

SOFTWARE DEVELOPMENT AND MIDDLEWARE
FOR VIRTUAL ORGANISATIONS

By
Pedro Omar Contreras Albornoz. Licenciado

MASTER OF PHILOSOPHY
SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE
THE QUEEN'S UNIVERSITY OF BELFAST
SEPTEMBER 2006

THE QUEEN'S UNIVERSITY OF BELFAST

Date: **September 2006**

Author: **Pedro Omar Contreras Albornoz**
Title: **Software Development and Middleware For Virtual Organisations**
Faculty: **Engineering**
Department: **School of Electronics, Electrical Engineering and Computer Science**
Degree: **Master of Philosophy**

Signature of Author

Signature of Supervisor

Signature of Supervisor

Signature of Internal Examiner

Signature of External Examiner

Acknowledgements

I would like to thank my family for all the support that they gave me.

I would like to thank my supervisors Prof. Fionn Murtagh, Dr. Peter Milligan and Dr. Paul Sage. Without them this work could not have reached its end. Particularly I would like to thank to Prof. Murtagh whose help has been essential to provide me with support, motivation, good advice and not less important funding.

I would like to thank Mohsen Farid for his support, friendship and guidance. Dimitri Zervas who helped me to develop the NodeMap software. And Thomas Huey my house mate.

Also I would like to thank all the secretaries and engineering support personnel in the School of Computer Science at Queen's University Belfast. They always were attentive to requests for help.

I would like to thank Stephanie Bachorz for fixing my English, for her comments and support.

Finally, thanks to the Staff Training and Development Unit at Queen's University Belfast which has funded in part my fees. AstroGrid project and its funding body the Particle Physics and Astronomy Research Council (PPARC), UK. WS-Talk project, funded by the European Commission under the CRAFT programme (COOP-CT-2004-006026).

Belfast, United Kingdom
September, 2006.

Pedro O. Contreras Albornoz

Abstract

Computer science is a fast changing field, the introduction of new technologies requires rapid adaptation and creation of new methods regarding –but not limited to– software development. This document presents tools from the open source community that are useful when developing software in a distributed manner. Examples are given regarding how these tools are used to develop software in real-life projects.

The AstroGrid project has implemented a series of tools which facilitate collaborative work in geographically distributed locations such as news, forum and discussion pages, all of them inspired by the open source and licencing philosophy. Using the same tools that the AstroGrid project uses on its software development process is not limited to AstroGrid or any of its partners in the International Virtual Observatory Alliance.

The AstroGrid case will be used to exemplify and illustrate virtual collaborative environments. A considerable part of this dissertation is related to AstroGrid, but not all. Chapter five is related to my work in the Sixth Framework Project, WS-Talk, “Web services communicating in the language of their community”.

This dissertation is separated into six chapters. Chapter one, “Introduction” gives a description of this dissertation, its goals and a brief summary of each chapter. Chapters two to four describe the context from which AstroGrid was born, the way that AstroGrid organised its work, focusing on the tools used to co-ordinate, manage, and synchronise the code developing process, and the resulting software of these processes. Chapter five, “NodeMap, an Ontology Constructor”, presents a user-friendly tool created by the author of this thesis for designing concept hierarchies within (but not limited to) the WS-Talk context. Finally the potential for future research and a conclusion are provided.

Chapters 2, 3, and 4 present the context of my software development work in the AstroGrid. Chapter 5 presents one area of my work in the WS-Talk project.

Table of Contents

Acknowledgements	iii
Abstract	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Goals	1
1.2 Thesis overview – content of this thesis	1
2 AstroGrid project	4
2.1 Introduction	4
2.2 Astronomy data spectrum	5
2.3 AstroGrid project	7
2.4 Grid computing technology	8
2.4.1 Web services	9
2.5 International Virtual Observatory Alliance	10
2.6 AstroGrid’s ten science problems	11
2.7 AstroGrid architecture	14
2.7.1 AstroGrid modules	15
2.8 Summary	17
3 AstroGrid software	18
3.1 Introduction	18
3.2 AstroGrid Workbench and AstroGrid Client Runtime	18
3.2.1 Workbench data discovery	19
3.2.2 Workbench data analysis	20
3.2.3 Workbench system services	21
3.2.4 Workbench helper	21
3.2.5 Workbench advanced	23
3.3 Summary	25

4	Continuous integration and collaborative work in AstroGrid	26
4.1	Introduction	26
4.2	Communication tool	27
4.2.1	Wiki	28
4.2.2	Forum	28
4.2.3	News	29
4.2.4	Content Management System	29
4.2.5	Other tools	29
4.3	Programming tools and languages	30
4.3.1	Coding standards and Java	30
4.3.2	Documentation	31
4.3.3	Unit test – JUnit	31
4.3.4	Bug tracking – Bugzilla	31
4.4	File access and security	32
4.4.1	Version Control System	32
4.4.2	Secure shell – SSH	35
4.5	Other tools	35
4.5.1	Apache web server	35
4.5.2	Apache TomCat	36
4.5.3	Building deployment automatically	36
4.6	Summary	37
5	NodeMap, an ontology constructor	38
5.1	Introduction	38
5.2	NodeMap	40
5.2.1	Ontologies and concept hierarchies	40
5.2.2	NodeMap features	42
5.3	NodeMap applet implementation	47
5.4	NodeMap and the concept hierarchy format	50
5.5	Search	52
5.5.1	Searching documents	52
5.5.2	Looking into the database	54
5.6	Servlet and database client configuration	57
5.7	Exploring NodeMap’s Java code	59
5.7.1	Client/Server communication on NodeMap	59
5.7.2	Compression on NodeMap	60
5.7.3	Understanding the graphics on NodeMap	65
5.8	Summary	72
6	Future work and conclusion	73
6.1	Current and future work	73
6.2	Conclusion	75
	Publications related to this thesis	76
	Acronyms and web references	78

Bibliography	83
A NodeMap UML diagrams	89
B Astronomy ontology	111
C Animal ontology	113

List of Tables

5.1	NodeMap toolbar	44
5.2	NodeMap applet toolbar description	49
5.3	Concept hierarchy text representation	51
5.4	Compression ratios for text files	61
5.5	Compression ratios for query (HTML) files	62
5.6	Data compression	63
5.7	Data decompression	64

List of Figures

2.1	Electromagnetic spectrum in different wavelengths	6
2.2	AstroGrid architecture	15
3.1	AstroGrid WorkBench data discovery	19
3.2	AstroGrid WorkBench data analysis	20
3.3	AstroGrid WorkBench system services	21
3.4	AstroGrid WorkBench helper	22
3.5	AstroGrid WorkBench advanced	23
4.1	CVS repository and client	34
5.1	NodeMap main screen	43
5.2	NodeMap tree and level view	45
5.3	Import CH from database	47
5.4	NodeMap applet	48
5.5	NodeMap seek documents	53
5.6	Concept hierarchy database schema	54
5.7	Commit to the database	57
5.8	Servlet configuration	58
5.9	Database configuration	58
A.1	TreeDataBase Java class	90
A.2	NodeMap Java class	91
A.3	Node Java class	92
A.4	BinaryTree Java class	93
A.5	OntoTableModel Java class	94
A.6	DrawingPane Java class	95

A.7	GZip Java class	96
A.8	Config Java class	97
A.9	PorterStemmer Java class	98
A.10	Util Java class	99
A.11	Cluster Java class	101
A.12	MoreTerms Java class	102
A.13	Matching Java class	103
A.14	TextProcessing Java class	104
A.15	Interpreter Java class	105
A.16	Query Java class	106
A.17	Search Java class	107
A.18	Connect Java class	108
A.19	TestServletConnection Java class	109
A.20	ServletUtilities Java class	110
B.1	Astronomy ontology displayed with NodeMap software	112
C.1	Animal ontology displayed with NodeMap software	115

Chapter 1

Introduction

1.1 Goals

The goals for the work presented in this document are the following:

- Present the AstroGrid project, and the context in which it has emerged and evolved, focusing on the tools used to develop software and distribute knowledge.
- Use the tools presented to develop software in a context different to that introduced by the AstroGrid project (different from the point of view of the problems to be solve). But similar in the way that the software development process and knowledge dissemination is approached.

1.2 Thesis overview – content of this thesis

Computer science is a fast changing field, the introduction of new technologies requires rapid adaptation and creation of new methods regarding –but not limited to– software development. This document presents tools from the open source community that are useful when developing software in a distributed manner. Examples are given regarding how these tools are used to develop software in real-life projects.

The AstroGrid project has implemented a series of tools which facilitate collaborative work in geographically distributed locations such as news, forum and discussion pages, all of them inspired by the open source and licencing philosophy. Using the same tools

that the AstroGrid project uses on its software development process is not limited to AstroGrid or any of its partners in the International Virtual Observatory Alliance.

The AstroGrid case will be used to exemplify and illustrate virtual collaborative environments. A considerable part of this dissertation is related to AstroGrid, but not all. Chapter five is related to my work in the Sixth Framework Project, WS-Talk, “Web services communicating in the language of their community”.

This thesis is separated into six chapters.

Chapter one, “Introduction” is the current chapter that gives a description of this thesis, its goals and a brief summary of each chapter.

Chapter two, “AstroGrid project” explains the context from which AstroGrid is derived, and brief discussion about grid technology is given. Additionally explanations about AstroGrid’s role in the International Virtual Observatory, and the ten science problems that AstroGrid aims to solve are described. Finally “AstroGrid Architecture” is explained together with the most important software components such as portal, community, workflow, registry, myspace, etc.

Chapter three, “AstroGrid software development” describes the necessary steps to deploy the AstroGrid software, and to use the AstroGrid client.

Chapter four, “Continuous integration and collaborative work in astroGrid” describes how AstroGrid supports it functionally, focusing on the tools used to co-ordinate, manage, and synchronise the code development process.

Chapter five, “NodeMap, an ontology constructor”, presents a user-friendly tool for designing concept hierarchies within (but not limited to) the WS-Talk context, which has been developed using similar tools to the work presented in chapter four.

Chapter six, “Future work and conclusion”, discusses the potential for future research.

Some suggestions on improving the NodeMap software are presented. Finally some conclusions from this dissertation are presented.

A list with related publications and reports produced by the author of this thesis is included in the section “Publications related to this thesis”.

A list of the most important acronyms and web references included in this thesis and cited papers is included.

Appendix A includes the UML diagrams of the most important classes of the NodeMap client and servlet applications –there are around 45 Java files including NodeMap client, NodeMap Servlet and NodeMap applet. Here just a subset is presented–. Appendices B and C includes ontologies that can be used to test the NodeMap software.

Chapter 2

AstroGrid project

2.1 Introduction

Astronomy is facing a massive data deluge due to new astronomical instruments, such as telescopes, satellites and detectors. New computer technology allows astronomical instruments to produce terabytes of images and catalogues. These datasets will cover the sky in different wavebands, from gamma rays, X-rays, optical, infrared, through to radio signals.

New survey telescopes, now being planned, will scan the entire sky every few days and produce data volumes measured in petabytes. These technological developments are changing the way astronomy is done. Thus, for example, the Hubble Space Telescope Archive [34] (HST) is growing at 1-2 TB per year. Likewise the Solar & Heliospheric Observatory [64] (SOHO) archive is growing at 1 TB per year. This is typical of major ground-based observatories around the world. This data growth is largely a consequence of the growth of detector size, which has been exponential for three decades. Quite soon now (available in 2007) Visible and Infrared Survey Telescope for Astronomy [80] (VISTA) will be using a Gpixel array. Additionally the Multi-Element Radio Linked Interferometer Network [47] (MERLIN) and the Wide Field Infrared Camera For UKIRT [86] (WFCAM) will also produce a massive amount of data that needs to be stored and analysed.

Massive data handling is not the only problem faced by this area. Further complications arise when the same astronomical object is referred to by different names in different

astronomical catalogues. This means that even when the data is stored in digital form (which is not always the case), different astronomical data centres store information in a different manner and using different standards. Although standards like FITS [22], SIAP [61], and VOTable [83] are widely used and accepted, clearly they are not enough to standardise the processes needed to integrate data across different data centres.

It is for this reason that over the past years the concept of the Virtual Observatory (VO) has emerged rapidly to address the data management, analysis, distribution and interoperability problems in astronomy. The VO is a system in which the vast astronomical archives and databases around the world, together with analysis tools and computational services, are linked together into an integrated facility.

2.2 Astronomy data spectrum

For a better understanding of the challenges presented by the massive amount of data that astronomers need to manage, it is important to understand the sources of this data. Different kinds of “telescopes” are needed to gather the information presented by the universe, information that is collected through observation on the electromagnetic spectrum as shown in Figure 2.1 [24].

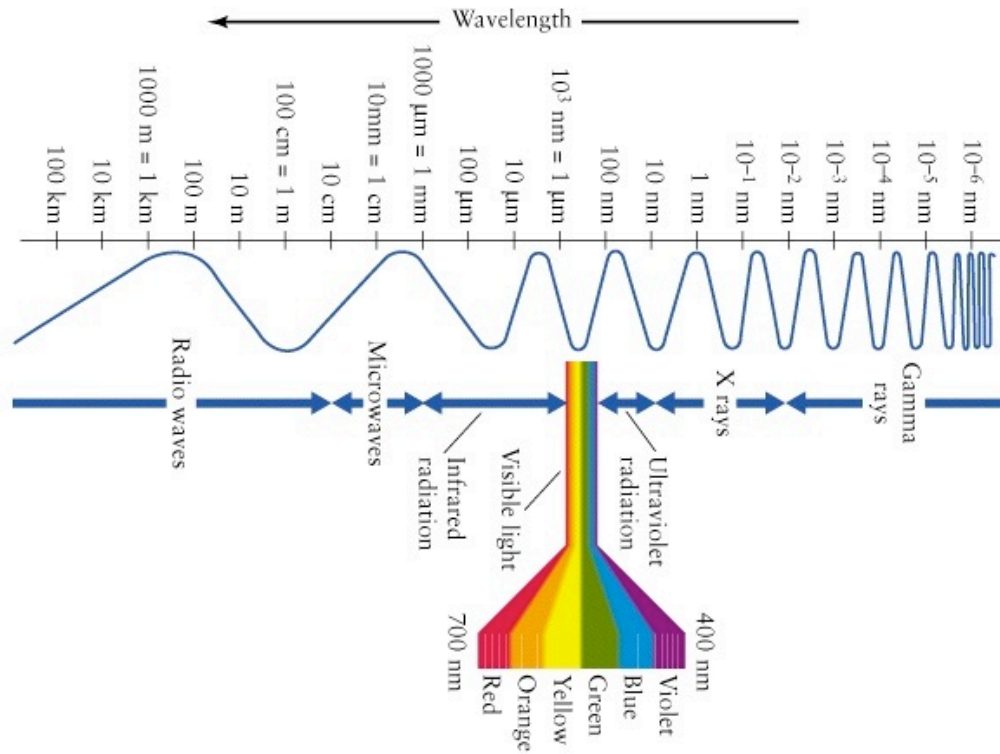


Figure 2.1: Electromagnetic spectrum in different wavelengths

The following list includes a brief description of the kind of instrument used to collect this astronomical data in different wavelengths.

- **Radio waves:** this is the longest wavelength, detectable by large radio dishes like the Very Large Array [81] in New Mexico, the Arecibo radio telescope [4] in Puerto Rico, and the Parkes Observatory in Australia, just to name a few. The radio sky is dominated mainly by gas clouds.
- **Submillimeter radiation:** instruments that study submillimeter radiation are either satellites or located on the earth's driest and highest places, like the submillimeter array in Mauna Kea, Hawaii. In this band complex molecules in dark clouds are studied.
- **Far infrared light:** can only be seen from space observatories like the Spitzer Space Telescope [65]. Sources of infrared light are embedded in dense regions of

gas and dust.

- **Visible and near infrared light:** this is from where “traditional” astronomy obtains its data. The instruments used to obtain information for this wavelength are the traditional astronomical observatories (e.g. Palomar Observatory [55]) and the Hubble Space Telescope.
- **Ultraviolet light:** this light is too blue for humans to see. The atmosphere blocks most ultraviolet radiation, therefore observations must take place mainly in space with instruments such as the NASA GALEX [27] satellite.
- **X-Rays:** this is the light beyond ultraviolet in the spectrum. Space telescopes like NASA’s Chandra [10] X-Ray Observatory and the European Space Agency’s XMM-Newton [91] can detect black holes and the X-ray galaxy chemical composition in this wavelength.
- **Gamma rays:** are studied with ground-based telescopes, satellites, and balloons. X-Gamma bursts, which are the most violent explosions in the universe, can be observed through instruments like the recently launched Swift gamma-ray burst satellite [69].

2.3 AstroGrid project

AstroGrid [5] is a £10 million UK government project – funded through the Particle Physics and Astronomy Research Council (PPARC) and by the European Commission – aiming to produce a more economic, faster and effective astronomy through the generation of open computer standards and interfaces. The AstroGrid framework helps to enable different data centres across the United Kingdom to publish services and data into a data-grid infrastructure accessible on the Web. Such a framework also makes it easier for the different data centres to interact; it offers a wide range of analysis and visualisation tools through a common interface.

