for eXtensible 3D, an open standards file format and runtime architecture to represent and communicate 3D scenes and objects using XML. It is the successor of VRML. X3D is an ISO standard.

XHTML
XHTML is XML based language, it is an extension to HTML, used to write webpages.

XML
XML stands for eXtensible Markup Language and is a language that describes data/documents in a form of elements and attributes. When comparing to HTML, we could say that XML is the language that describes *what the data is*, where HTML describes *how the data looks.*

XMLSchema
XML Schema, much like DTDs, describe the structure of an XML document.

XQuery
XQuery is an XML Query language. XQuery can be used to query (finding and extracting) XML files for its data (elements and attributes), it is built on top of XPath expression.

XPath
XPath stands for XML Path Language and is a language that is used by XSLT to access (or refer to) specific parts in an XML document. [1]

XSLT
XSLT stands for eXtensible Stylesheet Language Transformation; an XML based stylesheet that defines the transformation from one XML document to another document (which can be another XML document, but it can also be CSV-based, PDF etc).

[1] XPath is also used by XLink ("XML Linking"), which is a specification that allows elements to be inserted into XML documents in order to describe links between resources.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The web is a big resource of information, ranging from text, videos, 2-dimensional images to 3-dimensional virtual worlds. It has come from a static web, where a lot of textual information was available, to a dynamic web. People used to browse the information, now they are contributing and increase the information that is available. With the web becoming more dynamic, people started building social networks for both pleasure and professional activities. When we take into account that the number of users on the web is still increasing, it becomes clear that the amount of information of the web increases rapidly.

The increasing information leads to new challenges. How do we sort this information, how do we visualize it? What can we do to help the users finding information, other than keyword matching which is the base for most search-engines at the moment.

To answer to these challenges, new techniques and approaches have been developed and researched. In fact, a lot of research is still being done. These techniques and approaches are grouped under the label of "semantic web technology", techniques that help to classify and interpret information that is available on the web, so that machines can understand the meaning and reason on this information.

Besides this, we are also assisting to the possibility to have Virtual Environments (VEs) over the web, due to new technologies. Virtual Environments are 3-dimensional virtual worlds, containing 3d objects in which users can navigate and interact with objects or other users. The most famous VE is Second Life. Second Life is an exceptional case, since most VEs have never been very success-

ful and even Second Life is no longer as successful as it used to be. Some state that the failure of success of VEs is related to the fact that they are visually not as attractive as video games. While this might be partially true, the visual quality has improved a lot, we can now have attractive VEs. VEs are also very expensive to develop and are therefor not accessible to web authors, this might be another reason for the lack of success.

The fact remains that the web is broadcasting information continuously, and a lot of this information is about social networks. This implies that VEs must incorporate this information into their environments in order to have a chance to be used over the web. Besides the ability to use this information, VEs must also adapt to what the user does. In other words, some kind of real-time customization should exist. This will facilitate the way users use the VE. A general view should be that VEs should be developed in the spirit of mashups, where users can combine the different types of information and use it to customize VEs and to display information that is relevant to potential users.

The aim of this thesis is to explore how semantic web technologies can be used to provide this facility to webbased VEs. This exploration is done by implementing a concrete scenario, based on the campus of the “Vrije Universiteit Brussel”.

1.1 Aim of this thesis

The aim of this thesis is to explore how we can enrich VEs with information so we can have more user specific VEs, with information coming from web-resources and how we can help users to find his way through this information in a 3-dimensional environment.

This is a challenging task, for several reasons. The first challenge is related to the fact that 3-dimensional information contains information on shapes only. This information tells the computer how to display the information from a geometrical and material point of view. The computer cannot provide any meaning to what it displays and therefor it is up to the user to derive this based on shapes and context. The second challenge is to have a VE having a virtual world that adapts to both the user and the external information, which can be obtained in real-time and typically changes rapidly. We can create a new source of information by combining different sources, the question remains how we can use and visualize this information inside the VE without completely rewriting it. With this increasing amount

of information, the third challenge is how users can navigate in a convenient way. This thesis will introduce an approach that addresses some of these challenges using semantic web technologies with the aim of having VEs that can adapt quickly to new information to be used.

1.2 Structure of this thesis

This thesis is structured as follows:

1. Chapter 1 provides an introduction to the context in which the research work is being conducted.
2. Chapter 2 provides related work where enabling technologies and research results are discussed.
3. Chapter 3 describes an approach to have dynamic Virtual Environments, based on ontologies.
4. Chapter 4 explains a way we use the ontology to improve navigational issues.
5. Chapter 5 provides a case study, based on the campus of the "Vrije Universiteit Brussel".
6. Chapter 6 discusses the overall limitations of our approach.
7. Chapter 7 provides a conclusion and provides some ideas for future work.

Chapter 2

Related work - Setting the Scene

This chapter provides related work, it is split into two parts. The first part explains existing technologies that are used in this thesis, the second part of this chapter discusses related work, both scientific and industrial, to the contents of this thesis.

2.1 Technology

This section highlights some of the technologies that play a role ranging from an Internet point-of-view to more specific web-related technologies and 3d technologies.

Besides discussing the current technologies, this section also mentiones previous technologies that have led to the current state of technology. Furthermore, it describes the emerging technologies which are used in this thesis in more detail.

2.1.1 The World Wide Web

The Internet, or "the world wide web", or "the web" [1] as we often refer to it, has come a long way.

Web 1.0

When the World Wide Web emerged (the first proposal for HyperText was released on November, 12th in 1990 [50]), it contained a lot of hyperlinked, informative pages. These pages were all static in content, there was a lot of information available on the web but understandable only by humans. We now sometimes call this version of the Internet "Web 1.0".

The rise of Java, and more specifically: Java applets, showed that some interaction with web-pages was possible. Starting with small games, dynamic navigation afterwards and even smaller application within pages later on. These technology changes can be seen as the evolution of the web, creating a broader platform to work with.

Slowly, programming languages leveraged the ability to create dynamic pages in a relatively easy way, based on content coming from other sources, like databases, but also coming from user input and more. Dedicated web-frameworks accelerated web-development and at the same time, client-side Javascript, a scripting language often embedded in web-pages which became more and more popular for basic web-page interaction. Internet slowly became a commodity; people started interacting, expressing themselves on the web using forums, taking a web-identity. These were the first indications that something was about to happen. Web 2.0 was on the horizon.

The term Web 2.0, was first used by Darcy DuNucci in 1999 [23]:

[1] We often see the Internet and "The web" or "The World Wide Web" (WWW) as the same thing. It is important to realize that what we call "The web" refers to only a subset of what we call the Internet. "The web" refers to web-pages, or from a more technical point of view, namely information in the form of web-pages, shared via the HTTP protocol.

The Internet embodies a family of network related protocols like HTTP (principal protocol to send web-pages - displayable information across the Internet), FTP (File Transfer Protocol - principle protocol to send files over the net), SMTP (Simple Mail Transfer Protocol - sending mail) etc.

> The Web we know now, which loads into a browser window in essentially static screenfulls, is only an embryo of the Web to come. The first glimmerings of Web 2.0 are beginning to appear, and we are just starting to see how that embryo might develop. The Web will be understood not as screenfulls of text and graphics but as a transport mechanism, the ether through which interactivity happens.

The term "web 2.0" became popular after Tim O'Reilly [39], hosted the Web 2.0 conference for the first time in 2004

Web 2.0

Web 2.0 does not relate to a new version of specifications; instead it can be seen as a combination of several evolutions from a technical point of view. From a business point-of view however, it can be seen as a revolution. The web is no longer about information only, it is now also about collaboration, the web as a platform, rich user experience and more.

> Tim Berners-Lee calls the term a "piece of jargon".
> He argues that the web was designed to do exactly those things that now are called web 2.0 and that nobody even knows what web 2.0 means [17]

Web 2.0 is about collaboration, information sharing, user-centric design, allowing its users to be active; interact with each other as contributors to websites, where before, users could only passively view information on web-pages.

Prashant Sharma [48] describes 7 features of web 2.0:

- **User-centered design**
  Customizable pages, fitted to the need of the user. One of the typical examples is iGoogle, offering a personalized Google page, where users can add news, weather information, photos and more.
- **Crowd-sourcing**
  Millions of contributions give a website a higher relevance. Typical exam-

ples include blogging platforms, like Blogger [2] and Wordpress [3] that beat conventional media company by producing extremely frequent and relevant content.

- **Web as platform**
  Web-applications replace more and more desktop functionality. Those web-applications are platform-independent and don't require specific client-side downloads. Google Maps is an excellent example of this aspect.

- **Collaboration**
  One of the most often indicated 'features' of web 2.0 is collaboration. Collaborative information outpaces traditional information sources. A typical example is Wikipedia that provide better and more content than traditional encyclopedia.

- **Power decentralization**
  Web 2.0 follows a self-service model instead of an administrator dependant model. A typical example is Google Adsense where users setup their own advertisement platform without administrator interventions needed.

- **Dynamic content**
  Web 2.0 is also about highly dynamic content. The user-provided information that was listed in crowd-sourcing can be used to lift a site's prestige, it can also help to influence the content.

- **SaaS**
  "Cloud application services" or "Software as a Service" deliver software available as webservices without any platform dependencies. As example, we will refer to Google again. Their mail application and on-line office package are excellent examples of software as a service.

The following question is easy to foresee: Will there be a next revolution and if so, what will it look like?

Web 3.0

Where web 2.0 introduced the first revolution of "the web", at least from a business perspective, people are already discussing its successor, web 3.0 [4]. What will

[2] http://www.blogger.com
[3] http://www.wordpress.org
[4] And, of course, lots of speculations on web 4.0 already exist.

web 3.0 look like?

Regardless of the many definitions that exist, most definitions agree that the "semantic web" is part of web3.0

2.1.2 The Semantic web

Tim Berners-Lee mentions [15] that there are two types of information available; Human Readable Information and Machine Readable Information.

> "The Human Readable Information is presented by web-pages, it is the Internet as we currently know it. The Machine Readable Information is data that is explicitly prepared for machine reasoning; part of the semantic web".

The semantic web is about linked data on the web, it is about distributed knowledge. It is seen as a future evolution of the web, where the information that is available can be understood by machines and not necessarily humans. The purpose of the semantic web is to enable computers to more easily find information, combining it and act upon it, without the need for human intervention.

Figure 2.1 shows the web-standards that enable the semantic web. It shows a picture, also known as the layered cake, were each layer is built on top of the technologies that are referred in the layer underneath. Each layer is general than the layer underneath it. [14]

The different building blocks that are shown in this image, often referred to as the semantic web layered cake, require some basic explanations. The technologies that are explicitly used in this thesis will be explained in more detail afterwards:

- **URI/IRI**
  Make sure we use an international characterset (IRI is an abbreviation for "Internationalized Resource Identifier") , and provide a way of uniquely identifying resources on the web. In other words: unambiguous names.
- **XML**
  The combination of XML and XML namespaces, as well as XML Schema,

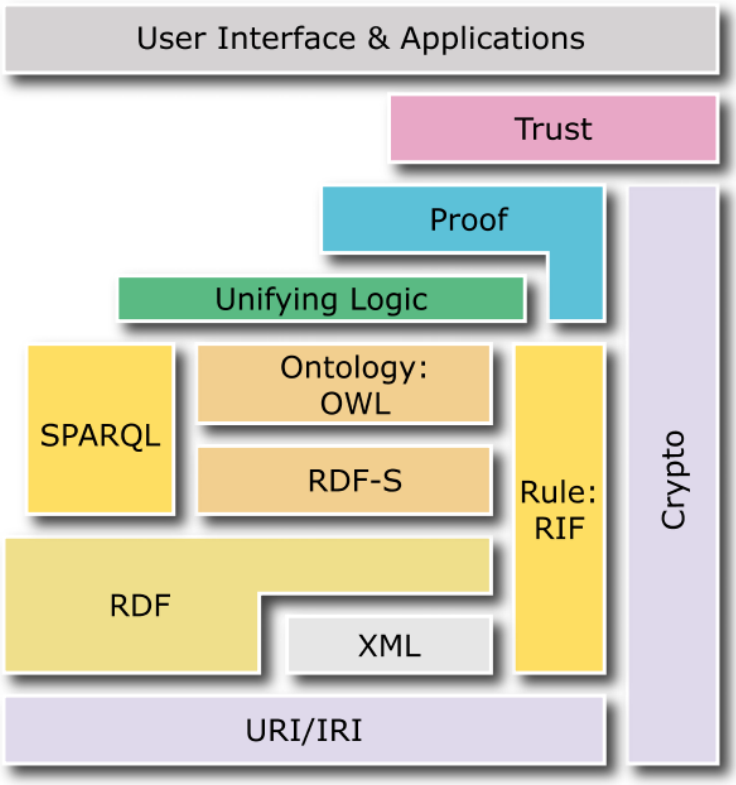

Figure 2.1: Tim Berners Lee - The Semantic Web, layered cake

provide self descriptive documents in a standard way. XML Schema is a layer that restricts the structure of XML documents and additionally extends XML with datatypes. This layer makes sure that we can integrate the definitions with the other XML based standards. This layer is all about syntax.

- **RDF**
  A standard for describing resources on the web, a datamodel for resources and their relationships. RDF is used to describe Metadata, this layer is all about data interchange.

- **RDF Schema (RDFS)**
  A vocabulary for describing properties and classes of RDF resources. With RDF and RDFS it is possible to make statements about objects with URIs and define vocabularies that can be referred to by URIs. This layer is about

taxonomies. [5]

- **SPARQL**
  The language that is used to query ontologies, similar to the way that SQL is used to query relational databases.

- **Ontology: OWL**
  OWL stands for "Web Ontology Language"[6]. This layer supports the evolution of vocabularies, it can define relationships between different concepts. It extends RDF and RDFS, by defining relationships between classes, cardinality etc.

- **Rules: RIF**
  RIF stands for "Rule Interchange Format" . Rules are basically statements in the form of `IF <condition> THEN <conclusion>`, RIF provides a family of languages to encapsulate these statements, so computers can execute them. At this moment. the standard is not yet complete.

- **Logic**
  The logic layer enables the writing of rules, currently under active research.

- **Proof**
  The proof layer executes rules, currently under active research.

- **Trust**
  Evaluate whether to trust the given proof. Currently under active research.

- **Crypto**
  Also called 'the digital signature layer', used for detecting alterations to documents.

The layered cake of the semantic web illustrates the fact that standards are built upon other standards, going from syntax through structure, semantics towards proof and trust. Each layer builds upon the previous layer.

[5] A taxonomy is a hierarchical way of classifying objects, RDFS allows the notion of hierarchy by introducing inheritance

[6] A correct abbreviation of Web Ontology Language would be 'WOL'. Guus Schreiber asks the question: "Why not be inconsistent in at least one aspect of a language which is all about consistency" It might refer to the Owl of Winie the Pooh, who misspells his name as 'WOL', or it might be a reference to an Artificial Intelligence project called 'One World Language' [28].

Ontology

The web was originally built to display information in the form of hyperlinked pages. This information is understandable by humans, but although the enormous amount of information is readable by machines, it is not *machine-understandable*, making it difficult to automate any form of information processing.

By adding a layer of additional information that describes the contained data, we can describe the contents that we are actually dealing with. This layer of additional data, also referred to as "data about data", is called "Metadata".

By adding this extra layer of data, we add knowledge to our system. We describe concepts in a certain domain as well as the relationship between those concepts. It is this combination, when properly expressed, that can be understood by machines, it is exactly what entails an ontology. An ontology consists of types, properties and relationship types representing ideas, entities, events along with their properties and relations according to a system of categories.