The AstroGrid project was originally proposed in 2001 by leading investigators from the following seven institutions:

- School of Computer Science, Queens University Belfast
- Institute of Astronomy, University of Cambridge
- Institute for Astronomy, University of Edinburgh
- Department of Physics and Astronomy, University of Leicester
- Mullard Space Science Laboratory, University of London
- Jodrell Bank Observatory, University of Manchester
- Space Data Division, Rutherford Appleton Laboratory

The follow-on AstroGrid-2 project expanded the consortium to eleven institutions¹. The new institutions are listed below.

- Department of Computer Science, Royal Holloway, University of London
- School of Physics, University of Exeter
- School of Computing, University of Leeds
- Department of Physics, Bristol University
- Institute of Cosmology and Gravitation, University of Portsmouth

2.4 Grid computing technology

The terminology “Grid Computing” has been widely adopted in reference to the use of distributed computer resources across a network, working in a co-ordinated way and sharing not just information but computer processor cycles.

Grid technology is applied in a number of scientific areas, from climate simulation to finance and astronomy just to name a few. In order to work in all these different environments Grid networks follow a series of guidelines; for this standards and protocols are needed, and a range of work groups such as those affiliated to the Global Grid Forum [23] have been established.

¹Queen’s University Belfast was replaced by Royal Holloway, University of London

As mentioned above, astronomy is not an exception to this trend: astronomers and computer scientists are working to establish standards for a data-grid platform. This context is where AstroGrid is playing an important role as a member of the International Virtual Observatory Alliance (IVOA).

The need for “Grids” in astronomy is a natural evolution when the data analysis problem is considered, especially when presented with the massive data sets of today’s astronomical instruments.

Grid computing can be a difficult concept to understand, because it does not refer to a particular technology or standard, but to a set, aiming to share information and computer resources. For example peer to peer computing (see [12] [40]), Globus [29], Cactus [9] and Web Services can fall into this classification.

2.4.1 Web services

Web Services (WS) describes a distributed computer model that differs from CORBA or Java RMI (Remote Method Invocation) in that it is centred in simple Internet-based standards defined by W3C [84] and other standards bodies heavily supported by industry. WS are currently used by AstroGrid, and other VOs [87] [70] [44].

Simple Object Access Protocol, SOAP [63]: This is an XML-based communication protocol and encoding format for inter-application communication. It was originally conceived by Microsoft but currently the W3C XML protocol working group is in charge of the specification. SOAP usually exchanges messages over HTTP: the client POSTs a SOAP request, and receives either an HTTP success code and a SOAP success response or an HTTP error code. A SOAP message consists of an envelope containing an optional header and a required body. The header contains blocks of information showing how the message is to be processed. This includes routing and delivery settings, authentication or authorisation assertions and transaction contexts. The body contains the actual message to be delivered and processed. Anything that can be expressed in XML syntax can go in the body of a message.

Web Services Description Language, WSDL [89]: This provides the necessary structure to describe the Web Service functionalities, by means of supplying an XML grammar to be used in the Web Service. Usually this description can be automatically created using libraries freely available, such as `java2wsdl` and `wsdl2java` from Apache Axis [7].

Web Services Registry: This component helps adding, deleting, locating and retrieving web services. There are various implementation to achieve this, UDDI being one of the most widely used protocols. UDDI registry can be used by a public, or a private network. It offers a standard-based mechanism to classify, catalog, and manage web services, so that they can be discovered and consumed by other applications or services. AstroGrid uses IVOA registry standard, and not UDDI as this is too “business-oriented”.

2.5 International Virtual Observatory Alliance

AstroGrid is not alone in its attempt to standardise access to astronomy information. It is part of a worldwide effort to transform astronomical data repositories into a world Virtual Observatory [30] (VO), an organisation of which it is also a founding member. This initiative is led by the International Virtual Observatory Alliance [37] movement, which consists of sixteen active international members:

- AstroGrid. United Kingdom. <http://www.astrogrid.org>
- Australian Virtual Observatory. <http://avo.atnf.csiro.au>
- Astrophysical Virtual Observatory. EU. <http://www.euro-vo.org>
- Virtual Observatory of China. <http://www.china-vo.org/en/>
- Canadian Virtual Observatory. <http://services.cadc-ccda.hia-ihp.nrcnrc.gc.ca/cvo/>
- German Astrophysical Virtual Observatory. <http://www.g-vo.org/>
- Italian Data Grid for Astronomical Research. <http://vobs.astro.it/>
- Japanese Virtual Observatory. <http://jvo.nao.ac.jp/>

- Korean Virtual Observatory. <http://kvo.kao.re.kr/>
- National Virtual Observatory. USA. <http://us-vo.org/>
- Russian Virtual Observatory. <http://www.inasan.rssi.ru/eng/rvo/>
- Virtual Observatory of India. <http://vo.iucaa.ernet.in/~voi/>
- Hungarian Virtual Observatory. <http://hvo.elte.hu/en/>
- France Virtual Observatory. <http://www.france-vo.org/>
- Spanish Virtual Observatory. <http://laeff.esa.es/svo/>
- Armenian Virtual Observatory. <http://www.aras.am/ArVO/arvo.htm>

Within the scope of this dissertation it is important to highlight that AstroGrid itself is a Virtual Organisation, i.e. that its members work in institutions all over the United Kingdom. To achieve its goals (presented below) the work is based on collaboration, and most of the generated information, including almost all aspects of its work processes, is open to the public and accessible over the Internet.

2.6 AstroGrid’s ten science problems

The Scientific aims of AstroGrid are [85]:

- to improve the quality, efficiency, ease, speed, and cost-effectiveness of the on-line astronomical research
- to make comparison and integration of data from diverse sources seamless and transparent
- to remove data analysis barriers to interdisciplinary research, and
- to make science involving manipulation of large datasets as easy and as powerful as possible.

In order to make practical these scientific goals also a list of “practical goals” were defined, which are presented below:

- to develop, with their IVOA partners, internationally agreed standards for data, metadata, data exchange and provenance
- to develop a software infrastructure for data services
- to establish a physical grid of resources shared by AstroGrid and key data centres
- to construct and maintain an AstroGrid Service and Resource Registry
- to implement a working Virtual Observatory system based around key UK databases and of real scientific use to astronomers
- to provide a user interface to that VO system
- to provide, either by construction or by adaptation, a set of science user tools to work with that VO system
- to establish a leading position for the UK in VO work.

In order to achieve the above goals an extensive list of requirements [58] was established in the form of astronomical science problems. The intention of this list is to keep the focus of AstroGrid regarding the scientific problems that it should solve. From this list a subset of “ten science drivers” [71] were selected to organise AstroGrid work and deliverables; these are as follows:

1. The Brown Dwarf Selection [42]: A brown dwarf is a celestial body that is larger than a planet but does not have sufficient mass to convert hydrogen into helium via nuclear fusion as stars do. They are also called “failed stars”.

Although they should exist in large numbers, brown dwarfs are difficult to find using conventional astronomical techniques, because they are dim compared with true stars. A number of brown dwarfs have been identified, the first in the Pleiades star cluster in 1995. The first X-ray-emitting brown dwarf was detected in Chamaeleon dark cloud number I in 1998. A year later, several so-called methane dwarfs were discovered; these are thought to be older brown dwarfs that have cooled sufficiently over billions of years so that large amounts of methane could form in their atmospheres. One of the key questions in the area of star formation is the form of the stellar mass function at the lower end - i.e. what is the contribution of brown dwarfs to the stellar mass budget?

2. The Deep Field Surveys: The Hubble Deep Field [20] (HDF) is a small region in the constellation Ursa Major, based on the results of a series of observations with unprecedented resolution and sensitivity by the Hubble Space Telescope (HST), revealing about 3000 faint galaxies within a 3 arcmin-square region. Much effort is involved in aligning the data sets, and searching for significant correlations between sub-sets of properties.
3. Galaxy Clustering: The science aim for this problem is to address the evolution of galaxy populations in clusters. Thus, clusters of galaxies can be used to trace distribution of matter in the universe over large scales.
4. HiZ Quasars: The science challenge for this problem is the discovery of very high redshift quasars (QUASi-stellar radio source) in the redshift range $z=5$ to 10. For example the algorithm described by Richards et al [19] can be applied for discovering this kind of object. Quasars at larger redshifts are important because they provide vital clues to the processes involved in the formation of the first bound object.
5. Low Surface Brightness Galaxy (LSB) Discovery: LSB is a diffuse galaxy with a surface brightness that, when viewed from Earth, is at least one magnitude lower than the ambient night sky. Most LSBs are dwarf galaxies, but it is important to locate and understand the properties of such galaxies since they can contain significant mass.
6. Magnetic Storm Onset: Magnetic storms are frequently characterized by a sudden onset. Although these storms are not completely understood, they are more frequent during years of high sunspot number, also sometimes a magnetic storm can be linked to a particular coronal mass ejection. It is important to understand this phenomenon because these storms may interfere with the operation of electrical power lines and the operation of artificial satellites.
7. Solar Coronal Waves: Coronal waves have been observed on the sun regularly since SOHO (Solar and Heliospheric Observatory) was launched. They are generally related to solar flares and are a global phenomenon that can travel across the entire sun's disk in less than an hour; also they are less observable near strong

magnetic fields. The science goal in this problem is to determine whether coronal waves are Magnetohydrodynamic (MHD) fast mode waves (theory that predicts the motion of a conducting fluid in a magnetic field) occurring from a solar flare site, or if they are a global coronal mass ejection lifting off the surface of the solar disk.

8. **Solar Stellar Flare Comparison:** A solar flare is a violent explosion in the Sun's atmosphere, it produces electromagnetic radiation across the electromagnetic spectrum at all wavelengths from long-wave radio to the shortest wavelength Gamma rays. Stellar flares have also been observed on a variety of other stars. The science problem to solve here is to compare the attributes of solar and stellar flares in order to determine the mechanism causing flares, and determine if any extrapolation is possible.
9. **Solar Terrestrial Probes (STP) Solar Event Coincidence:** The primary goal of this is to understand our changing sun and its effects on the Solar System. Solar flares, Coronal Mass Ejections (CME), and progression of the solar cycle –just to name a few– cause electromagnetic disturbances in the Earth's magnetosphere, which –as mentioned before– can disrupt the telecommunications and power industries.
10. **Supernova Galaxy Environment:** Astronomers need to determine whether a supernova candidate is Type Ia or Type II. Determining galaxy redshift and morphology of the environment in which the supernova candidate was discovered will aid with its classification. The main activities in this science case involve searching archives for relevant image and spectroscopic data, where some database work is involved e.g. in cross-referencing the position of a supernova remnant (SNR, is the structure resulting from the gigantic explosion of a star in a supernova) candidate with other objects in the vicinity, such as galaxies or clusters of galaxies with known redshifts.

2.7 AstroGrid architecture

In concordance with the construction of the VO, AstroGrid has a modular architecture [43], open to all contributions from either data, services or resources.

Figure 2.2 shows the AstroGrid modular architecture at a high level which is explained in this section.

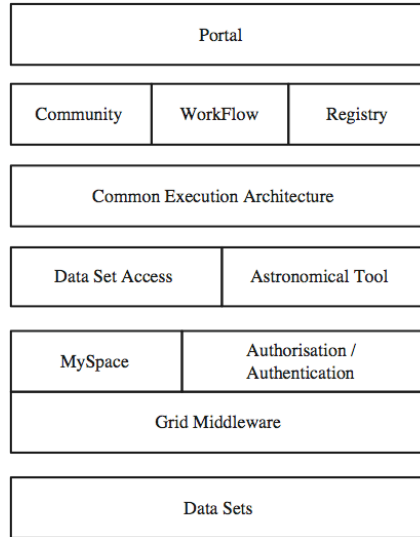


Figure 2.2: AstroGrid architecture

2.7.1 AstroGrid modules

Portal

The portal is a server-based component which provides an interface for accessing services in the VO. In AstroGrid all of the components which need to interact with the users will do so through a *Portlet*. A portlet gives a modular and flexible layer. Any developer is able to add his/her own functionality by wrapping it on a portlet.

Community

This module allows a group the ability to construct an online community with individuals and groups. A resource centre can then assign permission to use its resources to one or more groups within a community instead of having to name the individuals themselves. Within a community, the administrator of the community can assign rights to individuals and groups, including the right to add members and create groups.

AstroPass

This is a central server that stores users' credentials. Once permission has been assigned to a user, he/she decides how much information of its profile is passed to other VO portals.

Workflow

This enables the construction of complex tasks such as building queries and data analysis, upload/download of data and rendering the output in a different format such as tables or images.

Registry

This contains metadata descriptive information about resources available, which is based on IVOA standards. A resource can be a data set, web service, service, information on other registries, etc.

Common Execution Architecture

This concept –also called by the initials CEA– is an abstraction of application components and the necessary parameters to run it.

Astronomical tools

A number of essential astronomical tools such as object catalogue builder SExtractor [60] and photometric redshift analyser HyperZ [36] (which are freely available) will be incorporated to AstroGrid through a portal wrapper. The current release of AstroGrid WorkBench 2006.3.rc3 includes AstroScope, Helioscope, Aladin, TopCat, and other astronomical tools that will be presented later in this dissertation.

MySpace

This defines a virtual space to allocate both temporal and long term data, such as data sets generated by queries submitted to databases. MySpace is not necessarily in a local repository, but it will interact with the user as easily as if it were on a local machine.

Authorisation / Authentication

This is the component in charge of identification and granting of access to users, as well as maintaining security in AstroGrid.

Grid middleware

This allows AstroGrid to integrate different astronomical data centres and to share resources in a co-ordinated way.

Data sets

Data is allocated in a distributed way in different data centres, across the United Kingdom in AstroGrid's case, and across the globe once the Virtual Observatory alliance will be fully working.

2.8 Summary

In this chapter the AstroGrid project is presented, describing what are the scientific needs behind this project, its goals, and problems that it is meant to solve. Explaining AstroGrid's context makes it easier to understand its relationship with the Virtual Observatory and the processes needed to establish new standards and protocols within this area.

Chapter 3

AstroGrid software

3.1 Introduction

In this chapter some of the tools currently being deployed in AstroGrid software are presented, doing special emphasis in AstroGrid workbench application. The author of this dissertation has worked with a number of groups within the AstroGrid project such as the registry and portal groups, mainly in construction of software middle-ware (e.g. databases, performance evaluation, XQuery, XML marshalling/unmarshalling with Castor XML).

3.2 AstroGrid Workbench and AstroGrid Client Runtime

The AstroGrid Workbench is a client side application that comprises a set of user tools which provide access to Virtual Observatory services. The tools include a Datasource explorer (AstroScope), Workflow builder, a VOStore explorer, Query and Process manager (VO Lookout), Registry querying tool, etc, as well as the capability to call single services (SIAP, CEA, ConeSearch etc). In addition, Workbench provides helper functions such as the Sesame name resolver, coordinate transformation tools, Vizier and UCD (Unified Content Descriptors) tools.

The workbench is layered upon the Astro Client Runtime (ACR), a set of high level APIs to any registered IVOA compliant service, offering the fastest and easiest way for Astronomers and VO tool developers to gain programmatic access to IVOA resources. The ACR provides a high-level facade into AstroGrid services, accessible from HTTP /

XML-RPC / Java RMI (Remote Method Invocation) and GUI interfaces. It provides a single sign-on, single-configuration, and single cache for interactions with astrogrid servers, simplifying access to VO services.

3.2.1 Workbench data discovery

Figure 3.1 shows a screenshot of workbench data discovery tools.

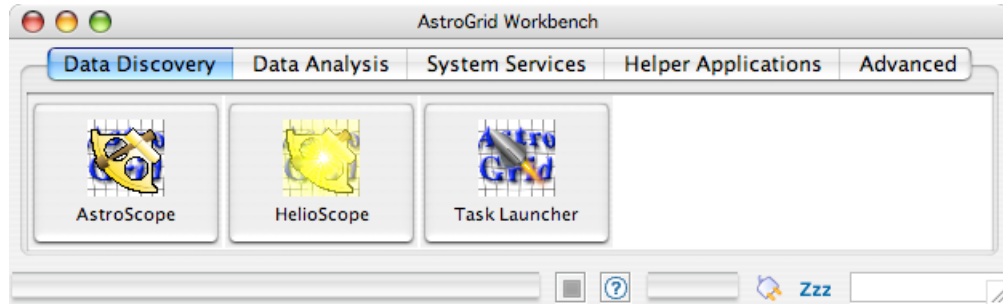


Figure 3.1: AstroGrid WorkBench data discovery

- **AstroScope:** [6] this allows to put in position an object in the sky along with a region. As a query is submitted a process is started looking up various catalogue and image servers asking them for data based around the query. If results are found then information is displayed in a graphical form on the screen allowing the user to surf the result in the form of various nodes. By default objects are seen in a "radial" form, requiring a "click of the mouse" to browse up and down the various nodes. Another option is the 'Hyperbolic' view which shows everything in a tree state where by a click of the mouse re-centers the selected node. The user may then select a particular object to have the data (including images) saved to MySpace or their computer.
- **HelioScope:** [32] this tool allows to query archives of images, spectra and catalogue data around a given position or SIMBAD [62] object, visualise the results and download data files either to a local machine or to a storage space on the Astrogrid storage system MySpace.
- **Task launcher:** this executes a single task on the AstroGrid system. A task can

be either querying an archive for data or running a tool that, for example, operates on data files or runs a model.