In computer science terms, an ontology refers to an explicit specification of a conceptualization, where a conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. [26]

While it is possible to define any type of ontology, there is a general expectation that ontologies resemble a part of real-life information. [25]

So far, it seems that ontologies are a special form of databases, they do not only contain data, but also a description on what exactly is stored. Where databases contain raw data, and have no knowledge on what is stored in its tables, ontologies have this extra layer of data that make the information meaningful.

Ontologies are said to be more scalable, since they are built on top of RDF, and RDF is distributed by nature since they use URIs as their foundation, which makes them different from classical databases. The true power of ontologies however, lies in the methods they borrow from Artificial Intelligence, called "Reasoning" and "Inference".

Ontology Reasoning and Ontology Inference

Reasoning means that we can draw conclusions by the use of reason. Inference means deriving new facts from existing ones [7].

Ontologies contain facts. Facts consists of types, properties and relationship types. Since ontologies are based on facts, we can apply inference to them, although the complexity of inference mechanisms may differ, related to the expressive power of the language that is used [44].

Web resources

The URI, or "Universal Resource Identifier" is one of the most fundamental specifications of Web architecture [15].

The specification basically indicates that anything on the web should be globally and uniquely identifiable, by a string of characters, the URI.

The URI is based on the idea by Douglas Englebart [24] called "Every Object Addressable", stating that every object should have an unambiguous address, capable of being portrayed in a manner as to be human readable and interpretable.

URIs are used to uniquely identify names or resources on the web, using either URNs, "Universal Resource Names" or URLs, "Universal Resource Locators". URNs uses names to uniquely identify resources, a good example is using ISBN to indentify a specific book, URLs uses locations to uniquely identify resources, we can think of a street address, but of course, URLs are typically used to address web-resources.

Boht URLs and URNs are specialized cases of a URI, but sometimes it is difficult to categorize a specific schema as either a URL or a URN, since we *can* use all URIs as names, a URN is a URI that identifies a resource by its name, or we can use URNs to talk about resources by their names, without specifying their location.

IRIs, "Internationalized Resource Identifiers" are a generalization of URIs, since URIs are based on ASCII, a limited characterset allowing only the English alpha-

[7] A famous example is the following: "Socrates is a man", "All men are mortal" from which we can derive "Socrates is mortal". This form of reasoning is called "deductive reasoning"

bet, IRIs are based on the Universal Character Set, Unicode, allowing many more characters, including Arabic, Chinese etc.

XML

XML, or "eXtensible Markup Language" is a markup language that is used for encoding documents, in a structured way. It is most often used to present text and data in a way that is can easily be processed, without the need of human or machine intelligence.

A key-point in XML is that it separates form and content. HTML, the language behind webpages, for instance, consists mostly of tags defining the layout of text, in XML tags define the structure and the content of data. A second important point is that XML is extensible, meaning that it is not a fixed format, like HTML for example is.

DTDs or "Document Type Definitions" is a set of markup declarations. They define exactly which kind of element may appear where in a document and what the elements' content and attributes are. DTDs are superseded by XML Schema, which also is used to put constraints on XML documents, using a set of rules, with a big difference that XML Schema is written using XML, DTDs are not.

RDF, RDFS

RDF stands for Resource Description Framework and is a W3C (World Wide Web Consortium) approved specification. It is part of the semantic web layered cake shown in Figure 2.1

RDF is a datamodel used to make statements about resources in the form of so-called 'triples', a combination of a subject which identifies what object the triple is describing, a predicate, also known as property, which defines the data we are going to give a value to, and an object, the actual value we will give. It is a decomposition of knowledge into smaller pieces. In fact, the goal of RDF is to be as simple as possible, so we can express any fact. On the other hand, it must be very structured, so it is easy for computers to process.

The example below contains an informal triple, where the subject is a person, the predicate points to this persons first name and the object, the actual value, is a

literal; it contains the text “Geneviève”.

```
Person, has firstname, ''Geneviève''
```

Note that each RDF statement should be complete and unique, therefore the subject must use a unique[8] URI. [9] , we call this a named resource. Since URIs can become long and unreadable, we often abbreviate them using the concepts of XML namespaces.[10]

RDF itself is an abstract datamodel, it comes in two formats, called ‘serializations’; XML (eXtensible Markup Language) and N3 (Notation 3) for a non-XML representation.

The informal example above could be written in RDF/XML format as follows:

```
<?xml version="1.0"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:nauta="http://www.nauta.be/rdf/index.html#"
    xmlns:example="http://www.example.org/"
>

<rdf:Description rdf:about="nauta:GC">
    <example:has_firstname>Genevieve</example:has_firstname>
    ...
</rdf:Description>
...
</rdf:RDF>
```

In a few words; RDF is used to model information that is implemented in web resources. It is a datamodel, making the semantic web that uses RDF, a decentralized platform for distributed knowledge. This in contrast to the current web which is a decentralized platform for distributed information visualization.

RDF Schema, or RDFS for short, adds additional semantics to RDF. These semantics are a type system of classes, including inheritance and properties

[8] The subject can also be a anonymous node or blank node (also known as bnode)

[9] This is a HTTP URI. HTTP stands for “HyperText Transfer Protocol”, one of the protocols that is used to transfer data over the Internet. Resources that can be accessed via the HTTP protocol are identified by a set of characters, also known as a “Universal Resource Identifier” (URI). URLs (Universal Resource Locators), often called a web-addresses, are a subset of URIs

[10] Namespaces have no significance in RDF, they are a tool to abbreviate long URIs.

- Class (type of resource), property
- Subclasses and subproperties
- Domain and range
- Comments and labels

Directed edge labeled graphs

The triples of RDF actually construct graphs[11] . Figure 2.2 shows a partial RDF graph of a family.

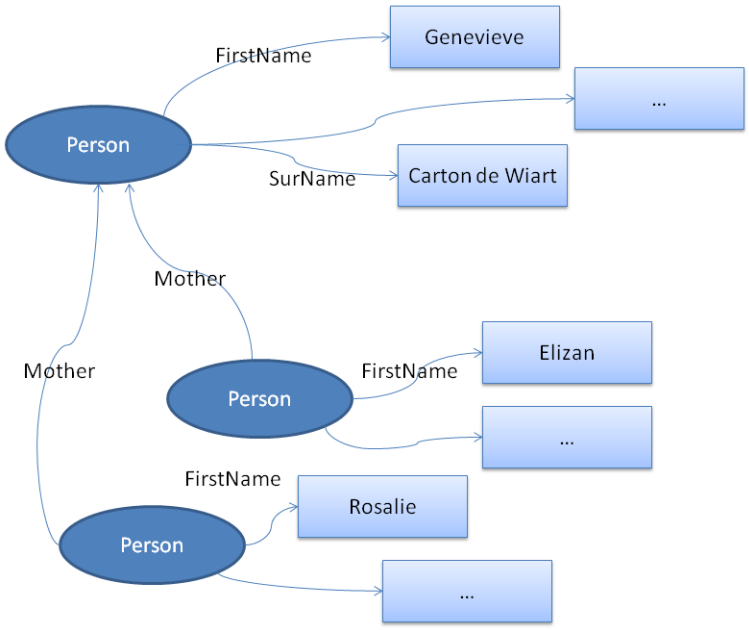

Figure 2.2: A RDF graph containing information on Persons

In general, the nodes in RDF graphs are “things”, arcs are relationship between “things”.

[11] A graph refers to a collections of nodes, or vertices, that may be connected. The connections between nodes are called edges. Edges may be directed, from one vertex to another, or undirected. We can also give a weight to edges, for instance the distance or a payload for travelling over the edge. Search-algorithms for the different type of graphs are different.

The notion of graphs is used, because in graph isomorphism theory [20], a lot of research has been done to the problem whether two (sub-)graphs are the same and how they can be merged. The research often deals with unlabeled, undirected graphs.

In RDF graphs, this problem becomes relatively easy, since most vertices are labeled with the URI of the resource and most edges have distinct labels from the URI of the property of the triple.

RDF compared to a relational database

If RDF is actually nothing more than a simple datamodel, why not use a relational database? If RDF is about information on web-resources, aren't we building a 'web-database'?

When we look at relational databases, we see that they contain tables. Each table consists of rows (the "things" we are storing information about) and columns with represent the attributes/properties of those "things". The combination of a row and a column gives us the value of a specific attribute of a "thing" stored in the database.

If we look at database containing person information, we can imagine the situation as described in Image 2.3.

ID	Firstname	Surname	Birthday	...
...	...	...	...	...
1	Genevieve	Carton de Wiart	...	...
2	Elizan	Nauta	07/07/03	...
3	Rosalie	Nauta	27/07/06	...
...	...	...	...	...

ID	Mother	Father
2	1	...
3	1	...
...	...	...

Figure 2.3: A table containing relational information on Persons

The rows in the first table represent a Person, the columns represent the attributes we know, and are interested in, of this Person. In the case of the image, the highlighted person has an attribute "Firstname" with value "Geneviève".

We can easily see that it is not hard to translate this information into an RDF graph. In fact, Figure 2.3 maps to Image 2.2. The nodes in RDF graphs are "things", arcs are relationship between "things". Foreign keys simply become a relationship to another entity. This is demonstrated by looking at the second table.

The power of using RDF graphs is that the "things" are identified by URIs, making them web-enabled and additionally, we can *uniquely* identify them. Nodes with the same URL are considered identical and this can be used to merge graphs.

Merging of RDF graphs has no limitations, any RDF graph can be merged with any other RDF graph. Formally, it is said that RDF is *monotonic*, merging graphs means merging triples of identical nodes. Adding triples never changes the meaning of a graph, which basically means that you cannot invalidate earlier conclusions nor can you unsay statements by adding triples.

To answer the question if we are building a web-database, the reply is "more-or-less". RDF uses URIs as unique identifiers, we use graphs so we can easily merge information. The information is available on the web, which means that it is not centralized. Therefore we can easily modify information and we can modify in parallel, which means that this is a very scalable solution. We *could* build a web-database. The problem lies in the fact that representing information in RDF is often considered a major overhead [18], therefore it is not often applied, so little information is available since people chose not to make their information available via RDF.

OWL

Web Ontology Language (also known as 'OWL') is a language that is built on top of RDF and RDFS. It includes a standard vocabulary for describing properties and classes, like datatypes, relations, cardinality, characteristics of properties etc.

There are currently two versions of OWL; OWL 1 released in 2004 [9] and OWL 2 which was released at the end of the year 2009 [8], the latter being backwards compatible, but is still a working draft.

OWL 1

OWL comes in three different sublanguages:

- **OWL Lite**
  OWL Lite supports basic classification hierarchies and constraints.
- **OWL DL**
  OWL DL extends OWL Lite, DL stands for description Logic, OWL DL supports all language constructs, but there are some limitations on transitive properties.
- **OWL Full**
  OWL Full extends OWL Lite and OWL DL

Each variant has an increased expressiveness, starting by OWL Lite, through OWL DL and finishing by OWL Full. The language become more expressive, by syntactic extensions of its predecessor. This means that each legal OWL Lite ontology is a legal OWL DL ontology, and each legal OWL DL ontology is a legal OWL Full ontology. [9]

The different profiles of OWL 1 are listed in Figure 2.4

Figure 2.4: OWL 1 Profiles - the onion

OWL 2

OWL 2 is a revision and an extension of the first version of OWL. The following major features are added to OWL 2:

- Improved syntax, making relations easier to express
- More forms of expressiveness

- More datatypes and ranges
- Metamodeling
- Annotations

OWL 2 supports, just like OWL 1, several profiles. There are three profiles in the new specification:

- **OWL 2 EL**
  OWL 2 EL is targeted to applications employing large ontologies that use a simple format. It keeps expressiveness power and consistency, while providing polynomial reasoning time.
- **OWL 2 QL**
  OWL 2 QL is based on OWL DL (and therefore also OWL Lite) and provides an intersection of RDFS and OWL 2.0 DL. It is targeted to data querying and storage. OWL QL also provides ways to express conceptual models.
- **OWL 2 RL**
  OWL 2 RL uses a syntactic subset of OWL 2 and part of its RDF based axiom semantics. This profile is targeted for applications that favor scalability and performance over expressive power.

The different profiles of OWL 2 are listed in Figure 2.5

SPARQL

SPARQL stands for "SPARQL Protocol and RDF Query Language" and is an RDF query language.

Like said before, RDF can be seen as the "web-database", provided that all information evolves to known ontologies. SPARQL can be seen as the web's query language. Every identifier in SPARQL is a URI, which is a global unique identifier. Using SPARQL, we can query in an unambiguous way, unlike in databases where we typically use a combination of "firstname" and "surname" to indicate a person, which is clearly not a unique combination. To avoid this, IDs are often applied as key for the record, making the ID in combination with its table unique in the database only.

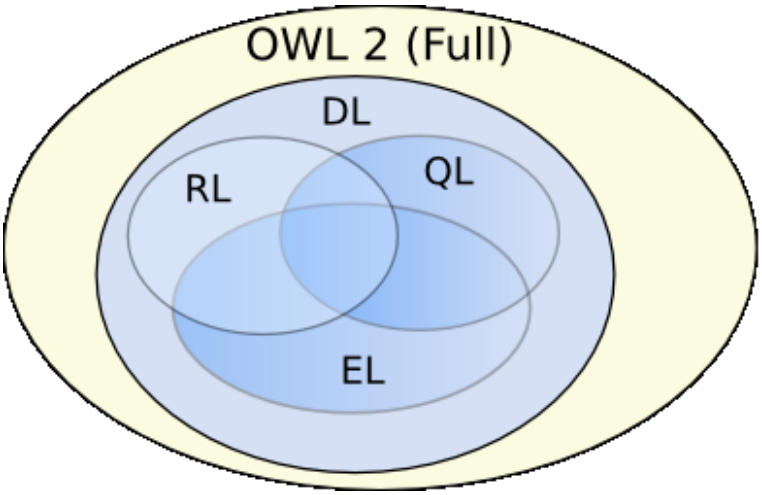

Figure 2.5: OWL 2 Profiles

When using the same example as earlier; we could imagine the following code query for a persons firstname.

```
PREFIX nauta: <http://www.nauta.be/rdf/index.html#>
PREFIX example: <http://www.example.org/>

SELECT ?firstName
WHERE
{
    nauta:GC example:has_firstName ?firstName
}
```

2.1.3 Web3d

We have discussed the future of the web, which is often called web 3.0. We have stated that most definitions of web 3.0 include the semantic web, but some definitions also include web3d in their definition.

Web3d refers to interactive 3d content in web-pages. At this moment, they typically require plugins to display the content, although work is on the way to replace 3d content by Javascript, which is currently the case for O3D, or embed it natively in the web-page. In fact, this is one of the main goals of HTML5, removing the need for plugins.

Although web3d refers to any type of 3d content, we will limit this in this paper

to virtual worlds; 3d worlds that resemble our world, displayable in a browser.

Virtual Worlds are not limited to a world as we currently imagine it. We can extend the notion of world to any sort of containing environment, grouping related objects, but we will use the notion of a paraverse throughout this chapter. A Paraverse is a virtual representation of the world, or a part of it, as we know it. In other words, in this document, we will use the term Virtual World and paraverse for the same thing; to describe a simulated environment that resembles a real-world environment.