3.2.2 Workbench data analysis

Figure 3.2 shows a screenshot of workbench data analysis tools.

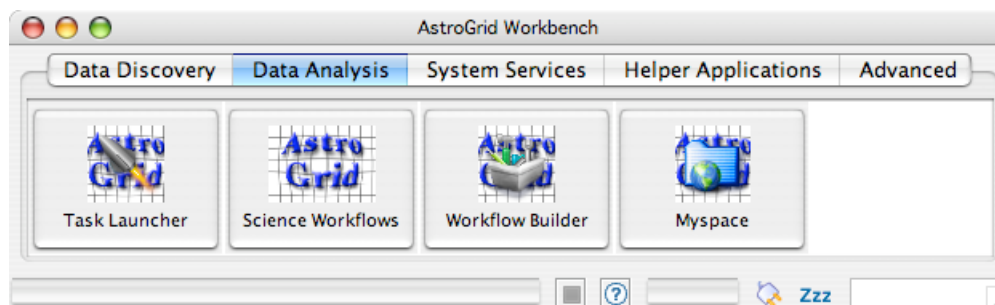


Figure 3.2: AstroGrid WorkBench data analysis

- **Science workflows:** this allows direct access and editing of workflows for commonly required analysis of astronomical data. This includes RedShift maker, colour cutter, solar movie maker, and CTIP [14] (Thermosphere Ionosphere Plasmasphere model) model.
- **Workflow builder:** this aims to accomplish a complex piece of work, for example an astronomical investigation. The workflow builder is designed to enable astronomers to design and develop these complex workflows in a simple and intuitive manner, whilst hiding much of the intricacies of the underlying XML document structure. The use of familiar drag and drop features, tooltips, examples and continuous error checking mean that a novice user can quickly produce simple workflows, and rapidly progress to ever more complex pieces of work. The workflow builder is also designed to interact seamlessly with other features of the workbench (e.g. MySpace and resource browser), and the wider VO community.
- **MySpace:** this is used to login into MySpace for browsing and utilisation of returned datasets, queries, and workflows in AstroGrid environment. See next section for a MySpace description.

3.2.3 Workbench system services

Figure 3.3 shows a screenshot of workbench system services.

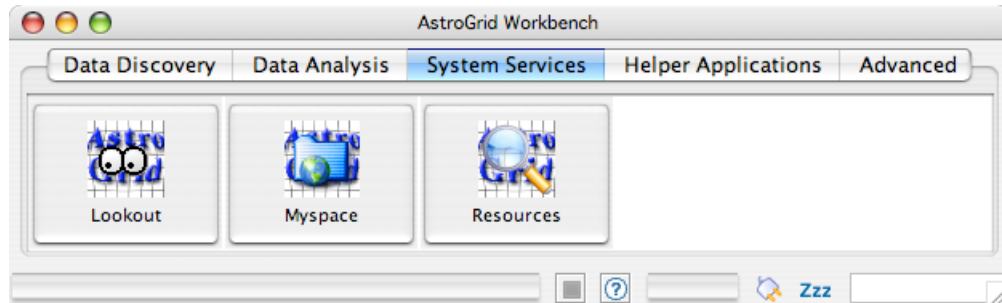


Figure 3.3: AstroGrid WorkBench system services

- **Lookout:** This is a tool for monitoring and controlling processes – queries, remote tool execution, and workflows. It uses the metaphor of an email program – each process is represented by a separate folder, which contains messages, information, results about this process. As well as tracking execution progress and accessing results, Lookout can also create, stop and delete processes.
- **Myspace:** This defines a virtual space to allocate both temporal and long term data, such as data sets generated by queries submitted to databases. MySpace is not necessarily in a local repository, but it interacts with the user as easily as if it were on a local machine.
- **Resources:** simple text search for available services and archives in the VO registry.

3.2.4 Workbench helper

Figure 3.4 shows a screenshot of workbench helper, that allows access to the following packages: Aladin, GAIA, SPLAT, TopCat, VisiVO, and VOSpec.

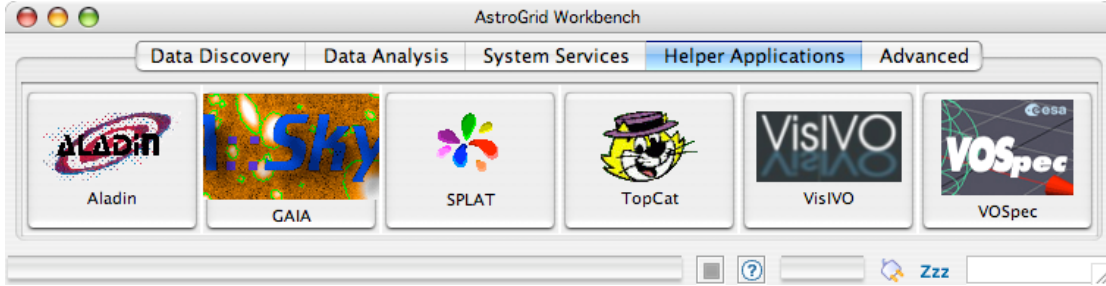


Figure 3.4: AstroGrid WorkBench helper

- **Aladin:** [1] this is an interactive software sky atlas allowing the user to visualise digitised astronomical images, superimpose entries from astronomical catalogues or databases, and interactively access related data and information from the Simbad database, the Vizier service and other archives for all known sources in the field. Aladin provides a visual summary of the multi-wavelength sky. It is particularly useful for multi-spectral cross-identifications of astronomical sources, observation preparation and quality control of new data sets (by comparing with standard catalogues covering the same region of sky).
- **GAIA:** [25] this is an image display and analysis tool. It provides the usual facilities of image display tools, plus more astronomically useful ones such as aperture and optimal photometry, contouring, source detection, surface photometry, arbitrary region analysis, celestial coordinate readout, calibration and modification, grid overlays, blink comparison, defect patching and the ability to query on-line catalogues.
- **SPLAT:** [66] this is a graphical tool for displaying, comparing, modifying and analysing astronomical spectra stored in NDF, FITS and TEXT files as well as the new NDX format. Also it can read in many spectra at the same time and then display these as line plots.
- **TopCat:** [76] this is an interactive graphical viewer and editor for tabular data. It has been designed for use with astronomical tables such as object catalogues, but is not restricted to astronomical applications. It understands a number of different astronomically important formats (including FITS and VOTable).

It offers a variety of ways to view and analyse tables, including a browser for the cell data themselves, viewers for information about table and column metadata, and facilities for 1-, 2- and 3-dimensional visualisation, calculating statistics and joining tables using flexible matching algorithms. Using a powerful and extensible Java-based expression language new columns can be defined and row subsets selected for separate analysis. Table data and metadata can be edited and the resulting modified table can be written out in a wide range of output formats.

- **VisIVO:** [79] this is a visualisation and analysis software for astrophysical data. VisIVO can handle both observational and theoretical data. It can be used both as a stand-alone application, that acts on local files, and as an interface to the Virtual Observatory framework, from which it can retrieve the data.
- **VOSpec:** [82] this is a tool to handle VO-SSAP (Virtual Observatory – Simple Spectral Access Protocol) compliant spectra.

3.2.5 Workbench advanced

Figure 3.5 shows a screenshot of advanced services. Currently Astro Runtime Interfaces is running under this option.

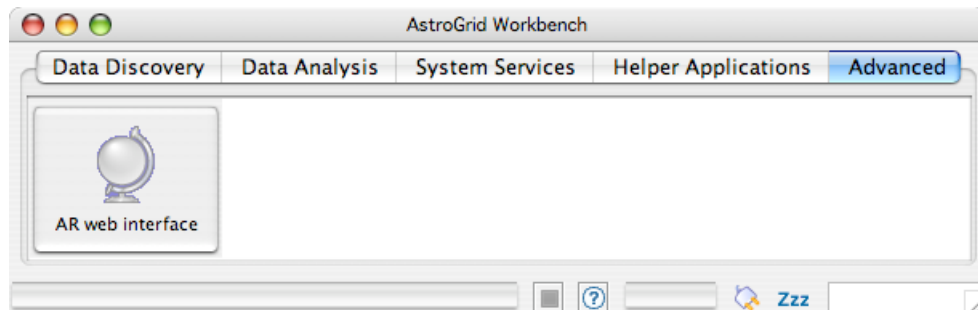


Figure 3.5: AstroGrid WorkBench advanced

Astro Runtime Interfaces is composed by the following modules:

- **cds**: access to services provided by Centre de Données astronomiques de Strasbourg (CDS), currently five services are available; **vizier** to access VizieR catalogues from CDS; **ucd**, which are Web Service for manipulating Unified Content Descriptors (UCD); **coordinate**, Astronomical Coordinate Web Service, from CDS; **glu**, Web Service to resolve GLU (Générateur de Liens Uniformes); and **sesame** Resolve object names to position by querying Simbad and/or NED and/or VizieR.
- **nvo**: NVO Services. Currently the **cone** service for Query catalogs using Cone-search services is available.
- **builtin**: intrinsic services that all others depend upon. Currently a **shutdown** for the ACR.
- **ivoa**: IVOA Standard Services. Where the following are available; **SIAP**, query for images from Simple Image Access Protocol services; **externalRegistry**, query an arbitrary registry service; **skyNode**, this is query for data from SkyNode services; **SSAP**, this is querying for Spectra from Simple Spectral Access Protocol (SSAP) Services; and **registry**, access the system-configured registry service.
- **util**: useful utility services, currently one service for functions for working with tables is available.
- **system**: this includes system components, such as, **RMI** to access information to the JavaRMI interface to the ACR; **apihelp**, for documents and provides access to functions of AR; **webserver**, to access information about the web-server component of ACR; **configuration**, to inspect and alter the configuration of the workbench and ACR; **browser**, to control the desktop web browser; **ui**, to control the main user interface of the workbench; and **help**, to control the workbench in-program help viewer.
- **astrogrid**: this includes the following AstroGrid services; **jobs**, execute and control workflows on remote job servers; **applications**, to work with remote applications; **stap**, to query for images from Simple Time Access Protocol (STAP) services; **myspace**, to work with MySpace – a distributed storage system, AstroGrid’s implementation of VOspace–; and **community**, the AstoGrid identity and

authentication system.

- **ui:** applications for working with the VO; **astroscope**, to control AstroScope; **helioscope**, to control AstroScope; **registryBrowser**, to control the registry browser UI; **myspaceBrowser**, to control the MySpace browser GUI; and **lookout**, to control the Lookout UI.
- **test:** services for internal testing.
- **plastic:** platform for Astronomical Tools Inter Connection; **hub**, the interface that a Plastic Hub should support.
- **dialogs:** this is the reusable UI dialogues; **toolEditor**, to display the remote invocation document editor as a dialogue; **resourceChooser**, to prompt the user to select a local file/MySpace resource/URL by displaying a resource chooser dialogue; **registryGoogle**, to prompt the user to select a registry resource by displaying a more advanced registry chooser dialogue (i.e. *selectResourcesAdqlFilter* to display the resource chooser dialogue, enabling only resources which match a filter; *selectResources* to display the resource chooser dialogue; and *selectResourcesXQueryFilter* to display the resource chooser dialogue, enabling only resources which match a filter).

3.3 Summary

This chapter presents some of the tools developed by the AstroGrid project, given special emphasis to the Workbench. This is relevant because the Workbench presents a good set of tools –some of them developed by the AstroGrid team, others not– that shows how integration is possible in the Virtual Observatory.

Chapter 4

Continuous integration and collaborative work in AstroGrid

4.1 Introduction

The AstroGrid development process is an open collaboration between researchers and technologists. To allow these people to work in partnership effectively, a range of web-based tools are required. In such a collaborative software development project there are different obstacles to overcome before the collaborative development can become effective. Firstly, good communication is essential in many aspects. Developers need to be aware of information regarding the project as a whole, information regarding specific sections, and they need to be able to directly communicate with other developers. The latter is of high importance in the AstroGrid case. Developers working within the project are assigned tasks in so-called development cycles, which are specified by astronomers through the definition of science cases. Developers will work on a certain task or project module for a few months before being moved to another area. In some cases more than one cycle or iteration is necessary to achieve a goal. In any case task assignment is under constant review by the project leaders, with a view towards reallocating human resources when and where they are needed.

The benefit of an approach like this is a highly cross-trained development team. For example in the case of a member's absence, another developer who has previously worked in that area can be assigned the task. The fact that, at some point, they would have worked in the particular area will mean they can quickly re-integrate themselves into the

development team for the particular project module. However, an adequate facility for communication must be in place for developers to be able to exchange ideas and to make and to respond to queries, so that the development process is as efficient as possible. One important issue in relation to this cyclic development process is development by second parties. As well as communication facilities there are other ways to ensure that the collaboration is as effective as possible.

Standards play an important role in facilitating code development by second and third parties. Standard forms of software design and standard coding styles all play a part in reducing development time. Standard software design methods will aid in code integration and coding style standards will help reduce the time required for second party developers to familiarise themselves with software. Standard forms of code documentation can also greatly aid second party developers.

4.2 Communication tool

To facilitate good communication practice, any person with expertise or simply an interest in the Virtual Observatory concept, Grid technologies or e-Science is free to post or comment upon news articles, to participate in the forum exchanges or to post documents making use of the AstroGrid portal. The portal provides an access point to the various communication tools available. The **Wiki** pages provide support for the posting of relevant documents for the project. Wiki is a leading-edge, web-based collaboration platform targeting the corporate Intranet world. TWiki [77] fosters information flow within an organisation. Some Wiki features are: web browser, auto links, text formatting, search, e-mail notification, structured content, attachment, templates and skins, statistics and plug-in resources. Further to the Wiki pages, the AstroGrid portal provides a **forum** facility which provides a number of grouped discussion areas. The groups deal with site-related topics, project-specific topics and general areas such as virtual organisations and Grids. To allow developers to be kept up to date with various information about articles posted, items of **news**, events and polls, a news section is available. The idea of a central portal providing access to the various communication platforms is essential to effective collaboration. There is no confusion over how to contact other members as

all contacts are made through the use of one central site. Further to these web-based community-oriented communication tools, one-to-one contact is necessary in order to ensure that developers can carry out their tasks efficiently. For example in the case of second party development, the ability to contact a previous developer to query some aspect of the work greatly reduces re-development time. To facilitate the necessary direct communications between project members, project contracted people keep close contact through **e-mail**, **telephone conferences**, **chat** programs and **regular meetings**.

4.2.1 Wiki

AstroGrid's Wiki pages <http://wiki.astrogrid.org> is a web site that allows the registered user to change the pages on a browser and to add their own pages. Pages can be formatted using either simple Wiki formatting marks (like **this** to make this) or the full range of HTML markup tags.

The site includes help pages, a tutorial and an experimental Wiki where you can try out adding pages, comments and whatever else you like. It is divided into a number of webs: one for the AstroGrid project, several general webs for VO, Grid, etc.

AstroGrid is not the only VO that uses Wiki pages. The following list includes some of the most important VOs:

- IVOA, <http://www.ivoa.net/twiki/bin/view/IVOA/>
- EuroVO, <http://wiki.eurovotech.org/bin/view/VOTech/>
- French VO, <http://www.france-vo.org/twiki/bin/view/ASOVFrance/>

4.2.2 Forum

The AstroGrid forum pages <http://forum.astrogrid.org> provide a number of grouped discussion areas. The groups deal with site-related topics, project-specific topics and general areas such as VOs and Grids. You can browse the forums and, if registered, add comments or new topics for discussion.

4.2.3 News

AstroGrid's news pages <http://www2.astrogrid.org/news> are the central part of the AstroGrid site. Here you will find articles posted, items of news events and polls. Any of these can be added by any registered user. The item will be checked by one of the admin staff and then released to the site. You can also comment on most items. Comments can be viewed in nested, threaded or flat form.

4.2.4 Content Management System

A content management system (CMS) is a web application designed to make it easy for non-technical users to add, edit and manage a website. Some of the most important characteristics of CMS systems are:

- automatically generate navigation elements
- making content searchable and indexable
- keeping track of users, their permissions and security settings.

AstroGrid-2 uses Plone (<http://plone.org/>) as CMS, but there are several others available, with implementations in a wide area of technologies, such as Java, PHP, C++, Perl, Phyton, ASP, just to name a few. Please refer to [45] and [17] for a comparison of different CMS systems.

4.2.5 Other tools

- e-mail and e-mail lists. This is essential for the day-to-day work.
- Messaging. Chat rooms and messaging programs can be important when communicating with other developers. AstroGrid uses the Jabber protocol, which allows different clients to connect when registered.
- Telephone. Teleconferencing is carried out often in AstroGrid.
- Meetings. This includes specific working group meetings as well as conferences.

4.3 Programming tools and languages

4.3.1 Coding standards and Java

Integration Code is tackled using Current Version System [15] to store, modify and keep track of code changes. This has various uses in software development practice helping returning developers to identify and understand changes that have been made since they last worked on a particular project area. JUnit [41] Test is used to prove code integrity and ensure quality of service. JUnit is a regression testing framework for code made with Java. Usually a unit test exercises some particular method in a particular context in order to prove that a specific piece of code does what it is supposed to do. Code bugs are tracked using Bugzilla [8], a “Defect Tracking System” or “Bug-Tracking System”. Defect Tracking Systems support individuals or groups of developers by keeping track of outstanding bugs in their software effectively. Once an error, misbehaviour or unexpected answer has been detected, this will be described and assigned to a developer in the form of a ticket which contains all the information related to the bug.

When a piece of code is developed, in order for further work to be effectively carried out, it is essential that the product is of high quality. To provide standard coding practices a number of tools are used in the AstroGrid project. A range of free Integrated Development Environment (IDE) tools such as Eclipse [18] and JBuilder Foundation [38] are used during the development process. These, in conjunction with code design standards (e.g. UML), are needed to establish the basis on which to allow different people in different places to modify each others’ code easily, the philosophy being that if everybody is using the same tools to build a product, the components of the product can be more easily modified or enhanced.

The main programming language used in AstroGrid is Java. Different standards such as UML, XML and Web Services are used for the process design to maintain consistency. When implementing code in Java a coding standard is utilised, which allows different people to understand the work of each other, maintaining coding style and consistency. This leads to a cleaner and easier coding process. AstroGrid code guideline standards are described in Vermeulen et al. [78], but a good set of standards can be found easily

on the Internet.

4.3.2 Documentation

Code management and documentation is addressed mainly through the use of standard Java Docs and Maven [46]. Maven is a Java project management and project comprehension tool. It is based on the concept of a project object model (POM) where all the Maven objects are a result of a well defined model. Builds, documentation, source metrics, and source cross-references are all controlled by the POM. Maven aims to make the developer's life easier by providing a well defined project structure, well defined development processes to follow and a coherent body of documentation that keeps developers aware of what is happening within the project. Maven alleviates a lot of what most developers consider a problem. This is essential in projects where there are not many people dedicated to the task of documenting and spreading the critical information about the project, which is necessary in order to dedicate resources to other critical tasks as coding and code testing.

4.3.3 Unit test – JUnit

A unit test is a small piece of code designed to test a specific functionality of the software being developed. JUnit [41] is a testing framework for Java, with it code examples can be implemented in order to see if the code being designed does behave how it is supposed to do. Some of the JUnit characteristics are the following [28]:

- It provides a template for writing tests.
- It allows to organise tests in a hierarchy.
- It allows you to automatically and easily execute tests.
- It decouples the test reporting from the execution, allowing to use different TestRunners with the same TestSuite.

4.3.4 Bug tracking – Bugzilla

Bug tracking systems are important since they help keep the software in good “health”, meaning that all bug reports are prioritised, organised, and assigned to be fixed.

Bugzilla [8] is the bug tracking system used by AstroGrid. The characteristics highlighted in its website (<http://www.bugzilla.org>) are the following:

- It improves communication.
- It increases product quality.
- It improves user satisfaction.
- It ensures accountability.
- It increases productivity.
- It adapts to multiple situations.

4.4 File access and security

One problem not included in the above section is security and the identification of which information is publicly accessible and which is not. As AstroGrid is an open project, almost every document is publicly accessible through the Internet. One exception is the code, as there are restrictions in place as to who is authorised to upload and to integrate code. Here SSH [67] protocol is used to allow secure communication between developers and the server repository. Though remote login is the primary use of SSH, the protocol can also be used as a general purpose cryptographic tunnel, capable of copying files, encrypting e-mail connections, and triggering remote execution of programs. Currently AstroGrid uses SSH Version 2 which operates over TCP. In its simplest mode of operation, it connects to a server, negotiates a shared secret key, and then begins encrypting the session. A username and password are passed over the encrypted session and, if authenticated, the server starts a command shell over the encrypted session.

4.4.1 Version Control System

Version control system software is a key technology when sharing code between programmers. There are different implementations of version control system (a list of packages can be found at [72]), but they share similar characteristics. Some important advantages are named by Dave Thomas and Andy Hunt [74]:

- It gives to the project the ability to undo the code to previous versions.
- It allows multiple developers to work on the same code base in a controlled manner.
- It keeps a record of the changes made over time.
- It support multiple software releases at the same time the development process can continue.

The most widely used version control systems used are Subversion [68] and the Concurrent Version System [15] (CVS). AstroGrid uses CVS to manage its code.

Version control system has named its processes with a common vocabulary. Some of the most important terms are the following:

- **Repository:** This is where the files are stored, often on a server.
- **Working copy:** This is the local copy of files from a repository, at a specific time or revision. All work done to the files in a repository is done to a “local” working copy.
- **Check-out:** This creates a local working copy from the repository. Either a revision is specified, a module, or the latest code is used.
- **Commit:** This happens when a copy of the changes made to the working “local” copy is written to the repository in the server side.
- **Update:** This merges changes that have been made in the repository (e.g. by other people) into the local working copy.
- **Module:** This is a given name to a project, or project part in the repository, which can also be a branch. When checking out a module name can be specify, doing so simplify the development process in large project by means of creating small modules.
- **Branch:** A set of files may be branched at a point in time so that, from that time forward, two copies of those files may be developed at different speeds and in different ways independently of the other.

- **Merge:** This brings together two sets changes to a file or set of files into a unified revision of that file or files.
- **Tag:** This refers to an important snapshot in time, consistent across many files. These files at that point may all be tagged with a user-friendly, meaningful name or revision number. This is relevant when a specific tagged version may be needed.
- **Conflict:** This occurs when two changes are made by different parties to the same document or place within a document. Since the software may not be intelligent enough to decide which change is “correct”, a user is required to resolve the conflict.

Figure 4.1 shows a simplified working representation of the control version system.

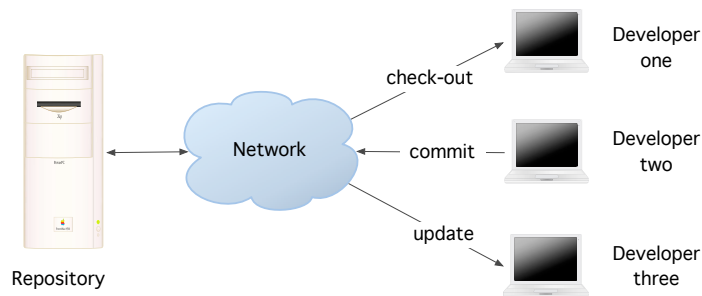


Figure 4.1: CVS repository and client

As mentioned above, usually the repository is created in a server. In the CVS case the repository can be accessed using the *Pserver* or *external* techniques. In Pserver mode, CVS runs a server process on the repository machine, and all clients connect to it. Pserver mode has some advantages:

- It is relatively simple to set up.
- It can enforce read-only users.
- It supports anonymous access.

However Pserver mode also has some drawbacks.

- It uses its own network port, and many corporate firewalls will not allow this traffic to pass.
- It uses very weak encryption of passwords, and file contents are transmitted in cleartext.
- It requires separate administration.

The other method used to access the repository is the external – ext – method, which works slightly differently. Here, CVS uses existing operating system commands to set up a data pipe (or tunnel) between the client and the server. The default version of external CVS uses “remote shell” (rsh). Also SSH can be configured to access the repository giving extra security when connecting to the repository.

4.4.2 Secure shell – SSH

Secure Shell [67] (SSH) is a set of standards that allows to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user automatically if needed. SSH provides confidentiality of data exchanged between the two computers using encryption and message authentication codes (MACs). SSH also supports tunneling, forwarding arbitrary TCP ports and X11 [90] connections; it can transfer files using the associated secure file transfer (SFTP) or secure copy (SCP) protocols. An SSH server listens on the standard TCP port 22 by default.

4.5 Other tools

4.5.1 Apache web server

With a market share around 70% [52], Apache [3] is by far the most used web server on the Internet today. The following list presents some characteristics of the Apache web server:

- Static and Dynamic web pages: Interaction with the most common dynamic content pages generators such as Perl, Python, Java Servlets and PHP.
- Highly configurable: Apache is built in a modular architecture, modules which can be added, removed or configured if needed.
- Security: Several methods of authentication are supported by the web server, such as, for example SSL encryption.
- Portability: Apache runs on almost any available platform today, and certainly is very well supported on Windows, Linux, Unix and OS/2.
- Licencing: All the Apache software is distributed under ASF (Apache Software Foundation) licencing, which basically means that the software can be used for free, subject to minor restrictions¹.

4.5.2 Apache TomCat

Apache TomCat [75] is a Java Servlets and JavaServer Page container (this technology allows to run Java code in a browser). TomCat is the official reference implementation for Servlet and JSPs, which means that one can be sure the Java code will run either in TomCat container or in any container that accomplishes with the standard definition.

Tomcat can contain a number of web application archive (WAR) files. When a web application is built with Java a WAR file should be created and copy within the “webapps” folder in TomCat. This will decompress the WAR file – process called deployment –.

4.5.3 Building deployment automatically

Additional to the tools already mentioned, there is a set of other Apache tools that help deploying the code automatically. Some of the most important are the following:

- Maven [46] is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project’s build, reporting and documentation from a central piece of information.

¹A good source of information regarding different licencing systems can be found on the Open Source Initiative <http://www.opensource.org>

- Ant [2] is used to create scripts, pretty much in the same way as the traditional make tools, with the advantage that Ant uses XML to define tasks to be carried out; making Ant scripts portable to different platforms.
- Jelly [39] is a tool for turning XML into executable code.

4.6 Summary

In this chapter we have a look to the software development process in AstroGrid giving special attention to the tools used to co-ordinate AstroGrid's work. Not just from the coding point of view, but also looking into the tools available to share knowledge and ideas.

Chapter 5

NodeMap, an ontology constructor

5.1 Introduction

Previous chapters have described the AstroGrid project and shown how it is managed and co-ordinated. It has been shown that AstroGrid can be considered as a virtual organisation, where each partner works from a different location. This is not restricted to AstroGrid or indeed to any of the other Virtual Observatory members. The same tools used to develop code and to co-ordinate work in AstroGrid can be used by any project with similar needs.

This chapter introduces the NodeMap ontology designer software for the WS-Talk [88] framework which was developed by the author of this dissertation (i.e. design, coding and documentation of NodeMap standalone, applet and servlet software) with the help of Dimitrios Zervas (i.e. NodeMap standalone and applet coding). In order to develop this software similar tools to the ones described in chapter four were used. Java is the main development language, TomCat is the servlet container, Apache is the web server, CVS the code sharing tool, and finally an eGroupWare (see <http://www.egroupware.org>) tool –this includes a Wiki, forum, calendar, file sharing facilities, etc.– which is used to disseminate knowledge.

NodeMap is a user-friendly tool to visually manage concept hierarchies within (but not limited to) the WS-Talk context (e.g. Astrogrid tools such as AstroScope and ElioScope

presented in chapter three are designed around hierarchical representation of information).

Currently the WS-Talk text mining area consists of a set of tools, which working cooperatively, help to retrieve a set of documents (or web services), based on applying different techniques to them. An approximate exploitation sequence is as follows:

- a) Load a set of documents to retrieve more frequent and relevant terms.
- b) Create concept hierarchies based on more relevant terms.
- c) Load concept hierarchies into the WS-Talk system (i.e. WS-Talk repository [13]).
- d) Load documents into the WS-Talk system.
- e) Query the WS-Talk system.

Concept hierarchy (CH) generation (point b in the above sequence) is of special importance since it constitutes the bridge between the terms to be searched and the documents to be queried. Therefore, this is used to provide indexes for document matching.

Experts in a relevant area create concept hierarchies, and once they are stable, which means that very few changes are introduced later into the ontology, they can be uploaded to the WS-Talk system.

NodeMap is an intuitive and easy to use tool that aims to help the expert user to create concept hierarchies in a visual way, and to export the results in a WS-Talk compatible format.

In this chapter, after introducing NodeMap, the following is presented; how the concept hierarchies are read and stored, how documents are searched, and how the NodeMap client interacts with its server component. This is followed by a brief look into the code.

5.2 NodeMap

5.2.1 Ontologies and concept hierarchies

Before presenting NodeMap software it is necessary to point out the importance of ontologies within the WS-Talk framework.

The word **ontology** has a long history in philosophy, and more recently it has entered the computer science field, where it has become important to the artificial intelligence community. From our point of view an ontology is the formal description of concepts and relationships within a domain, where a domain is a specific *subject* that can be defined as an information space.

WS-Talk uses *concept hierarchies* as a “specification mechanism” for information spaces within a domain. Section 5.4 of this document gives a detailed explanation of how CH are defined and used by NodeMap.

In this document the words ontology, concept hierarchy and hierarchy are use interchangeably.

Building a CH can be a complex process. Michael Denny [16] gives an approximation of the steps needed to create an ontology, which are presented below:

1. **Acquire domain knowledge:** Gather appropriate information and expertise that will define, with consensus and consistency, the terms used formally to describe things in the domain of interest. These definitions must be collected so that they can be expressed by a common language selected for the ontology.
2. **Organise the ontology:** Design the overall conceptual structure of the domain. This involves identifying the domain’s principal concrete concepts and their properties, identifying the relationships among the concepts, creating abstract concepts as organising features, referencing or including supporting ontologies, distinguishing which concepts have instances, and applying other guidelines of your chosen methodology.

3. **Flesh out the ontology:** Add concepts, relations, and individuals to the level of detail necessary to satisfy the purposes of the ontology.
4. **Check your work:** Reconcile syntactic, logical and semantic inconsistencies among the ontology elements. Consistency checking may also involve automatic classification that defines new concepts based on individual properties and class relationships.
5. **Commit the ontology:** Necessary for any ontology development effort is a final verification of the ontology by domain experts and the subsequent commitment of the ontology by publishing it within its intended deployment environment.

There are a good number of tools to create ontologies, and we will look at two of the most well known:

1. **Protégé:** [59] This is a free, open source, Java based ontology editor and knowledge-base framework. The Protégé platform supports two main ways of modelling ontologies:
 - **The Protégé–Frames editor** enables users to build and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC) [53].
 - **The Protégé–OWL editor** enables users to build ontologies for the Semantic Web, in particular in the W3C’s Web Ontology Language (OWL) [54].
2. **Thinkmap:** [73] This is a software platform for developing tailored and interactive visualisation interfaces from complex data. Thinkmap SDK includes a set of Java libraries for data visualisation, with easy integration to different data sources (flat files, database, XML, and Web Services). Some other advantages presented by this product are:
 - Real time data interaction.
 - The client can be deployed as a “thin” web-based client, standalone .exe or other modes.
 - Multiple Thinkmaps can be run simultaneously to represent the same data.

- Highly documented, with examples and templates available.

Within the WS-Talk context Thinkmap presented a very good option for data representation and mining. The price of this tool in order of the US\$ 20.000 for a license, therefore a big disadvantage.

The tools mentioned above presents a good option to implement ontologies, but within the WS-Talk context a tailored application is needed, in particular a tool that can be adapted to the use of WS-Talk text retrieval libraries and that can be deployed as standalone application as well as within a web browser.

5.2.2 NodeMap features

The main features of NodeMap include the following:

- Start a new concept hierarchy and then connect nodes.
- Import an existing ontology to add, remove, or reorganise information.
- Import relevant terms obtained from text mining techniques applied to a set of texts (or documents), and then organise these terms in a new ontology that can be saved to a file or uploaded directly to a WS-Talk compliant database.
- Use concept hierarchies currently uploaded into the WS-Talk system to search a document.

Figure 5.1 shows a screenshot of NodeMap version 1, which can be downloaded from <http://thames.cs.rhul.ac.uk/wstalk> (under the “Prototypes” link); an example file containing an astronomy CH can also be downloaded from that address for testing purposes. Once a concept hierarchy has been designed, it can be exported to a file that can be loaded into the WS-Talk system, such as the “repository demonstrator” [13] presented at <http://thames.cs.rhul.ac.uk/wstalk/prototype.html>. Should direct interaction with the database be needed, an option to commit a CH directly to the database is available.

Table 5.1 shows the list of features currently used in Version 1 of NodeMap. The first column shows the icon, the second column shows the keyboard shortcut, and the third column shows a hint or description representing the action carried out by the icon.