Susan Kish [33] identifies three major emerging types of universes:

- Massive Multiplayer Online Role Playing Games (MMORPGs)
- Massive Multi-learner Online Learning Environments (MMOLEs). Virtual environments that are focused on learning platforms, virtual training world, eLearning etc.
- Metaverses are typically virtual worlds that are both social and game oriented. Second Life, a Virtual World allowing users to explore the world, interact, participate in activities and socialize, is the classic example of a Metaverse.

Additionally, she mentions two types that are related to the aforementioned:

- Intraverses, a Virtual World on a corporate Intranet.
- Paraverse, a Virtual World that tries to resemble a (part of) real-life environment. Paraverses are also called mirror worlds. Google Earth is the classical example in this case.

Virtual Worlds (VWs), from a more technical perspective, are computer based environments, typically containing users and virtual objects. Some VWs allow users to interact with other users, some VWs allow users to manipulate objects, some VWs allow a combination of those.

Virtual Worlds are not only something we see, we can also participate in it and often people do that in form of Avatars, a digital representation of their presence.

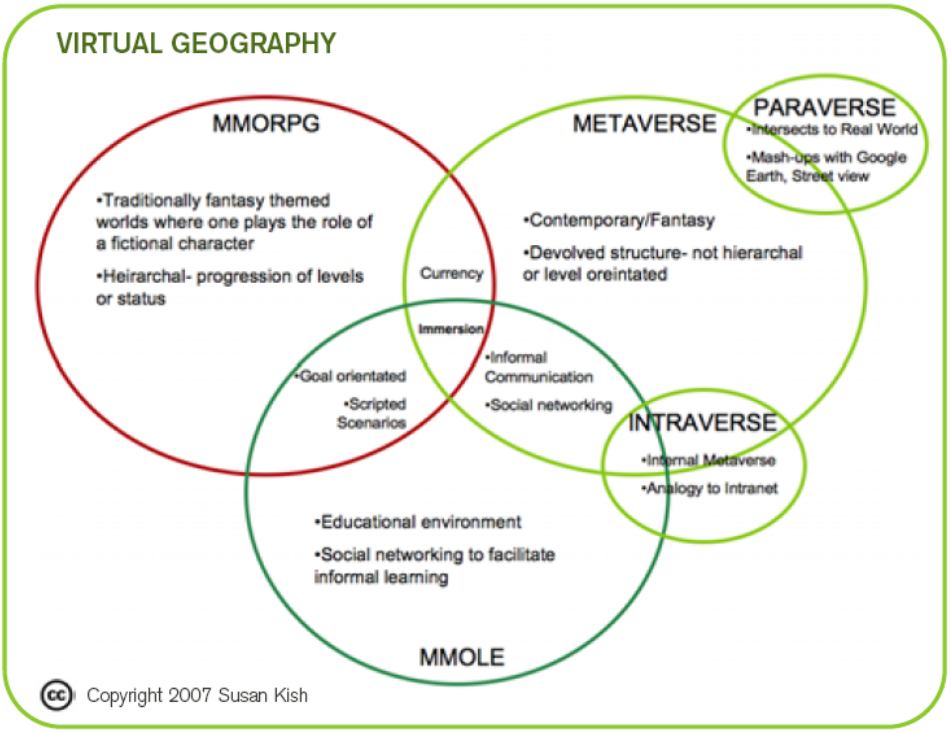

Figure 2.6: Susan Kish - Three separate kinds of Virtual Worlds

Avatars

The word 'Avatar' comes from Sanscrite. It signifies the incarnation or reappearance of a god in the living world.

When people are visiting a Virtual World, they are often represented by an avatar, digital representation, often in the form of a one-dimensional username, a 2-dimensional image or 3-dimensional model.

Head-Up Displays

When using a Virtual World, there is a lot of information available. Often, it is not desirable to display all information in 3d form and for this, Head-Up Displays are often used.

Head-Up Displays are used to provide additional means of showing information. HUDs provide a 2d component with data on a semi-transparent canvas so it doesn't obstruct the users view. Initially developed for military purposes, HUDs

are now also available in commercial aircrafts, cars etc.

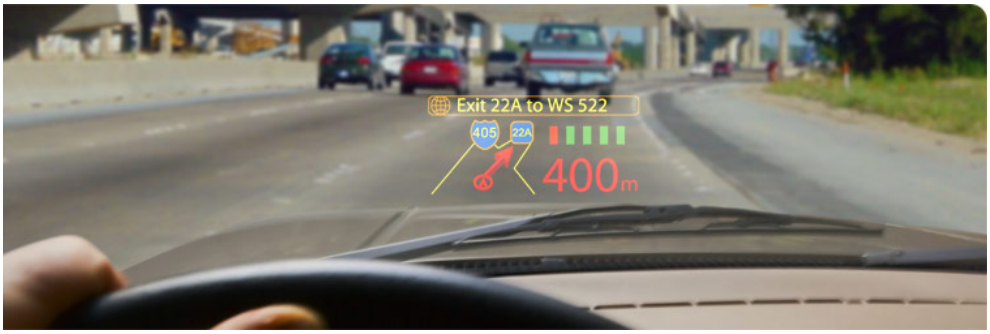

Figure 2.7: Microvision Heads-Up Display, HUDs in vehicles

HUDs are also often available in Virtual Worlds, providing additional information, without cluttering the 3d structures.

Scene

Anything you would like to represent in 3D is a scene. A scene contains the structure (wireframes, mesh), textures, cameras (viewpoints in the virtual world) etc. that are needed to display the 3d content.

Most of the 3d engines share a data structure known as a *scene graph*, containing all scene information. A scene graph is a hierarchical (tree) structure in which a specific node can have many children, but a child can only have one parent. [12]

Nodes in a scene graph are typically one of the following:

- **Group nodes**
  Group nodes are nodes that can contain children. These nodes are typically not visual. Effects that are applied to group nodes are also applied to all its children.
- **Leaf nodes**
  Leaf nodes are nodes that can actually be seen (or for audio; heard)

[12] Some scene graphs are implemented as directed acyclic graphs in which children can have multiple parents.

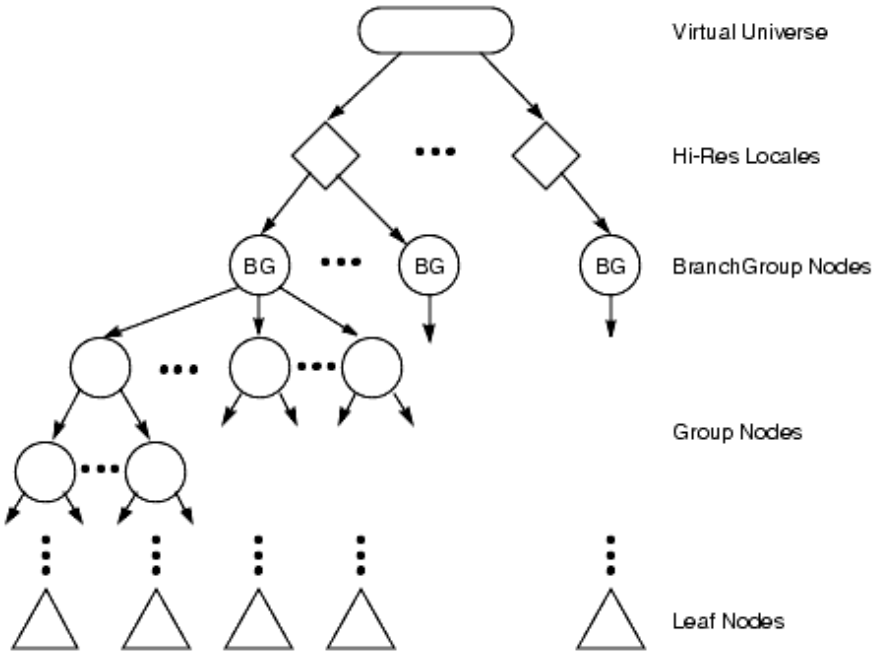

Figure 2.8: A Java 3D Scene Graph is a DAG (Directed Acylclic Graph)

VRML

VRML stands for "Virtual Reality Modeling Language" and is a standard file format that is used for describing 3D interactive graphics.

VRML is a textbased format that describes edges, polygons, textures, transparency parameters, etc [10]. It also specifies the ability to add sound, animations and more. Special nodes like the Timer node can be used to interact with the scene, likewise, script nodes allows a piece of program code, typically ECMAScript, to be added to the VRML file, which can be executed when certain events are triggered. These events can be timer events, but can also come from user-interaction.

VRML was designed to be used in web-pages, it has been superseded by X3D.

X3D

X3D is an acronym of "eXtensible 3D" and is an ISO standard specifying the file format for 3D content. It is the successor of VRML, adding extensions to VRML as well as the ability to encode scenes using XML syntax.

X3D defines several profiles, each adding some more capabilities to the less rich profiles:

1. **X3D Interchange**
   Providing basic features like grouping of objects.
2. **X3D Interactive** Extending X3D Interchange and adding features that enable interactivity like touchsensors, inline nodes etc.
3. **X3D Immersive**
   Adds scripts, audio and more.
4. **X3D Full**
   Adds Geospatial, NURBS (Non-uniform rational basis spline, a mathematical model for generating curves and surfaces), H-ANIM (Humanoid Animation)

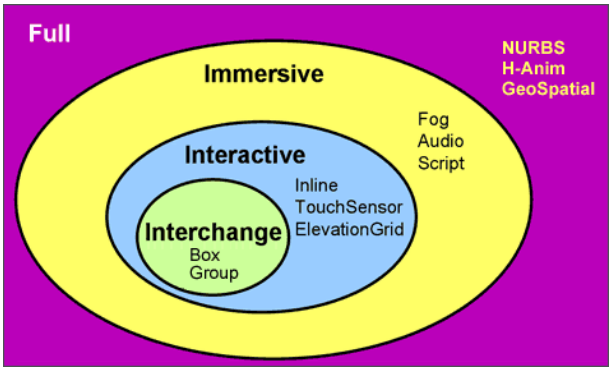

Figure 2.9: X3D Profiles

The traditional way of viewing X3D in a browser is by installing a x3d plugin, like Vivaty[13], Octaga or BS Contact. Most of these plugins use a standard called SAI

[13] On the 31st of March 2010, Vivaty announced that it was shutting down, their products are currently no longer available.

which is a Javascript API used to communicate with the plugin. It allows external parties to call predefined functions on the X3D plugin. BS Contact does not offer the SAI, but has a proprietary solution instead.

X3DOM

X3DOM is an open source framework and runtime to integrate X3D natively (meaning; without the need for plugins) in HTML5 by trying to fulfill the HTML 5 declaration for declarative 3d content.[14]

The idea is to include X3D elements in the HTML 5 DOM[15] tree, which would enable us to manipulate the 3d scene by modifying DOM elements, without the need for plugins (which are controlled via the SAI). The idea is illustrated in Figure 2.10 [1].

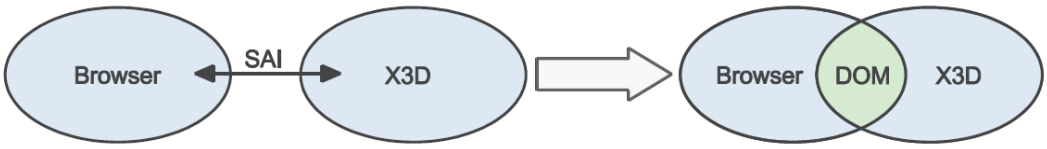

Figure 2.10: Moving from a loose plugin-based Scene-Access-Interface (SAI) to the tightly integrated X3DOM model.

WebGL

WebGL is a standard (a low-level API) to provide 3d content in webbrowsers, without the need for a plugin. WebGL runs in the HTML5 canvas, which means that it has full access to the DOM (Document Object Model) interfaces.

The fact that WebGL it is based on OpenGL has the advantage that OpenGL is cross-platform[16] and cross-browser. All major browser vendors already implement it in their current beta versions of the browser. The disadvantage on the

[14] The draft on HTML 5 at the following location: http://www.w3.org/TR/html5/no.html#declarative-3d-scenes specifies the following: **13.2 Declarative 3D scenes** Embedding 3D imagery into XHTML documents is the domain of X3D, or technologies based on X3D that are namespace-aware.

[15] DOM is an abbreviation for "Document Object Model", it is a hierarchical structure for representing and interacting with objects in a HTML page. Interaction with objects in the DOM is typically done via client-side Javascript.

[16] cross-platform should not be confused with platform independent. WebGL is cross-platform,

other hand is that it is tightly coupled to hardware-accelerated graphics which makes portability to other infrastructural platforms (like mobile devices) difficult or sometimes even impossible.

O3D

O3D is being developed by Google, and is one of the more popular 3DWeb formats. It is an open source Javascript API for creating interactive 3D applications. It started out as a browser plugin, but has been replaced by a javascript library using WebGL. The differences between the two versions are shown in Figure 2.11

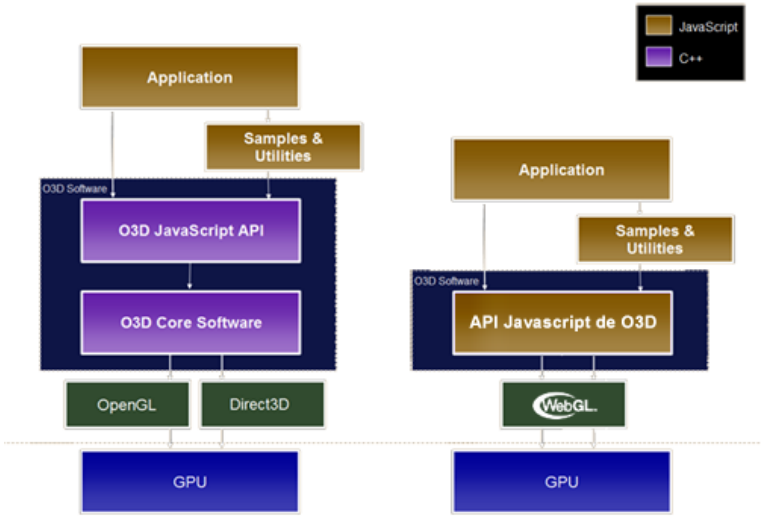

Figure 2.11: O3D Software Stack - plugin and future version

The previous version shows the O3D core software, the plugin that needed to be downloaded and installed in your browser, the new version shows that there is no longer a need for a dedicated plugin, the rendering will be handled by a Javascript API.

indicating that it runs on a variety of platforms, but not necessarily all. X3D, as example, is platform independent which means that it should be possible to implement render engines on *any* platform.

The reason why Google intialially built a plugin was that they did not expect that Javascript could meet the performance expectations. Recent development in Javascript engines shows that Javascript becomes more and more performant. [17]

Collada

Collada is an abbreviation for "COLLAborative Design Activity" and is the intermediate file format for 3d applications. Most 3D render applications (like Blender, 3dStudio), but also tools like Adobe Illustrator, Google Sketchup have facilities to export to this format.

Digital Assets in collada files are described in XML format, the files have the extension `.dae` which is an abbreviation for "Digital Asset Exchange", emphasizing the fact that collada provides an intermediate format. [18]

2.1.4 HTML

HyperText Markup Language, also known as HTML is the markup language that is used on the Internet. The language defines ways to structure documents, by using tags to indicate page elements. Browsers are applications that interpret this text and display it according to a well-defined standard.

HTML defines a way of describing layout of information, by using well-defined elements suchs as headings, lists, paragraphs and other items. In short; HTML describes the layout of webpages.

The current version of HTML is HTML 4.01

CSS, or "Cascading StyleSheets" is a standard that defines the appearance of the HTML elements in web-pages.

[17] Not all JavaScript engines are created equal. Google Chrome has V8, Firefox is working on TraceMonkey, and Apple has SquirrelFish. Conspicuously absent is Microsoft, which has opted to "invest more broadly in realistic scenarios" when developing Internet Explorerwith the result that one of the most popular browsers has, by most accounts, one of the slowest JavaScript engines. The good news is that O3D includes Google's V8 JavaScript engine, which ensures consistent performance across all browsers. [21]

[18] Google Earth standard export format is actually a `.kmz` file, which is a zip archive containing `.dae` files

The current version of CSS is CSS Level 2.

XHTML

XHTML is a XML based version that extends versions of HTML. Up to HTML 4, HTML was defined as an application of "Standard Generalized Markup Language" (SGML) , a very flexible markup language framework, XHTML is defined as an application of XML.

The difference between XHTML and HTML is that XHTML must be a well-formed XML document.

The current version of XHTML is XHTML version 1.1

HTML 5

HTML 5 is the proposed successor of HTML 4.01 [19] and has reached "Draft status" in March 2010 (which is 8 months behind schedule). The specification should become a W3C standard by the end of the year 2010.

One of the main goals of HTML 5 is to remove the need for plugins, like Adobe Flash, Microsoft Silverlight or Java FX. Most of those platforms require some sort of Virtual Machine (available as plugin) to run. The power of the use of these frameworks is related to the percentage of install base of the plugins (Javabased frameworks are an exception to this, since Javascript is provided natively is most of the browsers).

StatOwl[20] provides the following plugin market share, average percentage in the year 2009.

The browser penetration of Flash and Java is relatively stable, the penetration of Silverlight is strongly increasing, starting at 21.31% in January 2009, ending at 37.38% in November 2009.

Work is on the way to provide a native (i.e. no plugins) integration of X3D in

[19] Actually, HTML 5 is the proposed successor for the combination of HTML 4.01, XHTML 1.0 and DOM Level 2 HTML

[20] http://statowl.com/plugin_overview.php

Plugin	Avg 2009	Jan 2009	Nov 2009
Flash	96.52%	97.40%	96.71%
Java	80.74%	81.51%	81.68%
Silverlight	29.58%	21.31%	37.38%

Table 2.1: Browser plugin statistics

HTML. [2] [12]

2.2 Scientific and industrial works

This section describes related work, either scientific or from an industrial point of view.

SecondLife [3] is probably the most famous Virtual World that exists. It allows users, via avatars, to interact, socialize, participate in activities and more. Second Life also offers trade options and virtual property.

ExitReality[21] creates an instant virtual place from every web page that is available on the internet. ExitReality is based on the internet itself and not a closed environment like Second Life. Once the plugin is installed, the application allows you to interact and socialize with other users through their avatars and chat boxes. The application turns webpages into virtual rooms, adapting the room to the users interest. This is done for social network sites like Facebook or MySpace but they aim to transform every webpage into a 3-dimensional website. Navigation in these virtual rooms is not easy.

In Detroyer et al. [51], they have developed an approach to make VE adaptable to learners. It is based on the concept of virtual reality adaptation state where the virtual world adapts according to the learner. However, it does not used any kind of way to search and navigate easily in the 3-dimensional space. It is primary done in the context of adaptable E-learning applications.

H. Mansouri [35] used an ontology, in combination with the VR-WISE approach, to create a search engine. Their work is based on the assumption that, while the virtual world is constructed, semantic data can be added via annotations. This se-

[21] http://www.exitreality.com

mantic data can afterwards be used for a search engine and navigation in the VW. This approach is limited to static VWs meaning that the VW cannot be adapted.

Van Ballegooij & Eliens [52] state that in a web-based 3d virtual environment, users often encounter the problems of being 'lost-in-cyberspace'. Besides the notion of disorientation, users have the problem that it is hard to discover all that a VW has to offer, without spending a lot of time exploring it. They propose navigation by query to overcome these burdens. Navigation by query augments the possibility for users to navigate by allowing the user to query for content.

Peng et al. [42] discuss improvements in VR performance in 3D building navigation. They tackle performance problems of large scale VWs by dynamically loading models based on cell segmentation. They investigate route optimizations based on path planning to ease navigation.

K.H. Sharkawi et al. [47] discuss the combination of Geographical Information Systems (GIS) with 3d game engines. They explore virtual navigation, using real spatial information (colors and shapes), landmarks and other features in 3D geoinformation, leading to a significantly enhanced navigation system, compared to 2d maps that are typically used in GISs.

M. Haringer and S. Beckhaus [27] explore extensions to Virtual Reality systems to manipulate objects of a scene, primarily by applying effects at runtime. They implemented a system which provides a user-interface allowing moderators, authors, or automated systems to modify the scene online using the available effects.

J. Ibáñez [29] provides a querying model that allows users to find objects and scenes in virtual environments based on their size as well as their associated meta-information. This model is based on fuzzy logic and is even able to solve queries expressing vagueness. They did not look at the possibilities of applying queries in a VW that changes.

Denise Peters and Kai-Florian Richter [43] discuss the problems people have when trying to orient in large-scale environments. They propose to apply concepts and methods of schematization to focus on the relevant information in representing virtual cities. They do so by investigating processes which play a role in forming mental representations of city environments and use a cognitive agent for evaluating different schematization principles applied to a virtual city by simulating wayfinding tasks.

Xiaolong Zhang [56] takes a look at navigation in a virtual world using a "multi-

scale progressive model". He examines the use of scaling in virtual environment to improve the integration of spatial knowledge and spatial action.

Jia et al. [31] implement a dedicated search algorithm that can be used for navigation in large scale 3-dimensional environments. They use a browser based approach, implementing the algorithm in Javascript, so it can be embedded in the scene itself. The search algorithm is executed in the clients browser.

Yuhana et al. [55] use inference on buildings to determine relative locations of buildings with regards to others. By using distance-ranges, they determine connectivity, build a graph and use a search-algorithm afterwards to calculate shortest paths. This is a very interesting approach, but it is not useful for natural navigation, since the paths are considered as straight lines, while in real-life this is most often not the case.

F. Kleinermann et al. [34] explore navigational issues in Virtual Environments by using semantic information. They discuss the possibility of annotating Virtual Objects, allowing the creation of navigation paths and virtual tour guides. Their annotations are not limited to text but can are multimedia objects. This approach is not useable for our case, since we would like to use external data, that is typically subject to change and additionally, we might want to change appearance based on the external information. Issues that are difficult to annotate.

Chapter 3

Overview of the approach

We have explained in the introduction that the aim of this thesis is to address three challenges.

The first challenge is to add meaning to the Virtual Environment, more precisely its Virtual World (VW). We want to relate the internal structure of the VW, the virtual 3-dimensional objects, to human-understandable information. The second challenge is to be able to adapt the VW with new information. We would like to be able to display this information, but we would also like to be able to use it, reason on it and use the result to adapt the VW. The third challenge is to improve navigation inside the VW.

This chapter will explain step by step how we have designed an approach that addresses these challenges. We will first introduce the approach and then discuss it.

3.1 Approach

As explained in the introduction, we aim at using semantic web technologies to tackle these three challenges. We will first explain how we can add some human-understandable meaning to a virtual world by the use of ontologies. This will be the first building block of our approach. From there, we will explain how the approach can be extended to allow, on one hand, to visualize new types of information coming from the web inside the virtual world, and on the other hand how we

can reason on such information to adapt the virtual world consequently. Finally we will introduce how our approach can be extended to facilitate the navigation inside adaptable virtual worlds.

3.1.1 Ontology, Virtual World and Semantic Mapping

As explained in the previous chapter; Virtual Worlds display (composite) structures, apply effects and more, making the scene look like a part of the world we know. The scene graph tells the 3d engine *how* to display information, but the world has no idea *what* it is actually displaying.

There are several formats that can be used for 3-dimensional information and although it might be possible to store other information in those files as well, it is not a generic nor elegant way. We need to store this information elsewhere. Having different types of information at different locations implies that we need to have some sort of mapping between the information and its 3-dimensional representation

We will start by using the ontology as a way to relate information from the ontology to the virtual world.

3.1.2 Towards ontology mashups

The approach should allow an author to use the information coming from external information, like webservices or RSS feeds, to be visualized inside the virtual world.

Since RDF is XML based, we should be able to easily combine multiple sources, using stylesheets. This goes towards the concept of mashups that are used nowadays in web 2.0. Instead of having different information sources, we can combine them and use them as one new information source. Although this seems to be obvious, since RDF is an XML based language, numerous sources mention [54] [11] [7] that it is actually very difficult and frustrating to combine RDF and XSLT. [1]

[1] We could use XHTML in combination with XSLT to convert web-pages into RDF, but we cannot use XLST in combination with RDF/XML with complete certainty, since RDF is non-deterministic.

If we are able to push changes in the external information towards the Virtual World, in a near real-time manner, the virtual world can then be seen as a template where dynamic information can be visualized. Similar techniques already exist in ExitReality and SecondLife where information on 3-dimensional panels changes.

Changing display information is one thing, but we want to go one step further by adapting the virtual world. As example, instead of changing the information on panels in a 3-dimensional world, like the sun and clouds that are often seen in weatherforecasts, we could think of adapting the world by changing the sky, adding fog etc.

Another example would be where we have a virtual lecture theatre that changes color, based on the number of students that are following a specific course.

There are many datasources available, and lots of different dataformats like Yahoo Weather vs. METAR for weather information, Yahoo Stock vs. MetaStock for stock information, etc.

Regardless of the format of information, it needs to be translated into a format that we can understand, and the logical choice for the resulting format is RDF.

This implies that for each resource, we potentially need a dedicated translator. These are typically programmed, thus adding new resources means that code needs to be adapted. (unless translators can be reused, of course) and this also applies to changed dataformats. These translators are small stand-alone programs, that take external information, and translate it to RDF.

So, we have one or more external information sources. They need to be translated, so we need to write a dedicated application for this. This information needs to be added to the Virtual World, so we need to adapt the VW to handle this new information, if no means exist yet. In between, we need to map the information with the virtual world, which we call “semantic mapping”.

Gregory Sherman [49] researched the use of XSLT and large XML databases and concludes that, since XSLT has a tendency towards quadratic complexity, it should only be used on small- to medium-sized XML database.

3.1.3 Navigation

The third aim of the thesis is to facilitate navigation inside an adaptable virtual environment.

Before explaining our approach of navigation, we will introduce some terms and background information on navigation in 3d worlds.

Navigational awareness is defined as having complete navigational knowledge of an environment. [46] Glenna A. Satalich defines two distinct types of navigational knowledge:

- Procedural knowledge, or route knowledge. This type of navigational knowledge is ego-referenced and is gained by exploring the area. This type of knowledge is characterized by the fact that the user can go from one point to another, but has no knowledge of alternate routes.
- Survey knowledge is world referenced, typically attained by multiple explorations of the environment. This is similar to having a mental representation of a physical map, also called a 'cognitive map'. The characteristics of this type of knowledge is that distances and landmarks are known and routes can be inferred, even if they have not been travelled before.

Applying navigational knowledge to Virtual Worlds gives us the definition of "wayfinding".
Wayfinding is a dynamic process of using our spatial ability and navigational awareness of an environment to navigate in a Virtual World.

The problem with wayfinding in Virtual Worlds is that, from a human perspective, it is very difficult. [22] [53] [43]. It is very easy to lose orientation, to get 'lost-in-cyberspace' [52] [19] [32].

One of the ways to overcome this feeling is to use a 2-dimensional map next to the 3-dimensional world. This map would typically be placed in a Head-Up Display, the users location is indicated on both maps. This is an improvement, although this often leads to another issue called the "alignment problem" [36][2]. Another approach can be that we do not let the user navigate, but we have let the system guide us. This approach is based on "Navigation by query" [52].

[2] A map is called aligned when the upward direction of the map corresponds to the current direction of gaze in real space.

Our approach allows two types of navigation. The first type of navigation is done by querying and jumping to a place inside the virtual world. The query is based on SPARQL, we can build rich queries so we can use the full power of ontologies to get a result,

The other way of navigation is by following a tourguide where a "path manager" can build a path according to the query. The next chapter will explain this in more details. .

3.1.4 Resulting overview

Putting these three challenges together, we get the following overview:

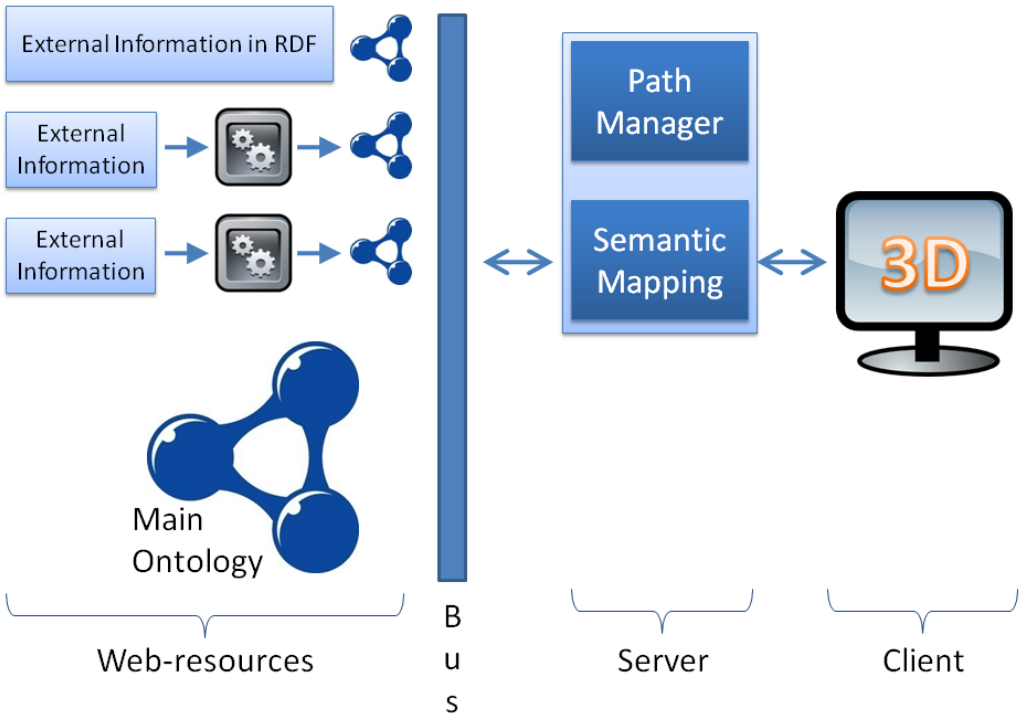

Figure 3.1: Application design - high level overview

Figure 5.1 shows a big ontology icon, with the text 'main ontology' written next to it. This is *our* ontology, the main-ontology containing the principal information

we want to display.

The image also shows external information, with a small execution icon and an ontology icon next to it. With this, we would like to indicate that, usually due to legacy reasons, there are many different information types available on the Web [3], there is no single way of providing information. to be able to use this information, we must translate it to a common form we know, we use a small application that transforms/translates the information to RDF.

The image also shows that we potentially need a translator for each different type of information resource we would like to use. Even a bit stronger, we can say that is unlikely that we do not need a dedicated translator for each type of information.

When questioning what common format we should translate to, the answer is obvious. We, as humans, know what information we are retrieving, we know what it means, so we can translate this into RDF. We supply the data, with an additional layer of data on top of it, the meta-data telling us what information we are actually processing.

RDF Data Bus

Tim Berners-Lee mentions the 'The RDF Data Bus' [16] [13], making 'legacy' information available in RDF format, providing one uniform way of accessing it.