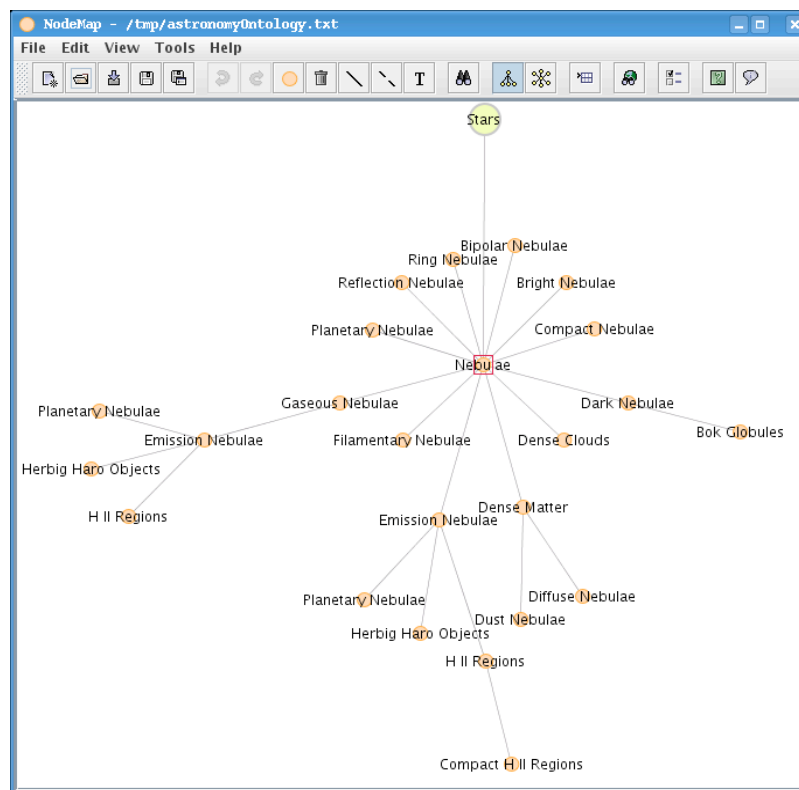


Figure 5.1: NodeMap main screen

- **New node map:** This option cleans the current NodeMap design board leaving a new empty working area.
- **Open file:** This opens a file storing a CH according to the standard defined in Table 5.3.
- **Import terms:** This option opens a list of terms stored in a file.
- **Save:** This saves the CH tree presented in the NodeMap designer board into a file.
- **Save as:** Saves CH tree presented in the NodeMap designer board into a new file.
- **Undo and Redo:** This feature includes the ability to undo and redo activities carried out on the CH tree, such as renaming nodes, disconnecting nodes, connecting nodes, etc.




















Icon	Short cut	Description
	Ctrl - N	New map
	Ctrl - O	Open file
	Ctrl - I	Import terms
	Ctrl - S	Save
	—	Save as
	Ctrl - Z	Undo
	Ctrl - X	Redo
	—	New node
	Delete	Delete node
	Ctrl - C	Connect node
	—	Disconnect node
	—	Rename node
	Ctrl - F	Find node
	Ctrl - T	Tree view
	Ctrl - L	Level view
	—	Seek Documents
	—	View Preferences
	—	Help
	—	About box

Table 5.1: NodeMap toolbar

- **New node:** This option creates a new node which is placed in the upper left corner of the NodeMap design board with the tag **node** and a sequence number.
- **Delete node:** This deletes the node currently selected, all the child nodes will also be deleted.
- **Connect node:** Nodes that are disconnected can be connected to other nodes by selecting this option, and a line from the node leading to the mouse pointer will be shown. When clicking on another node a new connection will be produced.
- **Disconnect node:** Nodes are connected with visible lines. These connections can be rearranged or modified. This option will disconnect the node currently selected.

- **Rename node:** This changes the name of a selected node.
- **Find node:** This searches for exact matches in the CH tree (not case sensitive). Should a match be found the tree will be expanded automatically. If the “seek documents” option is active, related documents will be shown.
- **Visualisation mode:** Two visualisation modes are available, one that gives access to the whole tree, as displayed in Figure 5.2, and a “level view” that allows visualising large ontologies. This consists of visualising cross sections or “tree levels” of an ontology, where an indicator (upper left corner) shows the current position within a tree. In this way the graphical space within the NodeMap drawing board is maximised.

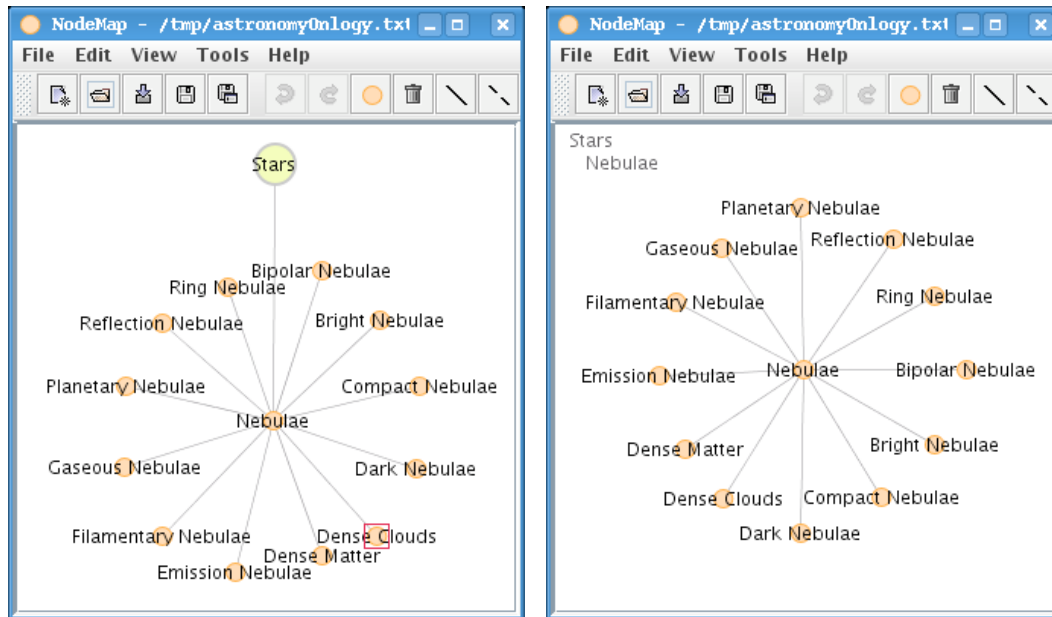


Figure 5.2: NodeMap tree and level view

- **Seek documents:** When this option is selected, a new panel appears on the screen. This panel shows document matches when a node is expanded or double clicked with the mouse pointer. In turn each document presents a list of associated terms, which trigger a new search when clicked. Additionally the text search panel contains tree icons; *back*, *forward*, *home*, for browsing the document's history.
- **View preferences:** This shows the pointers to the server side of the application, where a Servlet and a database should be configured if searching documents or committing ontologies to the database is needed.
- **Help:** This launches the browser with a URL pointing to a HTML help file stored locally.

Another important feature is the implementation of **working spaces**. A workspace is the combination of a hierarchy that has not been finished by the user, with loose nodes in the design board. This working space can be saved into a temporary file, to be completed later. With this option flexibility when designing an ontology is introduced.

Additionally a **file viewer** (see Menu → View → CH Table) feature is available allowing the visualisation of a concept hierarchy in raw format when being stored in a file or in a database, with the column ID (which is a sequence number), parent ID (which indicates parent node within an ontology), and label (which shows the node name), as shown in Table 5.1 in section 5.4 of this document.

Other important features included in the menu are: **Import from DB** (as shown in Figure 5.3) allowing direct communication with the ontologies stored in the DB. **Commit to the DB** which allows to insert a new CH into the database. **Import unorderer** which allows to import a list of terms (separated by and end of line) from a text file, to be organised onto a CH.

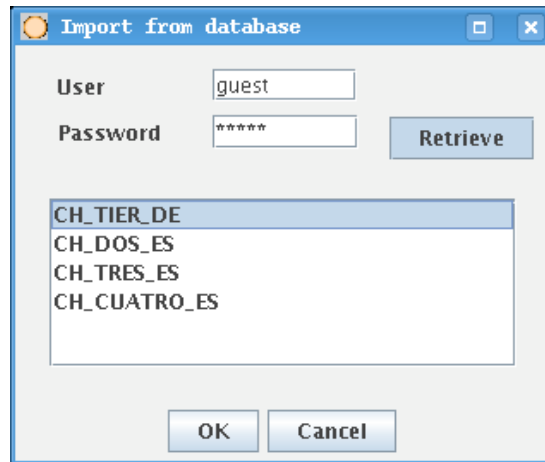


Figure 5.3: Import CH from database

5.3 NodeMap applet implementation

A Java standalone application offers a good solution for the ontology design problem. It gives flexibility from the developer point of view, but it also has some disadvantage as it cannot be displayed with a browser, or be embedded into a web site. In order to solve this problem a Java applet version of NodeMap have been designed and implemented.

Figure 5.4 shows a screenshot of NodeMap applet implementation running under Mozilla Firefox [21].

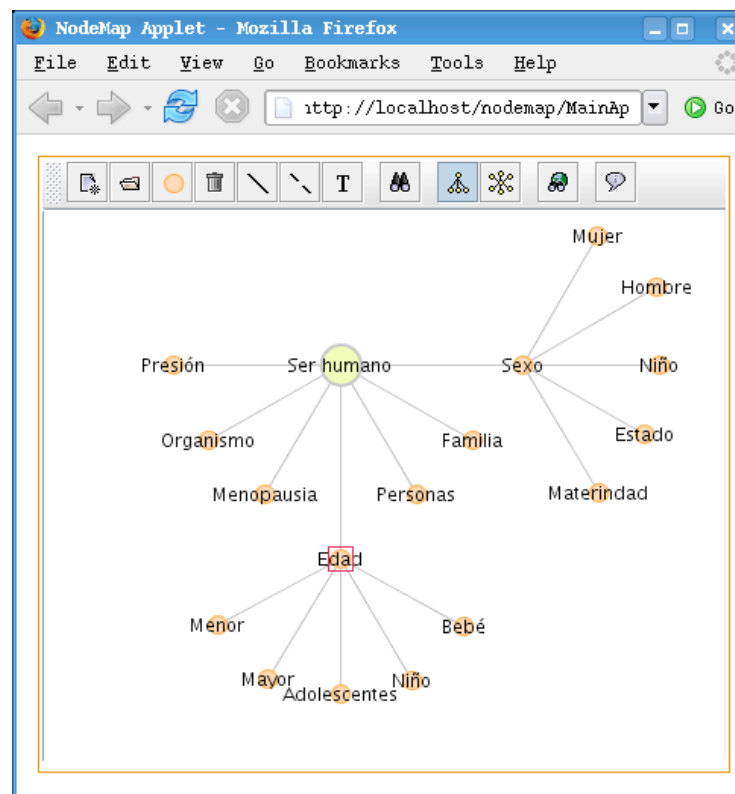


Figure 5.4: NodeMap applet

Table 5.2 shows the applet’s toolbar, which implements similar features as those used with NodeMap’s standalone version, but with some limitations, e.g. the applet application does not implement the feature of exporting ontologies into files, or undo and redo (for now). Another difference is that the open icon connects directly to the database and retrieves the existing ontologies which are presented to the user. Once one is selected it can be opened into the applet.

- **New node map:** This option cleans the current NodeMap design board leaving a new working area empty.
- **Open CH:** This connects to the database and retrieves the ontologies stored there.
- **New node:** This option creates a new node which is placed in the upper left corner of the NodeMap design board with the tag **node** and a sequence number.
- **Delete node:** This deletes the node currently selected, and all the child nodes













Icon	Description
	New map
	Open a CH from DB
	New node
	Delete node
	Connect node
	Disconnect node
	Rename node
	Find node
	Tree view
	Level view
	Seek Documents
	About box

Table 5.2: NodeMap applet toolbar description

also will be deleted.

- **Connect node:** Nodes that are disconnected can be connected to other nodes; by selecting this option a line from the node leading to the mouse pointer will be shown. When clicking on another node a new connection will be produced.
- **Disconnect node:** Nodes are connected with visible lines. These connection can be rearranged or modified. This option will disconnect the node currently selected.
- **Rename node:** This changes the name of a selected node.
- **Find node:** This searches for exact matches in the CH tree (not case sensitive). Should a match be found the tree will be expanded automatically. If the “seek documents” option is active, related documents will also be shown.
- **Tree visualisation:** This visualises a concept hierarchy in the form of a tree where the branches can be expanded or contracted in a “level view” that allows visualising large ontologies.
- **Level visualisation:** This consists of visualising cross sections or “tree levels” of an ontology, where an indicator (upper left corner) shows the current position

within a tree. In this way the graphical space within the NodeMap drawing board can be maximised.

- **Seek documents:** When this option is selected a new panel appears on the screen. This panel shows document matches when a node is expanded or double clicked with the mouse pointer. In turn each document presents a list of associated terms, which triggers a new search when clicked. Additionally the text search panel contains three icons; *back*, *forward*, *home*, for browsing in the document history.
- **Help:** This launches the browser with a URL pointing to a HTML help file stored locally.

5.4 NodeMap and the concept hierarchy format

Table 5.3 shows the file format currently managed by NodeMap to produce concept hierarchies, that is a sequence ID as the first field, a parent ID as second, and finally the node name or label (please refer to Appendices B and C for complete examples). In order to construct a CH, the parent ID field is related with the ID field. In this way complete concept hierarchies can be easily described. For instance, in the Table 5.3, “Bipolar Nebulae” and “Bright Nebulae” have as parent ID “2”, which means that “Nebulae” is their parent node. Following the same logic, “Nebulae” have as parent ID “1”, as this is the concept hierarchy root (Stars). Based on this structure NodeMap constructs a tree representation of the data.

Currently the encode standard used to read and write files is UTF-8, without byte order mark (BOM), and the tab character to separate the fields. For more information regarding this encoding please visit the following link: http://www.unicode.org/faq/utf_bom.html.

Additional to this format a new database schema is now being specified in conjunction with (WS-Talk partners) IRIT¹ and Lucky Eye², which will be supported in the next

¹Institut de Recherche en Informatique de Toulouse. France

²LuckyEye Group, Istanbul. Turkey

ID	Parent ID	Label
1	0	Stars
2	1	Nebulae
3	2	Bipolar Nebulae
4	2	Bright Nebulae
5	2	Compact Nebulae
6	2	Dark Nebulae
7	6	Bok Globules
8	2	Dense Clouds
9	2	Dense Matter
10	9	Diffuse Nebulae
11	9	Dust Nebulae
12	2	Emission Nebulae
13	12	H II Regions
14	13	Compact H II Regions
...
id_n	$P id_n$	L_n

Table 5.3: Concept hierarchy text representation

NodeMap release. With these partners also a data exchange format in XML has been planned. Additionally some other rules are applied to the graphical representation:

- a) There is only one root node.
- b) A parent node can have as many child nodes as necessary.
- c) A child node can have just one parent node.
- d) Child nodes cannot be connected to other child nodes.

NodeMap, when designing a concept hierarchy, checks all the above conditions.

5.5 Search

5.5.1 Searching documents

Figure 5.5 shows a NodeMap screenshot when the searching document option is activated. Each time a node is expanded, a new search is triggered. Each query result shows the following information: *number of search matches*, *document URL*, *related terms for each document*, *voting*, and *server processing time*.

The related terms associated with a document can be many, filtering may be applied to constrain the number of them. Due to the limited space available to display information and to keep things simple, a *more* option was implemented by me, so that in order to access the full list of related terms the *more* link should be followed. Also it is important to notice that each related term can produce a new search when clicked.

Document search uses Servlet technology for retrieving documents, whereas WS-Talk libraries (interpreter, locator, stemmer, and others) are used for text matching. A complete discussion regarding the method used for matching documents is given by Mothe et al. [57].

A client request is handled by means of calling a URL (HTTP) address where the Servlet is located, and passing the parameters: *query*, *language*, *Servlet host*, *Servlet port*, and *Servlet name*.

Thus the HTTP request takes the following form:

```
http://host:port/servletName/search?query=query&language=language&host=host
&port=port&servlet=servletName
```

As an example a request call to **thames** server at RHUL, querying for the word *mujer*, in *spanish* would look like the following URL:

```
http://thames.cs.rhul.ac.uk:8080/nodemap/search?query=mujer&language=spanish
&host=thames.cs.rhul.ac.uk&port=8080&servlet=nodemap
```

The request response is sent compressed with GZip, and since most modern browsers support decompression on the fly, the above address could be tried directly in a browser. That returns a list of matches for the *mujer* query.