3.2 Why use semantic web technologies?

When displaying information, it seems logical to think in terms of relational databases. They have been forming the backend of many applications for many years. We have chosen a different approach however, we base our approach on ontologies.

Ontologies are meant to facilitate either human-to-human or machine-to-machine communication, but there are many advantages, that can basically be grouped in

[3] A good example of representation differences is how weather information is provided. Dating from 1968, METAR is a raw format, publicly available, which is mainly used by pilots to retrieve weather information around airports. Yahoo weather provides weather information in the form of webservices. The two types of information are clearly incompatible.

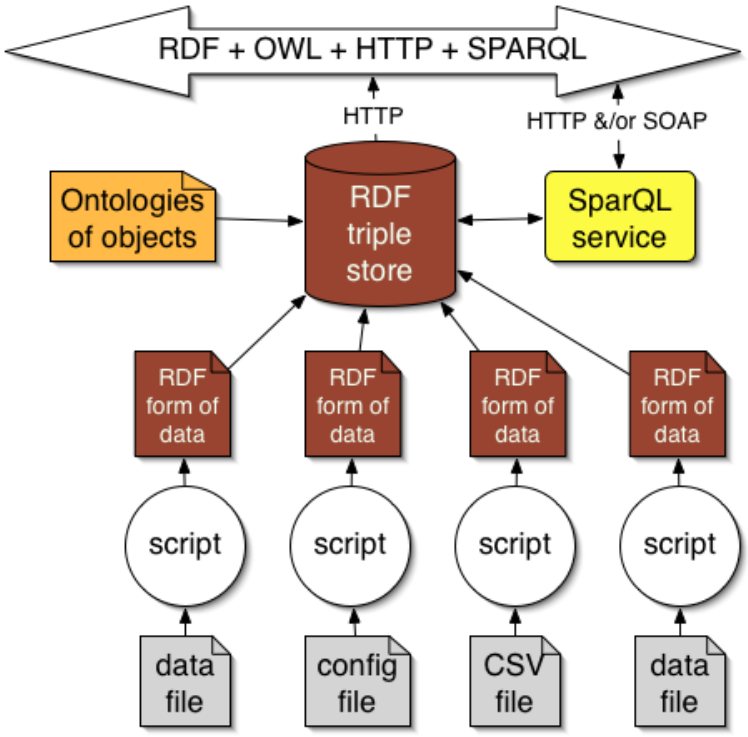

Figure 3.2: Tim Berners-Lee - The RDF Data Bus

three areas [30] of which the most important ones are [41]:

- **Communication** between people. An unambiguous but informal ontology is sufficient
- **Interoperability** among computer systems. The machine-to-machine communication, the ontology is used as an interchange format.
- **System engineering**, in particular:
  - *Reusability*, ontologies can be used and reused as a kind of component between software systems.
  - *Search*, using inference we can use the metadata of the ontology
  - *Reliability*, since ontologies are formal by nature, we can check for consistencies, up to some degree, leading to more reliable systems.
  - *Specification*, ontologies are expressed using the terminology of the domain and thus facilitate the process of identifying requirements.

3.3 Discussion

Recently, Tim O'Reilly reflected on what the web was and where it is going now. He states the following [40]

> "Web 3.0. Is it the semantic web? The sentient web? Is it the social web? The mobile web? Is it some form of virtual reality?
>
> It is all of those, and more."

He argues that the Web has become the world and that everyone participating in this world casts a "digital shadow" containing a wealth of information, which he gave the name "Collective Intelligence".

This thesis discusses a combination of emerging web technologies; the semantic web combined with a 3-dimensional representation. We will create a Virtual World based on information coming from an ontology. We add meaning to the virtual objects we display in the VW. We will adapt the VW with external information, we show that our VW is able to deal with changing external information, and finally we will use the ontology to overcome navigational problems, we let the ontology guide us to our destination.

Chapter 4

PathManager

Like discussed previously, it is difficult to navigate in 3d worlds. A lot of research has been put into improving navigation, avoiding 'getting lost in cyberspace'. Using 2d maps in combination with 3d maps, typically via the use of Heads-Up-Displays (HUDs), are also often applied, although this leads to misalignment problems.

4.1 Semantic mapping extended

We already use the ontology to query for the virtual objects that build up our initial Virtual World, this has been shown in the previous chapter and we use semantic mappings to relate the virtual object to their visual representation. We now extend this mapping by applying point-of-interest (POIs) to each virtual object. For each Virtual Object, we had already retrieved its 3-dimensional representation, we use the same method to retrieve a viewpoint for this object.

If, in real life, we go from one place to another, we follow a certain path. Since paths are seldom straight, we cannot use a starting point and an ending point alone, we need intermediate points that indicate angles, places in real life where the direction changes. POIs with a certain orientation are also known as 'viewpoints'[1].

[1] It is important to understand that there is a differerence between Points-of-interests and viewpoints. A Point-of-interest is simply a location, it has coordinates. A viewpoint also has orientation. A viewpoint in 3-dimensional worlds is also called a 'camera'.

In 3-dimensional worlds, we use viewpoints. Therefor the semantic mapping provides a binding

These POIs will be retrieved from the ontology and we apply the semantic mapping, in the same way as we did for the Vrtual Objects, only this time we map the Virtual Objects to viewpoints.

After having re-applied the semantic mapping, we have an in-memory map of all virtual objects, and we have attached viewpoints to them. We want to connect Virtual Objects so we can derive paths between them. We can connect Virtual Objects by indicating their relative positions, for instance Object A is above Object B. We could also define a less restrictive connection pattern, indicating that virtual object A is connected to virtual object B.

Since the ontology provides us with the virtual objects to display, it must also provide us with the connectivity between the virtual objects. In other words; the ontology must provide us with the 'cognitive map' we will use to navigate.

Our path manager is now aware of the different virtual objects and their corresponding viewpoints and the connectivity between the objects. We have constructed a graph.

4.2 Navigation by query

As we have explained in our previous chapter, our primary means of navigation will not be user-initiated, using mouse-gestures or arrow controls to go forward, turn etc. We will use a query-based navigation. We let the user ask a question to the system, the system will guide the user to the destination, based on the result of the query.

For our 'cognitive map', the representation of the map in the Virtual World, we will use a graph, connecting different landmarks. Intermediate landmarks will indicate places where the direction of the path changes.

The map, represented by a graph, can now be used to guide us from our starting point to our destination. Depending on the type of graph (directed, weighted, etc), different shortest-path search-algorithms, like breadth-first or Dijkstra, can be used. Based on the destination, mapped to a viewpoint in the graph, the pathmanager will return the path to follow when travelling from A to B.

---

to viewpoints and not POIs.

So, instead of letting the user navigate through the world itself, we propose a system in which a user queries some sort of engine, and the result is used to move the navigator from viewpoint to viewpoint, imitating a natural path as if we were walking in the Virtual World.

4.3 Constructing the navigation paths

Once we have a 'cognitive map', we can use a search algorithm to get from a certain starting point to our destination. Since we use viewpoints, each POI has an orientation, we can implement a navigation path that looks natural. The question is how we get the initial viewpoints, that make up our path, to start with.

Kleinermann et al. [34] use semantic annotations by letting users position a POI on an object or around it. They use two different ways for positioning POIs: *grid* positioning, where the designer can position POIs according to predefined positions for the POI, and *freehand* positioning, where the designer can position an object and add a viewpoint anywhere on (or inside) an object. Their definition of navigation paths follow the same approach, linking different landmarks, where a landmark is a POI with an orientation. All their annotations, landmarks etc. are uploaded into a reasoner.

Our approach is a bit different, we used a map on which we placed navigation paths. The intersections and end-points can be used as landmarks.

As example, we will look ahead to the situation we also use in our proof of concept, later on in this document. We take a map of the campus of the "Vrije Universiteit Brussel" (VUB) as indicated in the following image:

We overlay the map with a graph, each end-point and each intersection point becomes a point of interested. We have numbered the points for reference. *This implies that we use the ontology to store connections between Virtual Objects, not their relative locations!*

This graph can be easily stored in the ontology. All we need to say is that a point A is connected to point B. For bidirectional graphs we also encode the link between point B and point A.

The graph is ready, we know the different POIs and the connections between them. How do we relate these POIs to viewpoints, points with an orientation, useable in

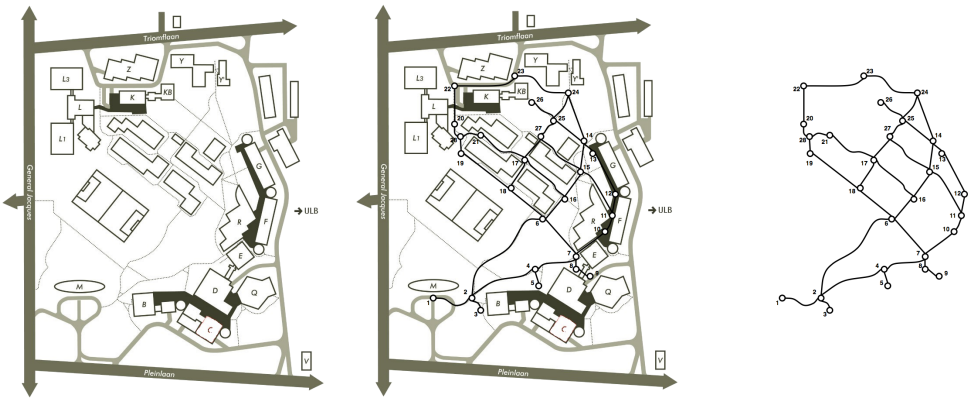

Figure 4.1: The map represented as graph

Virtual Environments?

The POIs that are stored in the ontology are names of points and their connectivity. These POIs have no orientation. For example; when looking at the intersection in the lower-left corner of the image, we see that POI number 2 is connected to POI 1, 3, 4 and 6. This implies that for POI number 2, we have to define 4 viewpoints, since viewpoints have an orientation as well. We defined a viewpoint for the path from viewpoint 2 to viewpoint 1, from viewpoint 2 to viewpoint 3, from viewpoint 2 to viewpoint 4 and from viewpoint 2 to viewpoint 6.

Using Google Sketchup, we navigated to the POIs that we indicated in the previous step. We orientate ourselves in such a way that our current camera/viewpoint is looking to the next viewpoint. This position, including orientation, is saved, so we can convert them to useable viewpoints afterwards, in the format we will use to construct our Virtual Environment (X3D, O3D, ...).

Chapter 5

Approach implementation - a case study

This chapter describes the application built on the ideas that we highlighted in the previous chapters.

As described earlier, our application shows a paraverse, a real-life kind of Virtual World (VW), that is driven by an ontology. With this we mean that the information we show in the VW comes from the ontology and additionally, we use the ontology to help us with navigation.

The application we built uses the campus of the "Vrije Universiteit Brussel" (VUB) as its paraverse. The ontology therefore contains notions of the buildings on the campus and any additional information that can help us for our query-based navigation. For example; we use names, telephone numbers and roomnumbers of employees of the university, building a minimal telephone-book.

5.1 The technical building blocks

When implementing a proof-of-concept (POC), some technological choices need to be made. These choices are related to 3d engines, application environments, related utilities etc. Which one do we use and why?

To build our POC, we will use the following technologies. Most of them have

already been explained in the previous chapters, in this section we will provide a justification why we chose these specific technologies over alternatives.

- **X3D**
  The choice for X3D is made because it is an open standard, supported by W3C. Additionally, it is proposed as part of the new version of HTML (HTML5), which means that, if implemented by browser vendors, no additional plugins will be needed. O3D, Google's API, is open-source, so it would have been a very good alternative. O3D is much tighter coupled to the platform on which is runs (WebGL), where X3D is platform independent. The first beta versions of all major browsers currently support HTML5.

- **Adobe Flex**
  Arbitrary choice, but again the choice is driven by the fact that the platform is open source and it is browser independent. Flex has an object-oriented language embedded, called ActionScript and additionally has a lot of strong widgets, making web-development easy. Silverlight (Microsoft) would have been an alternative platform. It has C# as backing language (in fact, any of the languages running on the Microsoft runtime environment can be used), which is an object-oriented language. It also has a strong widget set. The platform is closed and not free, making this a less obvious choice.

- **jQuery**
  It is a Javascript library and framework which is very strong in handling asynchronous requests with the webserver and DOM manipulation. This is also an arbitrary choice, any framework could have been used.

- **OWL**
  OWL stands for "Web Ontology Language" and is a family of languages used to write ontologies.

- **SPARQL**
  The SPARQL Protocol and RDF Query Language (SPARQL) is a query language and protocol for RDF.

- **Joseki**
  Joseki is an open-source HTTP engine, that supports SPARQL queries. It is based on Jena, a Java framework for building semantic web applications. This is an aribitrary choice, it was the first one we found and it serves our purposes.

Let's map these technologies to the design we have already highlighted in our approach:

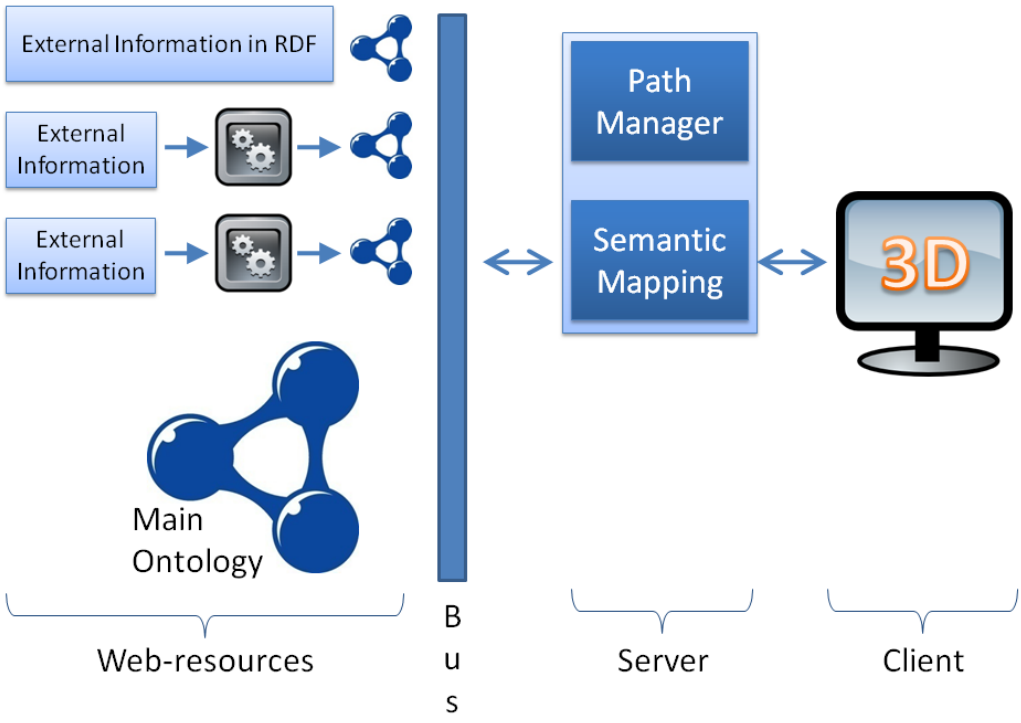

Figure 5.1: Application design - high level overview

When we look at the image, we can split it into three parts; the ontologies (web-resources) that are accessible via http and connected via the bus, our server-side application containing the semantic mapping and the path-manager and the client side, containing the browser for the virtual environment. We will now position the different technologies in these three blocks:

- **Ontology related**
  - Joseki - The HTTP engine that responds to SPARQL queries. We can access this engine from our application (server side) and query it via SPARQL. Joseki can be seen as the implementation of the bus in the image.

  - OWL - The ontologies themselves are written in OWL. OWL is one of the languages that can be used to write ontologies, OWL, on its turn, is based on RDF.

- **Server side application**

  - Adobe Flex - The server side of the application, including the Semantic Mapping and the Path Manager are written using Adobe Flex, a Flash based application framework that uses an Object Oriented language called 'ActionScript'.

- **Client side application**

  - X3D - The language used to build up our Virtual Environments.
  - jQuery - a javascript engine that is very good in handling asynchrounous callbacks.
  - Adobe Flex - The controls of the client side of the application are also written in Adobe Flex. In fact, a Flex application often provides a server- and a client-side part, where the client side is built with widgets that interact with the server side.

5.1.1 The ontology

The ontology we have used for the proof of concept is based on an existing ontology by Patrick Murray-John [37]

It describes a Campus consisting of CampusPlaces, Courses, Organizational units, Persons extending foaf:Persons[1].

We have extended this ontology by adding the following objects:

- Staff, extending the Person class from our base ontology.
  - AcademicStaff, extending Staff
    - Professor, extending AcademicStaff
    - Assistent, extending AcademicStaff

[1] FOAF stands for "Friend of a Friend" and is on ontology describing persons, activities and relationships.

        - Researcher, extending Assistent
    - AdministrativeAndTechnicalStaff, extending Staff
- Building, extending the CampusPlace class from our base ontology
- Floor, extending the CampusPlace class from our base ontology
- Room, extending the CampusPlace class from our base ontology

In our ontology, each Staff-member has a room. Rooms are logically assigned to Floors, floors on their turn to buildings.

It is clear that this is a very limited setup, but the goal was not to design a university ontology. We wanted to have information available which we could relate to campusplaces, which are mapped to points-of-interest in our application.

We use the ontology to query for a person's name, for instance, and as a result we get the building in which this person has a room.

The ontology stores triples, that are basically relationships between two nodes. We use the Web Ontology Language (OWL) to build our ontology.

Web Ontology Language

As a small example; the definition of a Professor Class in OWL:

```
<owl:Class rdf:about="#Professor">
    <rdfs:subClassOf rdf:resource="#AcademicStaff"/>
    <owl:disjointWith rdf:resource="#Researcher"/>
</owl:Class>
```

We define that a professor is part of academic staff, and a professor is not a researcher. When creating an object (an instantiation of a class), we might have something like this:

```
<Professor rdf:about="#Olga.DeTroyer">
    <hasFirstName>Olga</hasFirstName>
    <hasSurName>De Troyer</hasSurName>
    <hasTelephoneNumber>+32-2-2222222</hasTelephoneNumber>
    <hasEmail>Olga.DeTroyer@vub.ac.be</hasEmail>
    <hasFaxNumber>+32-2-1111111</hasFaxNumber>
```

```
        <worksForUnit rdf:resource="#Computer_Science"/>
        <hasRoom rdf:resource="&facilities;10F000"/>
    </Professor>
```

The relationships between nodes in the example above are, for example 'hasRoom' that connects a professor with a room. The room refers to another ontology, that is defined in the header of this specific ontology.

SPARQL

We use OWL as language to define our ontology, we also need a way to retrieve the information from it. SPARQL is a standardized RDF query language, we use it to query our ontology. Based on the example previously given, we also provide a small SPARQL query, an example to query for a roomnumber.

```
PREFIX vub: <http://www.vub.ac.be#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?professor ?roomNumber
FROM <Data/staff.owl>
WHERE
{
    ?professor a vub:Professor .
    ?professor vub:hasSurName "De Troyer" .
    ?professor vub:hasRoom ?room .
    ?room vub:hasNumber ?roomNumber
}
```

Other university ontologies

Some other projects (other than the base of our ontology) have also explored the possibilities to model a campus.

- Benjamin Nowack proposes "Semantic Campus - a FOAF extension" [2] that describes campus-related resources such as universities, departments, lecturers and students. It has no notion of physical, geographical locations.
- Jeff Heflin defined a university ontology [3], but has a strong focus on documents that are published and has no location information.

[2] http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/pp/semantic_campus/

[3] http://www.cs.umd.edu/projects/plus/SHOE/onts/univ1.0.html

While the concept has not been finalized yet, Patrick Murray-John is thinking [38] about a giant edu-graph that combines ideas, subjects, people and the resources used in teaching and learning. This project is interesting, since it combines several ontologies; FOAF (Friend Of A Friend), the Bibliographic Ontology, GeoNames, SIOC (Semantically Interlinked Online Communities) to name a few.

5.1.2 External information

We have a ontology that defines buildings, employees of the university etc. But we also mentioned that we use external information to enrich our virtual world. For our proof of concept, we used METAR information, weather information often used by pilots. This is an old format, dating from 1968, obviously not available in RDF. We wrote a small application that parsed the METAR string and returned it in RDF. The input string and the resulting RDF information are added in the appendix.

5.1.3 The Virtual World - The campus in 3D

Sye Nam Heirbaut has developed a 3D version of the campus using Google Sketchup, a tool that can be used for 3d modeling. His work is used as base for our 3d world. This work was done as a student job, unrelated to any university project. The result of his work is used in this project however, since it provided a complete model of the campus, and we could export the models to VRML. Using VRML and Vivaty Studion we could translate it to X3D.

This section describes the steps involved in creating a Virtual World in X3D based on the Google Sketchup models. We will start by a small explanation on X3D.

eXtensible 3D

We have chosen eXtensible 3D (X3D) for the fact that it is an open standard, supported by W3C and it is designed in such a way that it is platform independant. Scenes are not tied to underlying hardware specifications, screen resolutions etc.

X3D files are read and parsed by X3D browsers. An X3D Browser is responsible for the interpretation, execution and presentation of the X3D Scene Graph. It is

the browser that understand the contents and know how to render its information, resulting in the display of virtual objects. We call the representation of an X3D Scene graph a 'Virtual World'. Browsers can be desktop application or plugins in web-browsers.

Before going into more detail, we will provide a small example of an X3D file, based on an example by Don Brutzman [19]:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC
        "http://www.web3D.org/TaskGroups/x3d/translation/x3d-compact.dtd"
        "/www.web3D.org/TaskGroups/x3d/translation/x3d-compact.dtd">
<X3D>
    <head>
        <meta name='filename' content='geometryExample.x3d'/>
        <meta name='author' content='Don Brutzman'/>
        <meta name='created' content='8 July 2000'/>
        <meta name='revised' content='5 January 2001'/>
        <meta name='url'
            content='http://www.web3D.org/TaskGroups/x3d/translation/
                examples/course/geometryExample.x3d'/>
        <meta name='description' content='User-modifiable example to
            examine the role of the geometry tag. See what nodes
            can be replaced: geometry (no) and Cylinder (yes).'/>
        <meta name='generator'
            content='X3D-Edit, http://www.web3D.org/TaskGroups/x3d/
                translation/README.X3D-Edit.html'/>
    </head>
    <Scene>
        <Shape>
            <Appearance>
                <Material diffuseColor='0 .5 1'/>
            </Appearance>
            <Cylinder/>
        </Shape>
    </Scene>
</X3D>
```

This example demonstrates a simple cylinder. While in this specific case, the metadata may seem heavy, in larger files, this will be less an issue.

Each X3D application [4]:

1. implicitly establishes a world coordinate space for all objects defined, as well as all objects included by the application;
2. explicitly defines and composes a set of 3D and multimedia objects;
3. can specify hyperlinks to other files and applications;
4. can define programmatic or data-driven object behaviours;

5. can connect to external modules or applications via programming and scripting languages;

This leads to the following overview as described in Image 5.2:

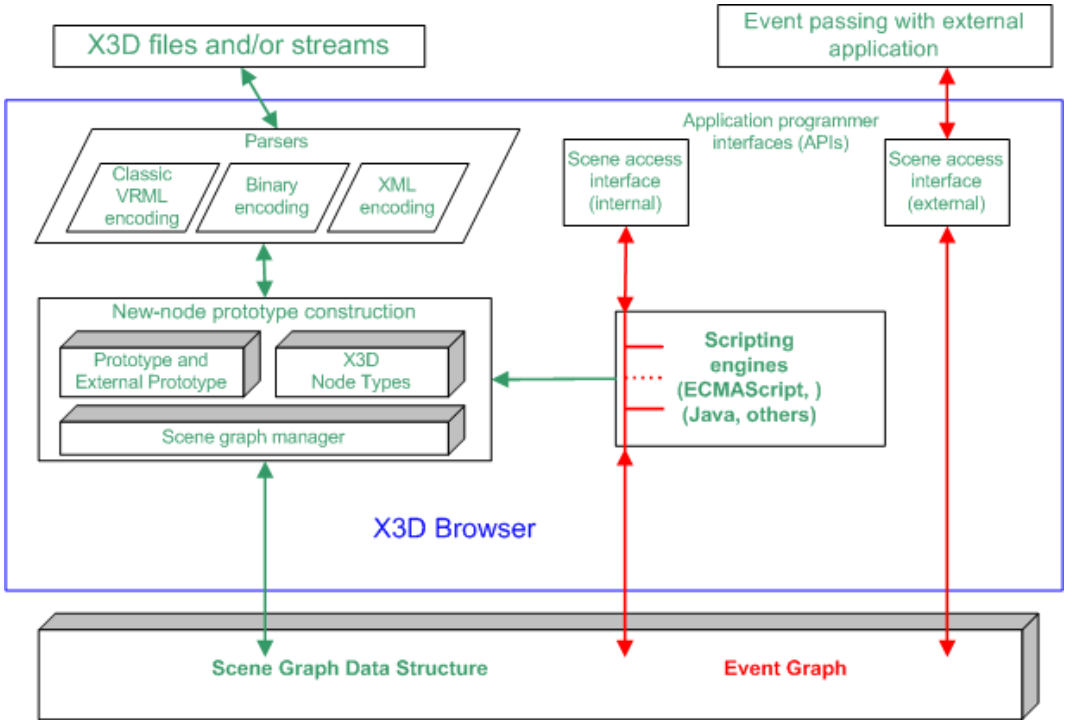

Figure 5.2: X3D System Architecture

The image shows that a browser should be able to render X3D, but also VRML, X3Ds predecessor and binary X3D. The Scene Access Interface (SAI) is used for event passing with external applications.

Scene Access Interface

The Scene Access Interface (SAI) is a application programmer interface (API) that defines runtime access to the Scene. The SAI is used to create nodes, modify nodes, send events to nodes etc.

We use the library created by Ajax3D [5] for our application. This library already defines Javascript methods that use the SAI to create an initial scene and add

virtual objects.

Google Sketchup

We received an initial file of the entire campus in Google Sketchup format. The file, as we received it, was not useable for the web, due to its size and format, so we had to make it web-friendly.

Starting from the initial file, we started by isolating all buildings. With this we mean that for each building in the campus, we removed all unrelated information. In other words, we started from the initial campus and removed all buildings, except the one we wanted to isolate, on a per-building base.

The result is that, instead of one big file containing the entire campus with all it's buildings, we now have created multiple files, one per building. The buildings themselves still have their geo-location which was available in the initial Skecthup file.

The next step was to create a simplified version of the building. Removing as much detail as possible, reducing the building to basic building blocks. This was typically done by taking the roof of the building, removing the entire structure underneath and afterwards extend the roof until the ground.

Afterwards, we applied a texture to the buildings. The texture itself was taken from an image created with the Google Earth plugin called "V-Ray for Sketchup" [4].

The final step in Google Sketchup was to export the buildings to VRML format.[5]

Vivaty Studio

Vivaty Studio, formerly known as Flux Studio, is a tool that allows the user to manipulate 3d scenes. The application allows, amongst others, to import VRML scenes and to export them to either uncompressed X3D or compressed X3D. We actually used the tool for both. We exported to uncompressed X3D to get the viewpoint we previously set in Google Sketchup [6], and we also used it to get a

[4] We also directly applied a default viewpoint to the scene, for later use.
[5] This is a feature that is only available in the Google Sketchup PRO version.
[6] Vivaty somehow loses the viewpoints parameters, these need to be manually set

compressed version of the scene. [7]

Binary compression of the X3D files, greatly enhances downloadspeed. The downside is that the DOM structure of the binary file is no longer accessible which implies that the scene is no longer modifiable from the SAI afterwards.

The solution to this problem is to compress the visual part of the scene after having seperated it from all non-visible nodes (like viewpoints, scripts etc). The non-visual nodes are placed in a the text-version which references the binary file. An example can be found in the appendix.

The reason for simplifying and compressing becomes clear when we look at the following table which highlights the file-size differences between the different possibilities (file-size of resulting .X3D files):

	Sketchup	VRML	Uncompressed X3D	Compressed X3D
Unmodified	13.6 Mb	17.3 Mb	-	-
Simplified	10.5 Mb	119 Kb	227 Kb	10 Kb
With textures	10.6 Mb	131 Kb	242 Kb	20 Kb

Table 5.1: Compression differences between VRML, X3D and X3D compressed

In case of the unmodified version, Vivaty studio is unable to export it as either form of X3D (uncompressed or compressed).

The result of the simplified version and the version with an applied texture can be found in the appendix.

We could improve the display result by using more detailed base structures etc (less abstraction), but this would increase the number of polygons that define the buidling. We could also improve the quality and the number of textures, which are simple images applied on a surface, but this would increase the file-size and thus also increase the render time. We chose a very simple texture to reduce the filesize of the image which means that we load the entire scene faster.

The trade-off is between appearance and speed.

[7] In the compressed version, we removed the viewpoint so we did not have any duplicate viewpoints.

5.1.4 Semantic mapping

We explained in our chapter on the approach, that we use a semantic mapping to relate ontologies to virtual environments. In our approach, we use OWL to define our ontology, we use SPARQL to retrieve information from it, and on the other side we use X3D for 3-dimensional representation of these objects. How does the semantic mapping connect these technologies?

When the application starts up, it will start by getting information on the virtual objects that we need to display. This information comes from our ontology, but this information contains no visualisation information, we receive the buildings we would like to display in our paraverse. To get the display information, we map the objects to their 3-dimensional representation.

The mapping uses an initial query to ask the ontology for the virtual objects to display. We also need a mapping file, that maps each object to an X3D file. When we have the objects we would like to display and we have their 3-dimensional representation in the form of an X3D file, we perform a function call on the Scene Access Interface, so we can add the object to our Virtual World.

It is important to realise that we add multiple objects. For each object we received from our ontology, we perform a mapping.

A mapping might look like this:

```
<SceneMapping>
    <Mapping>
        <URI>http://www.vub.ac.be/facilities/M</URI>
        <SceneURL>http://www.vub.ac.be/x3d/M.x3d</SceneURL>
    </Mapping>
    ...
</SceneMapping>
```

We have mapped our information from our ontology, building M indicated by the URI `http://www.vub.ac.be/facilities/M` to a 3-dimensional representation in X3D: `http://www.vub.ac.be/x3d/M.x3d`. Using the Ajax3D implementation [5] we already have a function available to add the specific X3D information to the existing virtual world.

External information

We also use the semantic mapping to map external information. The procedure is the same, except for the fact that we can no longer use the Javascript call from Ajax3D that was used to add virtual objects. External information might alter the virtual world, instead of adding a virtual object. This implies that for external information, we do not map to an X3D file that represents the information, we map to a Javascript function call instead. It will be the implementation of the Javascript function that handles the external information.

A small example will make things clear: Imagine that we would like to add weather information to our virtual world (this is actually what we do in our proof of concept). We do not add any objects when weather changes, instead we change the sky according to the weather conditions. We might also want to activate some fog in our virtual world. In this case, we will have a javascript function that receives weather information and modifies the scene accordingly.

5.1.5 PathManager

The PathManager is the component that guides the user to its destination by providing the path from its current location to the destination, using a search algorithm over a graph.

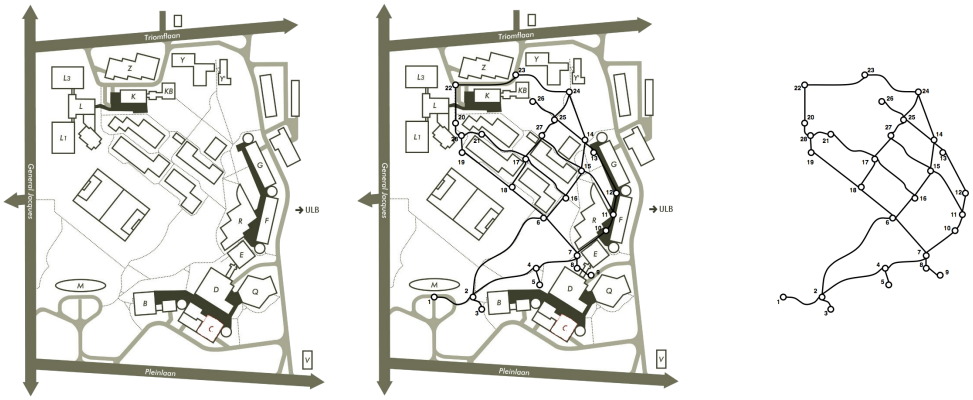

Figure 5.3: The map represented as graph

We started by creating a graph that is an abstraction of the walking paths a person

can take on the campus. The graph is created by taking a map of the campus and drawing routes on top of it. The resulting graph is shown in Image 5.3[8]

For the sake of simplicity, we use an unweighted bidirectional graph, so we can easily apply a Breadth-First-Search (BFS) algorithm to dynamically find a shortest path between two nodes[9]

We retrieve the connections between the different points from the ontology. They are mapped to viewpoints, points with a certain location. Since these viewpoints are non-visible elements of a SceneGraph, we can use the same semantic mapping as we used to add virtual objects to the scene. We also use the same Javascript function call, using the Ajax3D API.

At this point, we have a fully initialized application. The initial virtual objects are shown, it is enriched using external information, we have a representation of the Virtual World in the form of an in-memory graph which is handled by the PathManager and we have the query available to respond to any question coming from the user.

The process of guiding the user are the following:

1. The user submits a query. The application translates this to a SPARQL query (static mapping)

2. Whether we ask information about persons, rooms, courses, the result will always be a building. This result is passed on to the PathManager. The PathManager knows the current location and knows the destination, which is related to the result of the query. It will return the starting viewpoint, a list of intermediate viewpoints and the destination viewpoint based on a graph search algorithm.

3. For each step in the list of viewpoints, the renderengine receives a function-call to change the current viewpoint. We have added a small delay between each function invocation to give the navigation a more natural look and feel.

[8]The actual implementation only implements a part of this graph, due to the fact that Vivaty Player only supports a limited number (65) of viewpoints

[9] A more realistic approach would be to use a weighted graph, with the lengths of the paths as the weight of the vertices. This is possible of course, but then we would have to use a more complicated search-pattern, like Dijkstra's Shortest Path algorithm for example. This approach is out of scope for this POC, since it does not add any additional value to the challenges we address.

5.1.6 The User Interface

We have chosen for a minimalistic search interface. The user can select the viewmethod (either jump or guided tour) and provide text, as search-input.

Using ConcurTaskTrees (CTTs)[10], we come to the following overview:

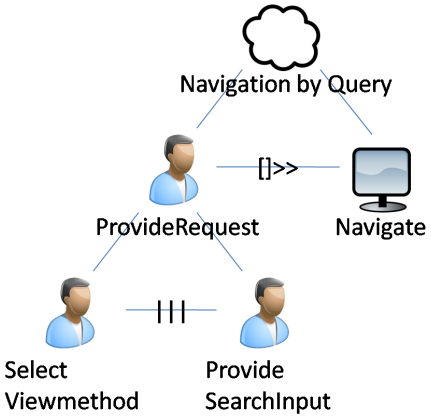

Figure 5.4: Concurrent Task Tree of the user interface

The overview is very limited, it tells us that when we provide a request, we need to select a viewmethod and provide a query, this input can be provided in any order (indicated by the `|||` operator). Afterwards the first task is terminated and the second task performs, based on information coming from the first task (indicated by the `[]>>` operator).

5.2 The result

We have created two versions of the application, a guided tour and a version in which we use a simple search-engine, which queries the ontology providing us with query based navigation.

[10] ConcurTaskTrees is a notation for task model specifications which has been developed to overcome limitations of notations previously used to design interactive applications. Its main purpose is to be an easy-to-use notation that can support the design of real industrial applications, which usually means applications with medium-large dimensions [6].

In both cases, the user-interface is very similar.

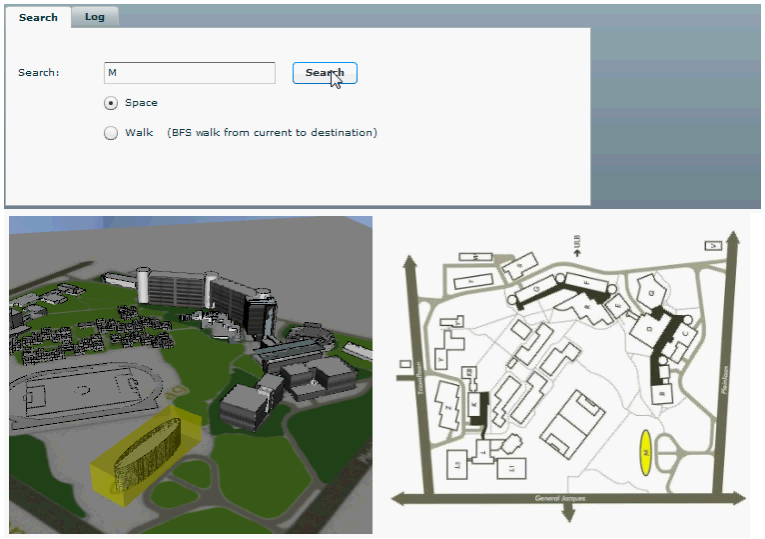

Figure 5.5: Screenshot of the application in action

Image 5.5 shows the version of the application in which we can query the ontology. The application shows that we have searched for building 'M', and where the result gets mapped to a building. The building gets highlighted, which is shown in both the 2d map and the 3d world.

There are two remarks concerning the image:

- The purpose was to highlight the building by changing its color. However, Vivaty doesn't allow to query the scenegraphs of binary scenes, so we placed a transparent box around the building as workaround.
- The 2d map was intended to be embedded as HUD, but Vivaty player is not stable enough to allow this, the map is now an embedded image in the webpage and is not a part of the scene.

5.2.1 Guided tour - Esplanada

The guided tour is the application we started with. It follows the 'travel control' way of navigation [56], or the 'Navigation by Query' [52], but in a very simplistic

way. Navigation by query implies that the user does not navigate himself, it is the system that guides the user through the Virtual World.

The esplanada website[11]. shows the visitor the location of interesting points/buildings on the campus by starting from a bird-eye view, zooming onto the requested location and zooming back out again. These movies are created in Google Sketchup and recorded as Flash movies. In fact, these movies are based on the same Google Sketchup model as we used for our POC. The 3d-world of our application mimics the behavior of the Esplanada introduction movies on the Esplanada website of the VUB.

The Esplanada demonstration was easy, simply proving the concept that we could navigate in a Virtual World. It contained hard-coded POIs (Point Of Interests), that are available for selection in a drop-down box, as well as hard-coded paths from a starting point, to the destination and back again. The fact that hardcoded POIs are used implies that we make no use of the ontology yet.

The viewpoints that were used for the movies were already available in the Google Sketchup project, so we started by extracting these viewpoints in the form of VRML and translated them to X3D using Vivaty Studio.

The Points Of Interests (POIs) are available in a dropdown list in the application. Of course, they are indicated by a name that is understandable for the user, like 'Restaurant', 'Building C' etc. A dedicated PathManager was written to return the viewpoints, as a path to follow when navigating from a starting point, typically the current location of the user, to a destination, the result of the query initiated by the dropdown box.

5.2.2 Query based navigation

A far more interesting application of the design is the use of the ontology to provide us with the destination viewpoint. Instead of supplying a dropdown box with predefined locations, we retrieve the locations from the ontology. The ontology is used to build a 'conceptual map' of the campus by connecting all viewpoints as a graph. This graph is managed by our PathManager.

The dropdown-box is replaced by a search option, which will be used to query the ontology. The result of the query will have some sort of geographical knowledge

[11] http://esplanade.vub.ac.be/drupal-5.9/

which will be used to query the PathManager for a path from the current position to the destination. The destination is a viewpoint, which is related to the result of the query.

The graph manager returns the route to follow based on its internal graph. Moving between viewpoints now becomes a fluent movement, simulating the path a user will take when walking from one point to another in a real-life environment.

5.2.3 Considered approaches

We examined some additional approaches for the POC, but had to abandon them due to technical reasons. Here is a small list that might be useful for future reference:

- The use of Level Of Detail (LOD). The idea was to change the image by one with a higher level of detail, when closer to the object. This way, we could only load the low-level image to start with, when approaching it, load the high-resolution version, so we could increase the level of details when zooming in. This does not work, since LOD requires that both versions are preloaded.
- Use of Javascript. In general we can say that Vivaty player and Javascript don't go well together. The player crashes often, reacts differently depending on the version. We therefor completely avoided the use of scripts within the X3D scene.
- The use of O3D. Google provides a tool to easily transfer **.x3d** files to **.o3d** files, in other words; to transfer x3d objects to o3d. This was very easy to implement, the result looked very good. The approach proved that the ontology did not need to change when changing the render engine. However, changing the viewpoints was a bit more challenging and this attempt was abandoned so we could focus on our main project.

5.2.4 Existing similar paraverses

A lot of universities have a Virtual University (VU) in Second Life, these VUs explore the aspects of online learning, collaboration etc. Besides that, a lot of universities have a pure visual 3d representation in Google Earth. These fall outside

the scope of this research, since they are based on virtual campusses for e-learning purposes. There are, however some projects that explore similar possibilities:

- F.E. Dozzi created for his/her master thesis a proof of concept (2000) to examine 3-d Virtual Worlds on the Internet using VRML, modeling the campus of the University of Amsterdam[12]
- Joris Gillis started an (unfinished) project at the University of Leuven where he uses Google Sketchup to create a 3d map of the campus and publish it on the web. [13]

None of these two projects used ontologies.

[12] http://www.math.vu.nl/~eliens/archive/scripties/fedozzi/
[13] http://www.student.kuleuven.be/~s0170633/projects/campus3D

Chapter 6

Overall limitations

Using information from the ontology in combination with semantic mappings gives us a relative flexible approach, but there are some limitations in our approach. The limitations fall under two categories; the semantic mapping being static and the pathmanager that uses a predefined graph.

6.1 Semantic mapping

The semantic mapping bridges the information from the ontology and the virtual environment. The main limitations of our approach lie in the fact that the mappings are static. Users cannot influence the mappings, this needs to be done by the maintainer of the application.

6.1.1 Adding new objects

Adding new virtual objects does not require code changes. The ontology needs to change, since the query now also must retrieve the new object. If the object is provided in the same way as the other virtual objects, the query (for instance: getAllBuildings) does not need to change.

However, the semantic mapping itself is static; adding new virtual objects requires that the mapping needs to be adapted. If no mapping is available, virtual objects

will not be displayed.

This as such is not a real limitation, the limitation lies in the fact that this mapping is implemented in text-files that reside on the server. The mappings are only editable by the application administrator, not by users.

6.1.2 Changing the queries

This mapping uses SPARQL to query the ontology, these queries are, just like the mappings, encoded in text-files. There are two issues with this approach.

The first issue is that the interfaces of these queries are fixed. If we would like to return more information from the query, or change parameter names, we need to adapt the query, but also the application code to deal with those changes.

Again, the queries are on the server, the queries cannot be modified by the user of the system. It is up to the maintainer of the application to decide which information is visible and in what way.

6.2 PathManager

The pathmanager queries the ontology for POIs and the connections between them. These connections are currently coded in the ontology, not derived via inference. If routes change, the ontology needs to be adapted. Since we use semantic mappings between the POIs and viewpoints in our virtual environment, we also need to adapt the mapping and the viewpoints. Again, these mappings are done at the server.

Chapter 7

Conclusion

Web 3.0 is on our doorstep, and although the defintions of what web 3.0 exactly is differ, most defintions agree that the semantic web is part of it, and possible also web3d.

In this thesis, we have used the semantic web (ontologies) to enrich the 3-dimensional web. Enrichment from data point-of-view, but also from usability point-of-view. We have done this by addressing three challenges:

- Initiating a Virtual World using ontologies
- Enriching the Virtual World with near real-time external data
- Using the ontology to help us with navigation

We have shown that we can use an ontology as base for Virtual Environments. We use semantic mappings to map our information of virtual objects to their 3-dimensional representation. Doing this, the virtual objects we show are becoming must more than a collection of polygons, textures, etc. that make up a Scene-Graph. The virtual objects in a virtual environment can be given a meaning, becoming objects of which we *know* what they are.

Additionally, we have shown that it is possible to enrich the virtual environment with external data. We can combine multiple external resources into a single new resources, which follows the idea of mashups and all this be done in a near real-time way. To be able to use different data-formats, we have used dedicated

translators that parse the data and provide it in RDF format, so we can query it in the same way we query the ontology.

Finally, we have used the ontology to help us with navigational issues. It is easy to get lost in virtual worlds, so instead of letting the user navigate, we let our application help the user. Our approach is based on 'Navigation by query'. We have introduced a path manager that, in combination with the semantic mapping, constructs a 'cognitive map' of the virtual world we are using. This map is internally represented as a graph, and the path manager can now construct our navigation path using a search-algorithm on this graph. It uses the current location as starting point and the destination will be tied to the result of a query to the ontology. This path is visualized as a sequence of viewpoints which we follow, giving the user the impression that he is walking in the virtual world.

7.1 Future work

Currently, our ontology has no knowledge of locations. We query the ontology for buildings on the campus, or in a more specific approach, we query the ontology for the virtual objects to display. The result is a list of buildings/virtual objects and the semantic mapping maps these to their visual representation. It is the visual representation that has geospatial knowledge.

A future improvement might be to extend the ontology with geospatial knowledge using GeoRSS encoding. The W3C Geospatial Vocabulary [1] defines a basic ontology and OWL vocabulary for representation of geospatial properties including points, lines, polygons, boxes and more.

We could also think of adding referential knowledge to the ontology, for example: building A is next to building B. This knowledge can be used, via reasoning, to create the graph that is used by our pathmanager for tourguides.

Another thing we can imagine is a more flexible approach towards external information. At the moment, external information is mapped to functioncalls at the client side. This mapping is static, it needs to be added by the application maintainer. We could imagine a more dynamic approach, where the *user* can connect the external information to the appropriate function call. This will lead to some additional challenges where the user also needs to have the ability to inject client-

[1] http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/

side calls, like Javascript functions.

Of course, we can extend the combination of ontologies and virtual worlds by adding multiple users, chat options. It would be very interesting to login, see if certain persons are online and directly have the ability to start a chat session, collaborate on documents etc.

Finally, the current limitations need to be addressed. The semantic mappings and the queries to the ontologies are static, they reside in files on the server and can only be modified by the application administrator. It would be interesting to see how this can be made more flexible.

Appendix A

Model simplification

Building B/C.

The unmodified version has a file-size of more then 17 Mb when exported to VRML. The reduced version has a filesize of 119 Kb, when exported to VRML. Afterwards we apply texttures. The export to VRML now takes 131 Kb, the size of the texture is 12 Kb. In X3D we can use binary (compressed) scenes, reducing the overall size to 20 Kb.

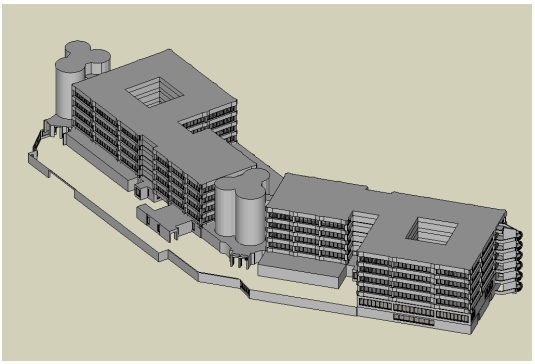

Figure A.1: Google sketchup - single building, original Sketchup version

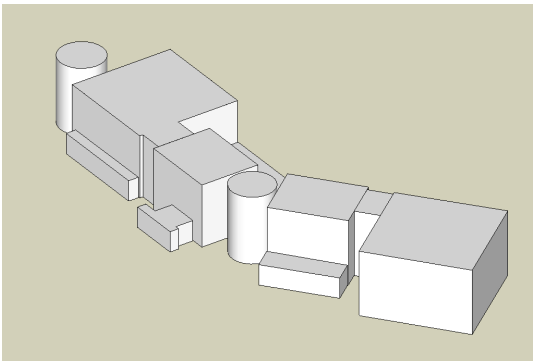

Figure A.2: Google sketchup - single building, simplified

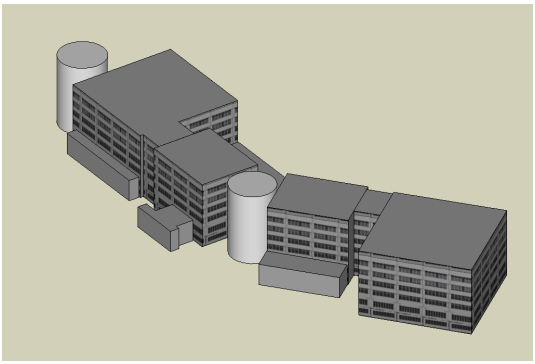

Figure A.3: Google sketchup - single building, simplified, textures applied

Appendix B

Source code - Embedding binary X3D

This appendix provides an example on how to embed a binary scene into an ascii-version. The binary version, called `compressed_scene.x3db` in the example, is a compressed version and thus downloads much faster.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN"
        "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1'
        xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
        xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>
    <head />
    <Scene>
        <ProtoDeclare name='CadProto'>
            <ProtoInterface>
                <field accessType="initializeOnly" name="binaryUrl" type="MFString"/>
            </ProtoInterface>
            <ProtoBody>
            <Group>
                <Inline>
                    <IS>
                        <connect nodeField="url" protoField="binaryUrl"/>
                    </IS>
                </Inline>
            </Group>
            </ProtoBody>
        </ProtoDeclare>
        <ProtoInstance DEF='A' name='CadProto'>
            <fieldValue name='binaryUrl' value='models/compressed_scene.x3db'/>
        </ProtoInstance>
    </Scene>
</X3D>
```