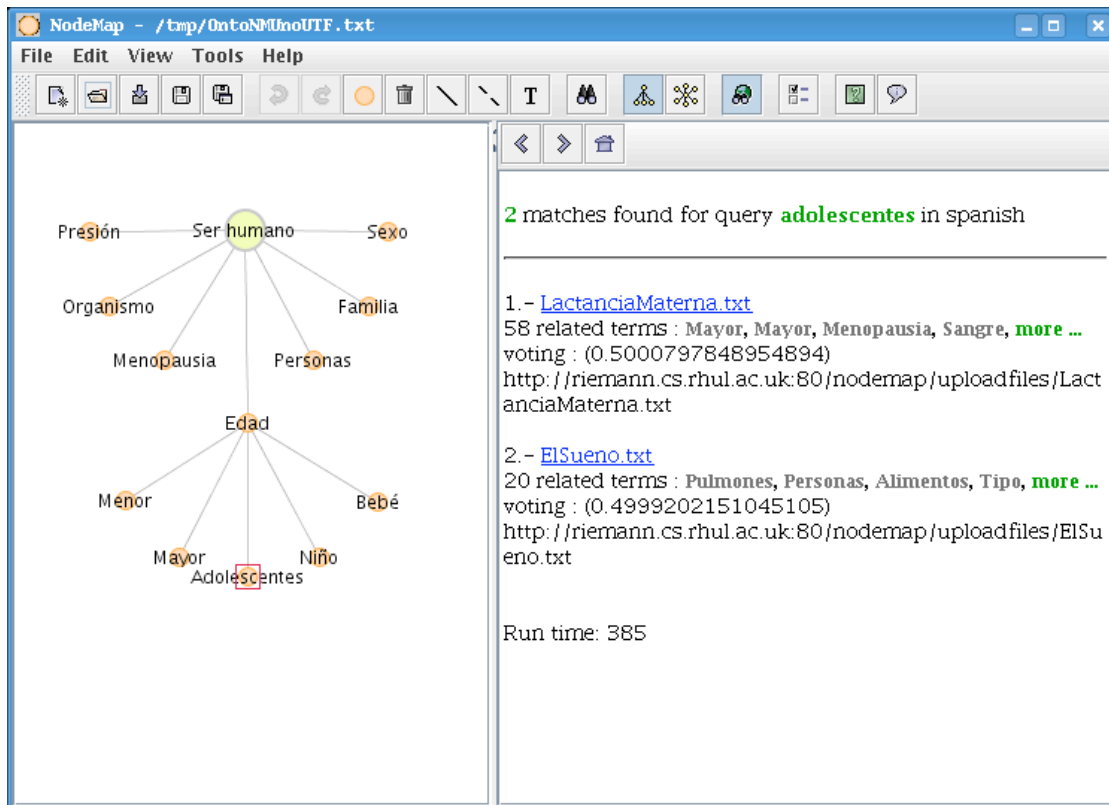


Figure 5.5: NodeMap seek documents

5.5.2 Looking into the database

The following sections provide a description of the tables used in the database to store data and indexes.

Figure 5.6 shows a partial representation of the database schema [57, 13]. White tables are “static”, which means they are created once during the system initialisation. If any of these tables are deleted, the concept hierarchies repository loses its data integrity. Within this category the following tables can be found:

- **URL:** This contains each document location as a URL address.
- **Documents:** This stores document annotations, or document metadata.
- **LinkCH.CHTable:** This stores concept hierarchy data, that is, the names of CH tables generated automatically by the system, their names, and language.
- **CHStemmed:** This stores all the CH stemmed entries.

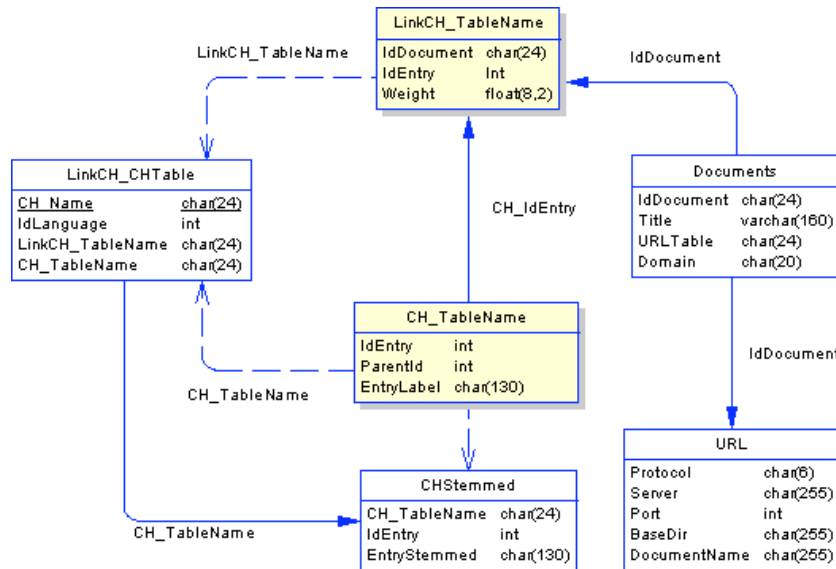


Figure 5.6: Concept hierarchy database schema

Grey-coloured (yellow if colour is available) tables are dynamically created when a new Concept Hierarchy is loaded in the database. For each new Concept Hierarchy, a new

CH_(TableName) and LinkCH_(Table Name) are created. Here “TableName” is the composition of the CH assigned name, and its language. For example when a new CH is loaded in the database with name “STAR” in the English language, the tables created will be called CH_STAR_EN and LINKCH_STAR_EN respectively. There are as many CH_(TableName) and LinkCH_(Table Name) tables as concept hierarchies.

Querying the database

Querying the database is an essential step to retrieve documents. When data is stored in the database, information can be retrieved using SQL language. This allows flexibility to format the data output in such a way that will suit different processes to be applied in order to match searched terms. Thus a key issue is what matching process is used for retrieving information.

Some experiments were carried out in order to test some retrieval methodologies. The following presents the sequential steps used to build a demonstrator that uses hierarchical clustering from the correspondence analysis factor projections [50].

1. The textual input is Porter-stemmed [56].
2. A contingency table is built up of QA (question-answer) texts crossed by all terms resulting from the Porter stemming. A selection of terms is made, based on the middle or discriminating region of the Zipf rank/frequency curve [92].
3. Term profiles are considered, endowed with the chi squared metric, which takes account of relative weighting. (Term profile vectors are doubled in the sense of the doubling operation used in correspondence analysis: this results in the terms being equally weighted in the analysis to follow). Because a chi squared metric is not a natural one for display, mapping it into a Euclidean space is needed. This Euclidean space is the correspondence analysis factor space. Its dimensionality is the smaller of the numbers of terms considered and the numbers of QA texts.
4. The factor projections of terms are used for the hierarchical clustering of terms. (For correspondence analysis, and hierarchical clustering, and software see [49]).

5. A partition of terms is derived from the hierarchical clustering. Clusters of too great a size (viz., number of terms) are excluded; and singleton clusters are of no benefit to us. Remaining clusters will be used for query expansion.
6. When the user submits a query, query expansion implies that closely related terms will be added to the user's request. All then will be used to rank QA text hits, before these are returned to the user.

An implementation of the above method can be found for the English and Spanish languages in the following address: http://thames.cs.rhul.ac.uk:8080/wstalk_en/TextQuery.

The WS-Talk consortium uses the voting method [33, 48, 57] to match documents, which is the one used by the NodeMap demonstrator presented in this document. But since NodeMap communicates with a Servlet that processes the information sent when a node is clicked, the introduction of the matching method presented above is a question of changing the pointers in the program, therefore a trivial thing to do. Thus, it can be said that NodeMap supports both matching method, the correspondence analysis factor projections and the voting method.

Committing new CH to the database

Committing a new CH to the database can be done directly with NodeMap software, without the need of third party programs. Figure 5.7 shows the dialogue window that opens with the option Menu → File → Commit to DB. This will require CH name, database user, database password, and CH language. Also this assumes that the database configuration has been completed as explained in section 5.6

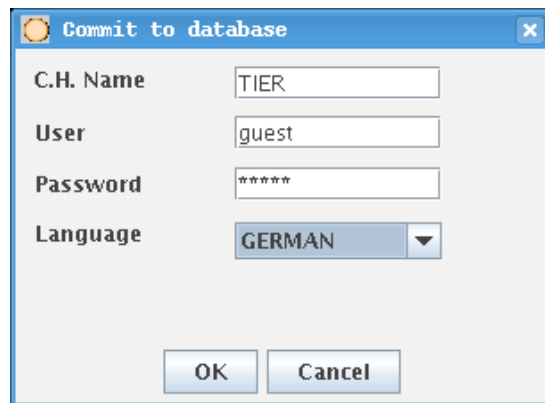


Figure 5.7: Commit to the database

5.6 Servlet and database client configuration

Configuration data is stored locally in the “data” folder which is inside the NodeMap installation directory. The configuration file is called “config.dat” and the parameters are as follows:

1. *Servlet host* e.g. thames.cs.rhul.ac.uk or IP address
2. *Servlet port* e.g. 8080, the default for Apache TomCat
3. *Servlet name* e.g. nodemap which is assigned when deploying the WAR file
4. *Servlet language* e.g. Spanish, English, German, etc. (only one)
5. *DB drivers* e.g. jdbc:mysql (in MySQL case)
6. *DB host* e.g. thames.cs.rhul.ac.uk or IP address
7. *DB port* e.g. 3306 which is default for MySQL
8. *DB name* e.g. wstalk which contains the indexes for a set of documents

It is important to keep the above order, since currently the parameters are read in that sequence from the “config.dat” file. In any case this file is changed automatically by NodeMap as shown in Figure 5.8 and Figure 5.9. Thus no direct file handling is needed. Therefore it is suggested that this file must not be modified.

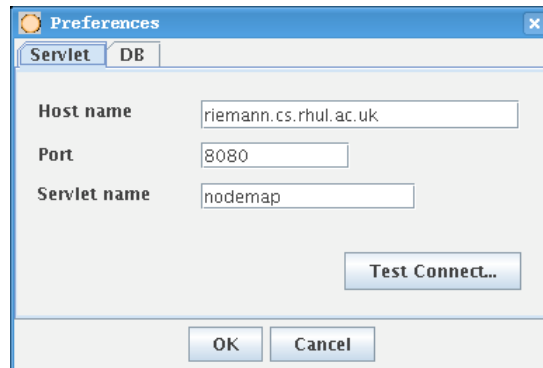


Figure 5.8: Servlet configuration

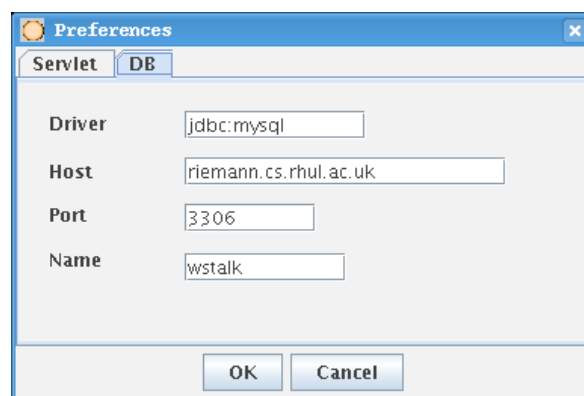


Figure 5.9: Database configuration

5.7 Exploring NodeMap's Java code

Before understanding the Java code it is necessary to highlight which are the components of the software presented in this document. We again note that this software was designed, developed and deployed by the author of this dissertation and Dimitios Zervas. Currently the software is composed as follows:

Java Swing client: It uses Java Swing technology to create the GUI.

Java applet client: It uses Java applet technology to create a client that runs over a browser.

Servlet component: It uses Java Servlet specification to display and dynamically create web pages. It is deployed as a WAR file

Database: It is used for indexing and to store information.

The three first components are in different Java packages. The database needs a valid user name with grants to retrieve and commit information into the database.

5.7.1 Client/Server communication on NodeMap

NodeMap is composed of a client and a server side. The client can perform some tasks independently of the server, which has two components: a Servlet that allows document search; and the database where the indexes are kept. Usually the Servlet is deployed in Apache TomCat (or any another Servlet compliance container) running on the port 8080. Currently MySQL [51] is used running on the port 3306.

The NodeMap client can be used without any interaction with its server side, but depending on the tasks to be carried out communication with the server may be necessary. For instance when designing a new ontology from scratch, importing relevant terms from a file, opening a working space, or saving changes to a file, no communication with the server is needed.

On the other hand the following includes a list of the features that needs interaction with the server:

- **Searching document:** The client requests to the server a matching from a keyword.
- **Commit CH into the DB:** The client requests inserting a new CH into the database.
- **Import CH from DB:** The client requests information from the database.
- **Preference configuration:** The client requests connection verification to the server.

Of course the model presented above can change if either the database or the Servlet are running physically in the same machine.

5.7.2 Compression on NodeMap

After the first tests retrieving data with NodeMap we were discouraged to see the response time, which was of the order of 10 to 15 seconds for retrieving a query. Thus we decided that some improvements were needed to speed up the application. The first target was to have a look at the database and the queries, then how the information was being requested, and finally how the information was transmitted. We made changes to all these areas, and some of the most useful were the implementation of compression when data is transmitted, and the SQL query optimisation. Regarding query optimisation MySQL nested query was changed to manage each query individually directly from the Java code and not from the SQL sentence since nested queries in MySQL work recursively for each sub query, slowing down the retrieving process from the database.

The NodeMap client communicates with its server side based on the TCP/IP protocol. When sending/retrieving data the HTTP/1.1 [35] protocol is used (through URL), which allows for clients to optionally request the compression of content from the server. The standard itself specifies two compression methods: “gzip” (the content wrapped in a gzip stream) and “deflate” (the content in a raw, headerless DEFLATE stream). Both are supported by many HTTP client libraries and almost all modern browsers.

Table 5.4 shows compression ratios in bytes for the set of twelve text documents currently used by the NodeMap prototype as demonstrations. It can be observed that the average compression ratio is $\bar{x} = 2.1$, with a standard deviation $\sigma = 0.2$

Table 5.5 shows the compression ratios for a set of twelve queries. Since query answers are HTML documents, it is observed that compression ratios are much higher when compared with documents text only. $\bar{x} = 7.3$ and $\sigma = 0.9$

GNU zip [26] (gzip) compression is a natural choice as compression method. It is well known, open source, built-in on most modern browsers, and fast. Should more information be needed regarding compression ratios and compression times please refer to Collin [11] and HDF5 [31].

File name	Non-Compressed	Compressed	Compression ratio
AlimentosQueCuran.txt	4085	1946	2.10
BotiquinDePrimerosAuxilios.txt	2485	1408	1.76
ConsejosParaUnaPielSaludable.txt	5500	2594	2.12
DeLaPiletaACasa.txt	2707	1397	1.94
DirectoAlCorazon.txt	3626	1661	2.18
DolorDePechoYAtaqueCardiaco.txt	2655	1447	1.83
ElAcidoFolico.txt	4238	1781	2.38
ElSueno.txt	5377	2390	2.25
InfeccionRespiratoriaInfantil.txt	2807	1428	1.97
LactanciaMaterna.txt	10624	4705	2.26
LaImportanciaDelCalcio.txt	5079	2213	2.30
PrevencionDeCancerDeMama.txt	3172	1515	2.09

Table 5.4: Compression ratios for text files

Query	Non-Compressed	Compressed	Compression ratio
agua	4947	844	5.86
causa	9328	1161	8.03
fibras	6057	888	6.82
mujer	6110	903	6.78
hombre	6127	904	6.78
riesgos	9013	1102	6.78
mayor	11861	1281	9.26
menor	7619	993	7.67
meses	6065	922	6.58
piel	6107	874	6.99
sangre	7554	988	7.65
tipo	7919	1079	7.34

Table 5.5: Compression ratios for query (HTML) files

Table 5.6 shows the Java Servlet method used to compress data when a request is produced. This method is called every time a response is made from a NodeMap client. Therefore all the communication between the client and the server side is compressed. Compression using Servlet technology can be achieved by many methods such as Servlet filter or direct compression. Experiments were carried out with this two methodologies, but due to simplicity direct compression was selected. This is presented in Table 5.6.


```

private PrintWriter getPrintWriter(HttpServletRequest request ,
                                     HttpServletResponse response)

    throws IOException {

    PrintWriter out = null;
    response.setHeader(‘‘Content-Encoding’’, ‘‘gzip’’);

    //compress query answer on UTF-8 format
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
    new GZIPOutputStream(response.getOutputStream()), ‘‘UTF-8’’));

    out = new PrintWriter(bw);

    return out;
}

```

Table 5.6: Data compression

Once a request to the server is made, the URL address containing the stream of data is produced. As seen above, this data stream is compressed using the gzip algorithm, and to make it readable by the user (on the client side) it needs decompression.

Table 5.7 shows the current Java code for decompressing an URL which contains the request response delivered by the server.

```

public static String decompress(URL url) {
    StringBuffer sb = new StringBuffer();

    //this decompress an URL's content
    try {
        GZIPInputStream gzip = new GZIPInputStream(url.openStream());

        BufferedReader zipReader = null;

        zipReader = new BufferedReader(
            new InputStreamReader(gzip, 'UTF-8'));
        char chars[] = new char[1024];
        int len = 0;

        //Write chunks of characters to the StringBuffer
        while ( (len = zipReader.read(chars, 0, chars.length)) >= 0) {
            sb.append(chars, 0, len);
        }
        chars = null;
        gzip.close();
        zipReader.close();
    }
    catch (IOException ex) {
    }
    return (sb.toString());
}

```

Table 5.7: Data decompression

5.7.3 Understanding the graphics on NodeMap

This subsection describes the NodeMap design and how the system can be used in practice.

NodeMap displays hierarchies graphically as rooted trees. Trees are constructed by interconnected nodes.

Nodes can have only one parent node but there is not a restriction on the number of child nodes they can have. The nodes are instances of the Node class and the TreeDataBase is the class which holds the whole tree:

The Node class

Attributes

- **Parent** is of type *Node* and is a pointer to the parent node.
- **Children** is of type vector and stores the child nodes.
- **X, Y** are of type float and used to store the node's current x, y position.
- **dirX, dirY** are of type float and used to store the direction a node has when attached to its parent node. When a user moves the node around using the mouse then the direction is used to move it to its initial position and create a rubber band effect.
- **armLength** is the distance a node has to its parent. It has the default value of 100 but it increases if the node is expanded by multiplying the EXPANDED_OFFSET constant and armLength.
- **Image** is used to store the image that represents the node. Currently two different images are used: one for the root node and one the rest of the nodes.
- **width, height** are the dimensions of the node and they are calculated using the image size. They are used for selecting and moving the node with the mouse.

- **Radius** of the node calculated using the width and height values. This is used together with x, y as a boundary sphere.
- **spanAngle** is the angle the node has with its parent.
- **entryLabel** is a string describing the node.
- **idEntry** is a unique id assigned to each node.
- **parentId** is the node's parent idEntry. It is used to construct the tree hierarchy.

Constants

- **SPEED**: It is the speed with which a node returns to its initial position after dislocation with the mouse.
- **EXPANDED_OFFSET**: It is the offset distance for expanded nodes.

Constructors

- `public Node(int idEntry, int parentId, String entryLabel)`
- `public Node(int idEntry, int parentId, String entryLabel ImageIcon imageIcon)`

Access methods

Access methods are used to access and set private members of the class.

- `public Node getParrent()`
- `public String getEntryLabel()`
- `public int getWidth()`
- `public int getHalfWidth()`
- `public int getHeight()`
- `public int getHalfHeight`
- `public int getIdEntry()`

- public Rectangle getRectangle
- public int getParentId()
- public void setParentId(int id)
- public void setParent(Node node)
- public float getRadius
- public int getNumChildren()
- public float getArmLength()
- public boolean isExpanded()
- public void setExpanded()
- public void setPosition(Point p)
- public Point getPosition()
- public Node getChild(int i)
- public int numChildren()

Other methods

- public void setShrink(): Shrinks an expanded node.
- public void animate (): Animates a node on the screen when the user moves it. The node returns to its initial position creating a rubber band effect.
- public void recalculateAngles(): Recalculates the positions and angles of the child nodes.
- public void setExpanded(): Expands the node (makes its children visible) for normal tree viewing.
- public void setExpandedLevel(): Expands the node (makes its children visible) for level viewing.

- `public void addChild(int idEntry, int parentId, String entryLabel, ImageIcon imageIcon)`: Adds a new child to the node.
- `public void addChild(Node child)`: Adds an existing child to the node.
- `public boolean contains(Point p)`: Checks if point x, y is contained within the boundary rectangle of a node.
- `public void deleteChild(Node node)`: Removes a child node using its instance.
- `public void deleteChild(int i)`: Removes a child using its index in the children list.
- `public void checkForCollisions(Node node)`: Checks for collisions using the boundary sphere of the node. This method currently is deprecated.
- `public void drawOnlyThis(Graphics2D g)`: Draws a node but not its children.
- `public void draw(Graphics2D g)`: Draws a node and its children.
- `public void drawLevel(Graphics2D g)`: Draws a node and its children in level format (not the whole tree).
- `public void animateChildren()`: Animates the children of the node.

The **TreeDataBase** class

Attributes

- **initMousePos** stores the mouse position. It is used for the movement of the whole tree using the mouse.
- **treePos** holds the x, y coordinates of the whole tree.
- **maxBounds** stores the maximum x, y of the tree and is used for setting the scrollbars.
- **Root** is of Node type and as its name implies is the root node for the whole tree.
- **Unconnected** is of type vector and stores the nodes not connected to the tree.

- **maxId** stores the maximum idEntry. It is used to assign larger idEntry numbers to new nodes.
- **unconnectedY** holds the Y offset of the newly added unconnected nodes.

Constants

- **TMP_FILE**, **TMP_FILE2** strings for filename used for temporary files which are created when the tree is modified. The files store information for the tree so the user can undo changes to the tree.

Constructors

- `public TreeDataBase()`
- `public TreeDataBase(int x, int y)`

Access methods

These are used to access and set private members of the class

- `public void setInitUnconnectedY(int y)`
- `public int getMaxId()`
- `public void setPosition(Point p)`
- `public void setPosition(int x, int y)`
- `Node getRoot()`
- `void setRoot(Node node)`
- `public void setMousePos(Point p)`

Other methods

- `public void addUnconnected(int idEntry, String entryLabel)`: Creates and adds a new node which is not connected to the tree using a default position (left upper corner of the screen).

- `public Node addNewUnconnected(String entryLabel, int x, int y)`: Creates and adds a new node which is not connected to the tree using the provided x, y coordinates.
- `void clear()`: Clears the whole tree database and the unconnected node list.
- `void clearUnconnected()`: Clears the the unconnected node list.
- `public void addEntry(int idEntry, int parentId, String entryLabel)`: Adds a new node in the tree.
- `private void addEntry(Node node, int idEntry, int parentId, String entryLabel)`: Adds an existing node to the tree.
- `private Node selectNode(Node node, Point p)`: Used by `selectNode(Point)`; see below.
- `public Node selectNode(Point p)`: Providing an x, y coordinate, a node is selected if the coordinate is on a node.
- `public Node selectNodeLevel(Node node, Point p)`: Providing an x, y coordinate, a node is selected if the coordinate is on a node. This is used when the tree is presented in level format.
- `public void drawUnconnected(Graphics2D g)`: Draws the unconnected nodes.
- `private void animate(Node node)`: Used by `animate()`; see below.
- `public void animate()`: Animates the whole tree.
- `private void moveAll(Node node, Point p)`: Used by `moveAll()`; see below.
- `public void moveAll(Point p)`: Moves the whole tree depending the mouse's movement.
- `private void checkForCollisions(Node node)` (Currently deprecated): Used by `checkForCollisions()`; see below.
- `public void checkForCollisions()` (Currently deprecated): Checks for collisions between nodes in the tree.

- `public void setAnimation(boolean val)`: Currently deprecated
- `private void getMaxBounds(Node node)`: Used by `getMaxBounds()`; see below.
- `public Dimension getMaxBounds()`: Returns the maximum bounds of the tree depending on its expansion.
- `public void deleteNode(Node delNode)`: Removes the selected node from the tree.
- `public void disconnectNode(Node selNode)`: Disconnects the selected node from the tree.
- `public void removeFromUnconnected(Node selNode)`: Removes a node from the unconnected list.
- `private void writeToFile(PrintStream os, Node node)`: Used `writeToFile(String filename)` see below.
- `public void writeToFile(String filename)`: Saves the tree to a UTF-8 text file. The text file has the following format: `idEntry, parentId, entryLabel`. The attributes are separated with tab and each entry with a new line.
- `private void AddToBT(BinaryTree bt, Node node)`: Used by the `writeToFileSoftenedByIdEntry(String filename)` function to sort the entries by their id using a binary tree, see below.
- `public void writeToFileSoftenedByIdEntry(String filename)`: Saves the tree to a UTF-8 text file sorted by their `idEntry`. The text file has the following format: `idEntry, parentId, entryLabel`. The attributes are separated with tab and each entry with a new line.
- `public void writeToDB(String url, String user, String password, String chName, String language)`: Saves the tree to a database.
- `public void saveWorkSpace(String filename)` throws `Exception`: Saves both the tree and the unconnected list to a UTF-8 text file.
- `public void loadWorkSpace(String filename)` throws `Exception`: Loads a UTF-8 text file created by `saveWorkSpace(String filename)`.

- `private Node search(Node node, String text)`: Used by `search(String text)`; see below.
- `public Node search(String text)`: Searches for a node within the tree using `entryLabel`. If the node is found, it is returned otherwise `Null` is returned.
- `void expandParent(Node node)`: Expands (makes visible) the parent node of a node.
- `private void expandTree(Node node)`: Used by `expandTree()`; see below.
- `public void expandTree()`: Expands the whole tree.
- `public void createTempFile()`: Saves both the tree and the unconnected list to a temporary UTF-8 text file. It is used for the Undo and Redo options.
- `public void loadTempFile()`: Loads a UTF-8 text file created by `createTempFile()`. It is used for the Undo and Redo options.

5.8 Summary

This chapter presents NodeMap ontology designer within the WS-Talk framework. This software was developed using a range of open source tools to allow inter-organisational co-operation not just limited to the code development process, but extending to sharing of knowledge. It is important to highlight that ontologies are not just used in the context of WS-Talk, but in a wide range of applications, such as the field of astronomy. Thus, the software presented here can be adapted to retrieve information in the astronomical field, using astronomical ontologies.

Chapter 6

Future work and conclusion

6.1 Current and future work

Within the NodeMap context some of the areas that currently are being developed are the following:

- **Servlet filters:** Filters are important when working with Servlets. A filter dynamically intercepts Servlet requests or responses to manipulate them. In the NodeMap case, filters can help implementing *cache* files, in order to speed up the information retrieval process. For instance, when a request (or query) is sent to the server, the server could check if similar requests have been made in the past, and send the file associated to the query. If no similar request exists the database will be queried and a new cache file is produced. Also a mechanism can be implemented to deal with old cache files, so the systems do not fall out of date.
- **Motion algorithms research:** The motion algorithms currently implemented in the software still have space for improvement. The methods related with node overlap and space optimisation in the visual area can be improved.
- **WS-Talk database and XML support:** Currently we are discussing in conjunction with Lucky Eye¹ and IRT² a new standard to be used on the database and XML format. Once this has been formalised, database and XML support for the new standards in NodeMap will be introduced.

¹WS-Talk partner in Turkey

²WS-Talk partner in France

- **Association rules:** Currently we are discussing in conjunction with UDP/Soluciones³ the implementation of association rules to extract related terms. Some progress has already been made but the current code still needs improvement.
- **Search configuration:** Currently WS-Talk offers a set of different methods for document (or web service) searching. Introducing a configuration module in NodeMap will give flexibility to the user at the moment of deciding which search methods will suit his/her needs best.
- **Customisation:** If needed, customisation can be added to the software in order to personalise its “feel and look”, e.g. change node colours and size, or the thickness of the link. Additionally, translation to at least one another language (Spanish) will be implemented.

³WS-Talk partner in Chile

6.2 Conclusion

Tools that group independent packages (e.g. wiki, news, forum, calendar, etc.) addressing the problem of how to organise distributed work and knowledge have started to emerge in the open source community. That is the case of eGroupWare <http://www.egroupware.org/> just to mention one. Commercial tools have an advantage in this case, specially the ones oriented to the corporate market (e.g. Lotus www.lotus.com from IBM). But open source (OS) software is catching up quickly in this area. This OS tools are the special interest of the author of this dissertation, since in inter-academic research projects licencing and price can be an issue.

Despite the fact that open collaborative tools can be considered as being in their infancy when working in a co-ordinated way, it is not less true that packages grouping a wide range of these tools will appear – as mentioned in the case of the eGroupWare –, focusing not just in the planning and knowledge sharing but in the way that this share-knowledge affects the software being developed. Following this idea it can be foreseen that packages integrating software development tools such as Maven, Ant, CruiseControl, and CVS – to name a few –, will be integrated with tools such as wiki, news, forum and other project management tools. This can of tools can hugely facilitate the work between virtual organisations.

Finally, if more information is needed regarding how the NodeMap prototype is used for building concept hierarchies, the RHUL web site for this project can be accessed at <http://thames.cs.rhul.ac.uk/wstalk> where an installer for Windows, Mac OS, Linux and other operating systems can be found; also an example file is presented there (Appendix B).

Publications related to this thesis

P. Contreras and F. Murtagh, “AstroGrid as an e-Learning Environment”, 1st International Kaleidoscope Learning Grid SIG Workshop on Distributed e-Learning Environments (Nicola Capuano, Pierluigi Ritrovato, and Fionn Murtagh, eds.), Italy. 2005, <http://www.bcs.org/server.php?show=ConWebDoc.3871>.

S. Johnstone, P. Contreras, F. Murtagh, P. Sage, “Peer-to-Peer Information Access and Retrieval”, Ingénierie des systèmes d’information. Volume 10 – no 1, France. pp 101 - 122, 2005.

P. Contreras, S. Johnstone, F. Murtagh and K. Englmeier, “Distributed multimedia content with P2P JXTA technology”, Inclusive Design in the Information Society, Volume 4 of universal access in HCI 2003 International Conference on Human Computer Interaction, Eds. C. Stephanidis, Crete. June 2003, Lawrence Erlbaum, Mahwah, NJ, pp. 690-694, 2003.

F. Murtagh, T. Taskaya, P. Contreras, J. Mothe and K. Englmeier, “Interactive visual user interfaces: a survey”, Artificial Intelligence Review, 19, 263-283, 2003.

P. Contreras, M. Köküer, M. Louys and F. Murtagh, “Semantic description of signal and image databases”, in: Opto-Ireland 2002: Optical Metrology, Imaging, and Machine Vision. Eds. A. Shearer, F.D. Murtagh, J. Mahon, and P.F. Whelan. Proceedings of the SPIE, Volume 4877, pp. 230-237, 2003.

P. Contreras, S. Johnstone, and F. Murtagh, “Peer to Peer based retrieval of distributed image stores: demonstrator using JXTA”, Proceedings of the iAstro/IDHA Workshop Strasbourg Observatory, 28-29, 2002.

F. Tao, P. Contreras, B. Pauer, T. Taskaya and F. Murtagh, “User interest correlation in log data”, Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality, Volume 1 of HCI 2001 International Conference on Human Computer Interaction, Eds. M.J. Smith, G. Salvendy, D. Harris, and R.J. Koubek, New Orleans, LA, August 2001, Lawrence Erlbaum, Mahwah, NJ, pp. 938-942, 2001.

Related reports

P. Contreras, D. Zervas, F. Murtagh, J. Pereira, R. González, and K. Englmeier, “WS-Talk Interpreter, Repository Specification, and Database Handling”. Deliverable D3.2/D3.3, September 2005, <http://thames.cs.rhul.ac.uk/wstalk/papers/rhulpapers/DeliverableUDP-RHUL.pdf>. CS, RHUL.

P. Contreras, F. Murtagh. “SOA and WS-TALK Integrating Web Services in WS-TALK with BPEL and UDDI”. July 2006, <http://thames.cs.rhul.ac.uk/wstalk/papers/rhulpapers/RHULdeliverableJuly2006.pdf>. CS, RHUL.

P. Contreras, F. Murtagh, D. Zervas. “Distributed Development Process: AstroGrid Experience for WS-Talk”. April 2005, <http://thames.cs.rhul.ac.uk/wstalk/papers/rhulpapers/wsTalkatRHUL.pdf>. CS, RHUL.

Acronyms and web references

Astronomical on-line data archives

2MASS: Two-micron All-Sky Survey - Infra-red (Caltech). <http://www.ipac.caltech.edu/2mass>

AAT: Anglo-Australian Telescope - archive. <http://www.aao.gov.au/archive>

ADC: Astronomical Data Center (GSFC). <http://adc.gsfc.nasa.gov>

AEQ: ADC External Query engine (GSFC). <http://tarantella.gsfc.nasa.gov/viewer/AEQdoc.html>

Aladin: Image Archive and Browser (Strasbourg). <http://aladin.u-strasbg.fr>

AMASE: Astrophysics Multi-spectral Archive Search Engine (GSFC). <http://amase.gsfc.nasa.gov/amase/WelcomeToAMASE.html>

APM cat: Astronomical Plate Measuring engine catalogue site (Cambridge). <http://www.ast.cam.ac.uk/~apmcat>

APS: Automated Plate Scanner (University of Minnesota). <http://aps.umn.edu>

ASCA: ASCA Science Archive (ISAS, Japan). <http://www.astro.isas.ac.jp/asca/index-e.html>

Astrobrowse: Astrobrowse query engine (GSFC). <http://heasarc.gsfc.nasa.gov/ab>

CADC: Canadian Astronomy Data Centre (Victoria, BC). <http://cadwww.dao.nrc.ca>

CDS: Centre de Données de Strasbourg. <http://cdsweb.u-strasbg.frCDS>

CEUV: Center for Extreme Ultraviolet Astrophysics (Berkeley). <http://www.cea.berkeley.edu/~science/html/Archive.html>CEUV

Chandra: Chandra X-ray Observatory Center (Harvard). <http://chandra.harvard.eduChandra>

Chandra-UK: Chandra data archives - UK mirror (Leicester). <http://ledas-cxc.star.le.ac.uk>Chandra-UK

DAO: Dominion Astrophysical Observatory (Victoria BC). <http://www.hia.nrc.ca/facilities/dao>DAO

DPOSS: Palomar Digital Sky Survey (Caltech). <http://www.astro.caltech.edu/~george/dposs/DPOSS>

EGSO: European Grid of Solar Observatories. <http://www.mssl.ucl.ac.uk/grid/egso>EGSO

ESO-archive: European Southern Observatory archive (Garching).<http://arch-http.hq.eso.org>ESO-archive

FIRST: Faint Images of Radio Sky at Twenty-centimeters (VLA) <http://sundog.stsci.edu>FIRST

GSC-II: Guide Star Catalog-II (STScI). <http://opposite.stsci.edu/pubinfo/pr/2001/18/GSC-II>

HEASARC: High Energy Astronomy Science Archive Research Center (GSFC). <http://heasarc.gsfc.nasa.gov>HEASARC

HERA: HEASARC data analysis tool (GSFC). <http://heasarc.gsfc.nasa.gov/hera>HERA

IMPRESS: IMage PeRimeters of Sky Surveys - graphical database interface (GSFC). <http://tarantella.gsfc.nasa.gov/impress/IMPRESS>

INES: IAU Final Archive data (Strasbourg). <http://godot.u-strasbg.fr/ines/INES>

INT: Isaac Newton Telescope - wide field survey. <http://www.ast.cam.ac.uk/~wfcsur/>INT

IoA: Institute of Astronomy (Cambridge). <http://www.ast.cam.ac.uk>IoA

IoA-archive: Institute of Astronomy (Cambridge) archive. <http://archive.ast.cam.ac.uk>IoA-archive

IPAC: Infrared Processing and Analysis Center (Caltech). <http://www.ipac.caltech.edu>IPAC

IRSA: InfraRed Science Archive (Caltech). <http://irsa.ipac.caltech.edu>IRSA

ISO: Infrared Space Observatory (Vilspa). <http://www.iso.vilspa.esa.es>ISO

IUE: International Ultraviolet Explorer archive (Vilspa) <http://www.vilspa.esa.int/iue/iue.html>IUE

jMAISON: Multi-wavelength Astronomical Image Service ON-line <http://maison.isas.ac.jp>jMAISON

LEDA: Lyon-Meudon Extragalactic Database. <http://leda.univ-lyon1.fr>LEDA

LEDAS: Leicester Data Archive Service. <http://ledas-www.star.le.ac.uk>LEDAS

MAST: Multimission Archive at STScI. <http://archive.stsci.edu/mast.html>MAST

NED: NASA/IPAC Extragalactic Database (Caltech). <http://nedwww.ipac.caltech.edu>NED

NSSDC: National Space Science Data Center (GSFC). <http://nssdc.gsfc.nasa.gov>NSSDC

NVSS: NRAO VLA Sky Survey. <http://www.cv.nrao.edu/~jcondon/nvss.html>NVSS

PDS: Planetary Data System (JPL). <http://pds.jpl.nasa.gov>PDS

Querator: Advanced Multi-Archive Search Engine. <http://archive.eso.org/querator>Querator

ROSAT: Röntgensatellit X-ray Data Centre (Garching). <http://wave.xray.mpe.mpg.de>ROSAT

SAO: Smithsonian Astrophysical Observatory (Harvard).// <http://hea-www.harvard.edu>SAO

SDAC: Solar Data Analysis Center (GSFC). <http://umbra.nascom.nasa.gov/sdac.html>SDAC

SDSS: Sloan Digital Sky Survey. <http://www.sdss.org>SDSS

SIRTF: Space Infrared Telescope Facility. <http://sirtf.caltech.edu>SIRTF

SSDS: Space Science Data System (GSFC). <http://ssds.nasa.gov>SSDS.