Appendix C

Utilities

An overview of the utilities that were used, as well as a short description of the utility.

C.1 Ontology

- **Protégé**
  Protégé is an open-source ontology editor and knowledge-base framework. It offers the ability to export ontologies to a variety of formats, including RDF and OWL.

- **Joseki**
  Joseki is an HTTP engine, that supports SPARQL queries. It is based on Jena, a Java framework for building semantic web applications.

C.2 3D modeling

- **Google Sketchup**
  Google Sketchup is a 3D tool that enables the user to model using real-world coordinates. It comes in two versions; Google Sketchup which is freely available and Google Sketchup Pro that offers additional functionality like export functionality.

- **Vivaty studio**
  Vivaty Studio is a tool that allows modification of 3d scenes. It can import from VRML and export to X3D. Vivaty Studio is no longer available, the company closed its doors in April 2010.

- **Vivaty player**
  Formerly known as 'Flux Player', a X3D (and VRML) viewer, a browser plugin.

- **Instant player**
  A X3D desktop viewer.

- **Xj3D**
  A Javabased X3D desktop viewer.

- **X3DEdit**
  An editor to modify X3D files, view scene graphs etc.

C.3 Application

- **Adobe Flex**
  A software developmint kit, used to create web-applications that are based on the Adobe Flash runtime environment.

Appendix D

METAR information

The information used in our proof of concept serves as a good example on how to translate an old legacy format to RDF and use it in our application.

METAR is typically used by pilots to retrieve information on the current weather and sky conditions.

This information is used as *external* information in our system, it is used to modify the scene dynamically.

We cannot query on any information that is available via this resource, this resource is used to dynamically color our sky, based on the available weather information.

D.1 Input

We query for weather information of Brussels, known as station `EBBR`. The METAR information for this station is:

```
EBBR 240550Z 22006KT CAVOK 1713 Q1016 NOSIG
```

METAR strings can contain a lot of information, from temperature, visibility to pressure, runway conditions, cloud coverage.

D.2 Result

Not all information from the input is returned in the RDF result, we are interested in the time, the temperature and clouds. The resulting RDF:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:weather="http://www.nauta.be/weather#">
    <rdf:Description rdf:about="http://www.nauta.be/weather/EBBR">
        <weather:temperature>17</weather:temperature>
        <weather:clouds>none</weather:clouds>
        <dc:date>2010-05-24T05:50+01:00</dc:date>
    </rdf:Description>
</rdf:RDF>
```

Bibliography

[1] http://www.x3dom.org/. [last accessed May, 9th 2010].

[2] http://www.web3d.org/x3d/. [last accessed May, 9th 2010].

[3] http://www.secondlife.com/. [last accessed May, 29th 2010].

[4] http://www.web3d.org/x3d/specifications/. [last accessed June, 3rd 2010].

[5] http://www.ajax3d.org. [last accessed Sept, 2009].

[6] http://giove.isti.cnr.it/tools/CTTE/CTT_publications/index.html. [last accessed May, 24th 2010].

[7] Grit - grokkable rdf is transformable, a collection of libraries/tools facilitating instrumental use of rdf. http://code.google.com/p/oort/wiki/Grit. [last accessed May, 9th 2010].

[8] Owl 2 web ontology language document overview - recommendation 27 recommendation 2009. http://www.w3.org/TR/owl2-overview. [last accessed May, 9th 2010].

[9] Owl web ontology language guide - recommendation 10 february 2004. http://www.w3.org/TR/owl-guide. [last accessed May, 9th 2010].

[10] The virtual reality modeling language - version 1.0 specification. http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html. [last accessed May, 9th 2010].

[11] Waseem Akhtar, Jacek Kopecký, Thomas Krennwallner, and Axel Polleres. Xsparql: Traveling between the xml and rdf worlds and avoiding the xslt pilgrimage. http://axel.deri.ie/publications/akht-etal-2008.pdf. [last accessed May, 9th 2010].

[12] Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. X3dom a dom-based html5/ x3d integration model. http://www.web3d.org/x3d/wiki/images/3/30/X3dom-web3d2009-paper.pdf. [last accessed May, 9th 2010].

[13] Tim Berners-Lee. Artificial intelligence and the semantic web. http://www.w3.org/2006/Talks/0718-aaai-tbl/. [last accessed May, 9th 2010].

[14] Tim Berners-Lee. Semantic web - xml2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html. [last accessed May, 24th 2010].

[15] Tim Berners-Lee. Web architecture from 50,000 feet. http://www.w3.org/DesignIssues/Architecture.html, Sept 1998. [last accessed May, 9th 2010].

[16] Tim Berners-Lee. Putting the web back in semantic web. http://www.w3.org/2005/Talks/1110-iswc-tbl/, 2005. [last accessed May, 9th 2010].

[17] Tim Berners-Lee. developerworks interviews: Tim berners-lee. http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html, 2006. [last accessed May, 9th 2010].

[18] David Booth. Rdf and soa. http://www.dbooth.org/rdf-and-soa/rdf-and-soa-paper.htm. [last accessed May, 9th 2010].

[19] Don Brutzman. Using extensible 3d (x3d) graphics for web-based accessibility and visualization. http://www.w3.org/WAI/RD/2003/12/Visualization/Brutzman.html. [last accessed May, 9th 2010].

[20] Jeremy J. Carroll. Matching rdf graphs. http://www.hpl.hp.com/techreports/2001/HPL-2001-293.pdf, 2001. [last accessed May, 9th 2010].

[21] Matthew Casperson. An introduction to googles o3d. http://www.pearsoned.co.uk/bookshop/article.asp?item=1178. [last accessed May, 9th 2010].

[22] Rudolph P. Darken. Wayfinding in large-scale virtual worlds. *Conference Companion of ACM CHI*, pages 45–46, 1995.

[23] Darcy DiNucci. Fragmented future. http://www.cdinucci.com/Darcy2/articles/Print/Printarticle7.html, 1999. [last accessed May, 9th 2010].

[24] Douglas C. Englebart. Knowledge-domain interoperability and an open hyperdocument system. *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 143–156, Oct. 1990.

[25] Lars Marius Garshol. Metadata? thesauri? taxonomies? topic maps! http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html. [last accessed May, 9th 2010].

[26] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, pages 199–220, Apr. 1993.

[27] Matthias Haringer and Steffi Beckhaus. Effect based scene manipulation for multimodal vr systems. In *Proceedings of the IEEE Virtual Reality Conference 2010, Waltham, MA, USA*, March 2010.

[28] Ivan Herman. Tutorial on semantic web - why "owl" and not "wol". http://www.w3.org/People/Ivan/CorePresentations/RDFTutorial/Slides.html#(114). [last accessed May, 9th 2010].

[29] Jesús Ibánez. *An intelligent guide for virtual environments able to answer fuzzy queries and tell stories from her own perspective*. PhD thesis, University of Murcia - Spain, 2004.

[30] Robert Jasper and Mike Uschold. A framework for understanding and classifying ontology applications. In *Proceedings of the Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW'99*, 1999.

[31] Jinyuan Jia, Guanghua Lu, and Yuan Pan. An efficient navigation algorithm of large scale distributed vrml/x3d environments. In *Proceedings of the Virtual Reality, second international conference, ICVR 2007*, 2007.

[32] Bin Jiang and Ferjan Omeling. Mapping cyberspace: Visualizing, analysing and exploring virtual worlds. http://www.casa.ucl.ac.uk/working_papers/paper11.pdf. [last accessed May, 9th 2010].

[33] Susan Kish. Second life: Virtual worlds and the enterprise. http://skish.typepad.com/susan_kish/secondlife/SKish_VW-SL_sept07.pdf, Sept. 2007. [last accessed May, 9th 2010].

[34] Frederic Kleinermann, Haitem Mansouri, Olga De Troyer, Bram Pellens, and Jesus Ibanez-Martinez. *The International Journal of Virtual Reality*, pages 53–58, 2008.

[35] Haïthem Mansouri. Using semantic descriptions for building and querying virtual environments. Master's thesis, Vrije Universiteit Brussel - Belgium, 2005.

[36] Mark May, Patrick Péruch, and Alain Savoyant. Navigating in a virtual environment with map-acquired knowledge: Encoding and alignment effects. *Ecological Phsychology*, pages 21–36, 1995.

[37] Patrick Murray-John. University ontology. http://www.patrickgmj.net/project/university-ontology, Feb. 2008. [last accessed May, 9th 2010].

[38] Patrick Murray-John. Thoughts towards a giant edugraph. http://www.patrickgmj.net/blog/thoughts-toward-a-giant-edugraph, Jan. 2009. [last accessed May, 9th 2010].

[39] Tim O'Reilly. What is web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2004. [last accessed May, 9th 2010].

[40] Tim O'Reilly and John Battelle. Web squared: Web 2.0 five years on. http://assets.en.oreilly.com/1/event/28/web2009_websquared-whitepaper.pdf, Oct. 2009. [last accessed May, 9th 2010].

[41] Bram Pellens. *A conceptual modelling approach for behaviour in virtual environments using a graphical notation and generative design patterns*. PhD thesis, Vrije Universiteit Brussel - Belgium, 2007.

[42] Qing-Jin Peng, Xiu-Mei Kang, and Ting-Ting Zhao. Effective virtual reality based building navigation using dynamic loading and path optimization. *International Journal of Automation and Computing*, pages 335–343, Nov. 2009.

[43] Denise Peters and Kai-Florian Richter. Enhancing wayfinding abilities in a large-scale virtual city by schematization. http://www.cosy.informatik.uni-bremen.de/staff/richter/pubs/peters-printed.pdf. [last accessed May, 9th 2010].

[44] Michael Rebstock, Janina Fengel, and Heiko Paulheim. *Ontologies-based Business Integration*. Springer, 2008.

[45] Stuart Russel and Peter Norvig. *Artifical Intelligence, a modern approach*. Prentice Hall, 2 edition, 2003.

[46] Glenna A. Satalich. Navigation and wayfinding in virtual reality: Finding the proper tools and cues to enhance navigational awareness. Master's thesis, University of Washington, 1995.

[47] K.H. Sharkawi, M.U. Ujang, and A. Abdul-Rahman. 3d navigation system for virtual reality based on 3d game engine. In *Proceedings of The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume 37, 2008.

[48] Prashant Sharma. Core characteristics of web 2.0 services. http://www.techpluto.com/web-20-services. [last accessed May, 9th 2010].

[49] Gregory Sherman. A critical analysis of xslt technology for xml transformation. http://ruleml.org/papers/CriticalAnalysisXSLT.pdf. [last accessed May, 9th 2010].

[50] R. Cailleau Tim Berners-Lee. Worldwideweb: Proposal for a hypertext project. http://www.w3c.org/Proposal.html, Nov 1990. [last accessed May, 9th 2010].

[51] Olga De Troyer, Frederic Kleinermann, Bram Pellens, and Ahmed Ewais. Supporting virtual reality in an adaptive web-based learning environment. In *EC-TEL*, pages 627–632, 2009.

[52] Alex van Ballegooij and Anton Eliëns. Navigation by query in virtual worlds. *Virtual Reality Modeling Language Symposium, Proceedings of the sixth international conference on 3D Web technology*, pages 77 – 83, 2001.

[53] Betsy van Dijk, Rieks op den Akker, Anton Nijholt, and Job Zwiers. Navigation assistance in virtual worlds. *Information Science Journal, Special Series on Community Informatics*, 6:115–124, 2003.

[54] Norman Walsh. Rdf twig: Accessing rdf graphs in xslt. http://rdftwig.sourceforge.net/paper/, 2003. [last accessed May, 9th 2010].

[55] Umi Laili Yuhana, Li-Lu Chen, Yung-Jane Hsu, and Wan-Rong Jih. An ontology based approach for searching neighborhood building. *3rd Information and Communication Technology Seminar*, 2007.

[56] Xiaolong Zhang. A multiscale progressive model on virtual navigation. *International Journal of Human-Computer Studies*, 66:243–256, Apr. 2008.

Index