Starcast: Starcast interface to STScI archive. <http://archive.stsci.edu/starcast>Starcast.

Starview6: Starview database browser - (STScI). <http://starview.stsci.edu/html>Starview6

STScI: Space Telescope Science Centre - archive (Baltimore). <http://archive.stsci.edu>STScI

SuperCOSMOS: Wide Field Astronomy Unit (Edinburgh)sky survey.<http://www-wfau.roe.ac.uk/sss/SuperCOSMOS>

SX: Sloan Digital Sky Survey Science Archive (Johns Hopkins University). <http://www.sdss.jhu.edu/ScienceArchive/home.html> SX

UKADC: UK Astronomical Data Centre (Cambridge). <http://archive.ast.cam.ac.uk/UKADC>

VizieR: Catalogue archive with browser (Strasbourg). <http://vizier.u-strasbg.fr> VizieR

Bibliographical sites and Documentation

ADS: The Smithsonian/NASA Astrophysics Data System. <http://adswww.harvard.edu/>

Astro-ph: Astrophysics preprint service (Southampton). <http://xxx.soton.ac.uk/archive/astro-ph>

ASDS: Astronomical Software & Documentation Service. <http://asds.stsci.edu/>

Bibcode: Bibliographic reference code. <http://cdsweb.u-strasbg.fr/simbad/refcode.html>

Simbad: Bibliographic database (Strasbourg). <http://simbad.u-strasbg.fr>

Virtual Observatory Projects and Forums

Astro-wise: Astronomical Wide-field Imaging System for Europe. <http://www.astro-wise.org>

Astrovirtel: Accessing Astronomical Archives as Virtual Telescopes. <http://www.euro-vo.org/astrovirtel>

Astrogrid: Astrogrid - UK's Virtual Observatory project. <http://www.astrogrid.ac.uk>

AuVO: Australian Virtual Observatory. <http://aus-vo.org/>

AVO: Astrophysical Virtual Observatory. <http://www.atnf.csiro.au/projects/avo>

China-VO: Virtual Observatory of China. <http://www.china-vo.org>

Digital Sky: Digital Sky Project (Caltech). <http://digital-sky.org>

iAstro: iAstro - Computational and Information Infrastructure in the Astronomical Data-Grid. <http://www.iaastro.org>

IVOA: International Virtual Observatory Alliance. <http://www.ivoa.net/>

NVO: National Virtual Observatory (USA). <http://us-vo.org>

OASIS: On-line Archive Science Information Services (Caltech). <http://irsa.ipac.caltech.edu/applications/Oasis>

Opticon: Optical Infrared Coordination Network. <http://www.astro-opticon.org>

OST: UK Office of Science and Technology. <http://www.dti.gov.uk/ost>

RVO: Russian Virtual Observatory. <http://www.inasan.rssi.ru/eng/rvo>

SEA: Scientists Expert Assistant (GSFC).

Skycat: Skycat data visualiser (Garching). <http://archive.eso.org/skycat>

SkyServer: Sloan Digital Sky Survey server (JHU). <http://www.skyserver.org>

Skyview: Virtual Observatory image viewer (GSFC). <http://skyview.gsfc.nasa.gov>

Urania: Universal Research Archive of Networked Information in Astronomy. <http://www.aas.org/Epubs/webinfo/Urania>

Bibliography

- [1] Aladin, *The Aladin Sky Atlas. Centre de Données astronomiques de Strasbourg*, 2006, <http://aladin.u-strasbg.fr/>.
- [2] Ant, *Apache Ant*, 2005, <http://ant.apache.org/>.
- [3] Apache, *Apache Software Foundation*, 2004, <http://httpd.apache.org/>.
- [4] Arecibo, *Arecibo Observatory*, 2005, <http://www.naic.edu/>.
- [5] AstroGrid, *Astrogrid Portal*, 2005, <http://www.astrogrid.org>.
- [6] AstroScope, *AstroScope Tool*, 2006, <http://www2.astrogrid.org/documentation/ag-help-docs/data-explore/astroscope/?searchterm=AstroScope>.
- [7] Axis, *Apache Axis*, 2005, <http://ws.apache.org/axis/>.
- [8] BugZilla, *Bug-tracking system*, 2005, <http://www.bugzilla.org/>.
- [9] Cactus, *Cactus code*, 2005, <http://www.cactuscode.org/>.
- [10] Chandra, *Chandra X-ray Observatory*, 2006, <http://chandra.harvard.edu/>.
- [11] Lasse Collin, *A Quick Benchmark: Gzip vs. Bzip2 vs. LZMA*, 2005, <http://tukaani.org/lzma/benchmarks>.
- [12] Pedro Contreras, Steven Johnstone, Fionn Murtagh, and Kurt Englmeier, *Distributed multimedia content with P2P JXTA technology*, HCI 2003 International Conference on Human Computer Interaction (Constantine Stephanidis, ed.), Lawrence Erlbaum Associates, Inc, June 2003, pp. 690–694.
- [13] Pedro Contreras, Dimitri Zervas, Fionn Murtagh, Javier Pereira, Rodrigo Gonzalez, and Kurt Englmeier, *WS-Talk Interpreter, Repository Specification, and Database Handling. Deliverable D3.2/D3.3*, 2005, <http://thames.cs.rhul.ac.uk/~wstalk/papers/rhulpapers/DeliverableUDP-RHUL.pdf>.

- [14] CTIP, *Thermosphere ionosphere plasmasphere model*.
- [15] CVS, *Concurrent Versions System*, 2005, <https://www.cvshome.org/>.
- [16] Michael Denny, *Ontology Building: A Survey of Editing Tools*, 2006, <http://www.xml.com/pub/a/2002/11/06/ontologies.html>.
- [17] CMS Directory, *Content Management Directory*, 2005, <http://content-management-directory.com/>.
- [18] Eclipse, *Eclipse Foundation*, 2004, <http://www.eclipse.org/>.
- [19] Gordon Richards et al, *Spectroscopic target selection in the sloan digital sky survey: The quasar sample*, The Astronomical Journal, vol. 123, The American Astronomical Society, June 2002, pp. 2945–2975.
- [20] Robert E. Williams et al, *The Hubble Deep Field: Observations, data reduction, and galaxy photometry*, Astronomical Journal **112** (1996), 1335.
- [21] FireFox, *Open Source Browser Mozilla FireFox*, 2006, <http://www.mozilla.com/>.
- [22] FITS, *Flexible Image Transport System*, 2005, <http://fits.gsfc.nasa.gov/>.
- [23] Grid Forum, *Global Grid Forum*, 2005, <http://www.gridforum.org>.
- [24] Roger Freedman and William Kaufmann, *Universe*, sixth ed., W. H. Freeman, 2001.
- [25] GAIA, *Graphical Astronomy and Image Analysis Tool*, 2006, <http://star-www.dur.ac.uk/~pdraper/gaia/gaia.html>.
- [26] Jean-loup Gailly and Mark Adler, *GNU zip*, 2006, <http://www.gzip.org/>.
- [27] GALEX, *The Galaxy Evolution Explorer*, 2006, <http://www.galex.caltech.edu/index.html>.
- [28] Alex Garrett, *JUnit antipatterns*, 2005, <http://www-128.ibm.com/developerworks/opensource/library/os-junit/?ca=dgr-lnxw07JUnite>.
- [29] Globus, *Globus Alliance*, 2006, <http://www.globus.org/>.
- [30] Robert Hanisch and Peter Quinn, *The International Virtual Observatory*, <http://www.ivoa.net/pub/info/TheIVOA.pdf>.
- [31] HDF5, *Compression Performance Evaluation Report*, 2006, http://hdf.ncsa.uiuc.edu/HDF5/papers/papers/bzip2/bzip2_reporttps.pdf.

- [32] HelioScope, *HelioScope Tool*, 2006, <http://www2.astrogrid.org/documentation/ag-help-docs/data-explore/helioscope>.
- [33] Nathalie Hernandez and Josiane Mothe, *Ontologies pour l'aide à l'exploration d'une collection de documents*, Veille Stratégique Scientifique & Technologique, October 2004, pp. 405–416.
- [34] HST, *Hubble Space Telescope Archive*, 2005, <http://archive.eso.org/archive/hst/>.
- [35] HTTP, *Hypertext Transfer Protocol - HTTP/1.1*, 1999, <http://www.w3.org/Protocols/rfc2616/rfc2616.html>.
- [36] Hyperz, *Photometric redshift*, 2005, <http://webast.ast.obs-mip.fr/hyperz/>.
- [37] IVOA, *International Virtual Observatory Alliance*, 2005, <http://www.ivoa.net/>.
- [38] JBuilder, *Borland JBuilder*, 2005, <http://www.borland.com/jbuilder/>.
- [39] Jelly, *Jelly Executable XML*, 2005, <http://jakarta.apache.org/commons/jelly/>.
- [40] Steven Johnstone, Pedro Contreras, Fionn Murtagh, and Paul Sage, *Peer-to-Peer Information Access and Retrieval*, Ingénierie des systèmes d'information **10** (2005), 101–122.
- [41] JUnit, *Java™ Testing Framework*, 2005, <http://www.junit.org/>.
- [42] Paul Lagasse, Lora Goldman, Archie Hobson, and Susan Norton (eds.), *The Columbia Encyclopedia*, sixth ed., Columbia University Press, New York, 2001-05.
- [43] Tony Linde, *Astrogrid Architecture Thesis*, 2003, <http://wiki.astrogrid.org/bin/view/AstrogridArchitectureThesis#Bib3>.
- [44] Tanu Malik, Alex S. Szalay, Tamas Budavari, and Ani R. Thakar, *Skyquery: A web-service approach to federate databases*, 2002, <http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0211023>.
- [45] CMS Matrix, *Content Management Directory Matrix*, 2005, <http://www.cmsmatrix.org/>.
- [46] Maven, *Apache Maven Project*, 2005, <http://maven.apache.org/>.
- [47] MERLIN, *Multi-Element Radio Linked Interferometer Network*, 2005, <http://www.jb.man.ac.uk/e-merlin/>.
- [48] Josiane Mothe, Gilles Hubert, Jérôme Augé, and Kurt Englmeier, *Catégorisation automatique de textes basée sur des hiérarchies de concepts*, Journées Bases de Données Avancées, October 2003, pp. 69–87.

- [49] Fionn Murtagh, *Correspondence Analysis and Data Coding with R and Java*, first ed., Chapman & Hall/CRC, 2005.
- [50] Fionn Murtagh, Dimitri Zervas, and Pedro Contreras, *A text analysis matching demonstrator*, 2006, http://thames.cs.rhul.ac.uk:8080/wstalk_en/TextQuery.
- [51] MySQL, *MySQL Database Manager System*, 2006, <http://www.mysql.org/>.
- [52] NetCraft, *Net Craft*, 2005, http://news.netcraft.com/archives/2005/04/01/april_2005_web_server_survey.html.
- [53] OKBC, *Open Knowledge Base Connectivity*, 2006, <http://www.ai.sri.com/~okbc/>.
- [54] OWL, *Web Ontology Language*, 2006, <http://www.w3.org/2004/OWL/>.
- [55] Palomar, *Palomar Observatory*, 2006, <http://www.astro.caltech.edu/palomar/>.
- [56] Martin Porter, *Porter Stemmer*, 2006, <http://www.tartarus.org/~martin/PorterStemmer>.
- [57] Sandra Poulain and Josiane Mothe, *Release of WS-Talk utilities. Deliverable D2.1*, 2005, <http://www.akra.de/ws-talk/>.
- [58] AstroGrid problems list, *Astrogrid*, 2002, <http://wiki.astrogrid.org/bin/view/V0/ScienceProblemList>.
- [59] Protégé, *Protégé Software*, 2006, <http://protege.stanford.edu/>.
- [60] SExtractor, *Object catalogue for astronomical images*, 2005, http://terapix.iap.fr/rubrique.php?id_rubrique=91/.
- [61] SIAP, *Simple Image Access Protocol*, 2005, <http://www.ivoa.net/Documents/latest/SIA.html>.
- [62] SIMBAD, *Astronomical Database, Centre de Données astronomiques de Strasbourg*, 2005, <http://simbad.u-strasbg.fr/>.
- [63] SOAP, *Simple Object Access Protocol*, 2005, <http://www.w3.org/TR/soap/>.
- [64] SOHO, *Solar & Heliospheric Observatory*, 2005, <http://sohowww.nascom.nasa.gov/>.
- [65] Spitzer, *Spitzer Space Telescope*, 2005, <http://www.spitzer.caltech.edu/>.
- [66] SPLAT, *Spectral Analysis Tool*, 2006, <http://star-www.dur.ac.uk/~pdraper/splat/>.
- [67] SSH, *Secure Shell*, 2005, <http://www.openssh.com/>.
- [68] Subversion, *Subversion Version Control System*, 2006, <http://subversion.tigris.org/>.

- [69] Swift, *Swift Gamma-Ray Burst Explorer Mission*, 2006, <http://www.swift.psu.edu/>.
- [70] Alexander S. Szalay, Tamas Budavari, Tanu Malika, Jim Gray, and Ani Thakara, *Web services for the virtual observatory*, 2002, <http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0208014>.
- [71] AstroGrid ten science problems, *Astrogrid*, 2002, <http://wiki.astrogrid.org/bin/view/Astrogrid/ScienceProblems>.
- [72] thefreecountry.com, *Free Source Code Version Control Management Software*, 2005, <http://www.thefreecountry.com/programming/versioncontrol.shtml>.
- [73] ThinkMap, *ThinkMap Software*, 2006, <http://www.thinkmap.com>.
- [74] Dave Thomas and Andy Hunt, *Pragmatic version control using cvs*, Pragmatic Bookshelf, Sep 2003.
- [75] Tomcat, *Apache Tomcat*, 2006, <http://tomcat.apache.org/>.
- [76] TopCat, *Tool for OPerations on Catalogues And Tables*, 2006, <http://www.star.bristol.ac.uk/~mbt/topcat/>.
- [77] TWiki, *Enterprise Collaboration Platform*, 2005, <http://www.twiki.org/>.
- [78] Allan Vermeulen, *The Elements of Java™ Style*, Cambridge University Press, Cambridge, 2000.
- [79] VisIVO, *Visualisation Interface to the Virtual Observatory*, 2006, <http://visivo.cineca.it/>.
- [80] VISTA, *Visible and Infrared Survey Telescope for Astronomy*, 2005, <http://www.vista.ac.uk/>.
- [81] VLA, *Very Large Array*, 2005, <http://www.vla.nrao.edu/>.
- [82] VOSpec, *A Tool to Handle VO-SSAP compliant Spectra*, 2006, <http://esavo.esa.int/vospecapp>.
- [83] VOTable, *Virtual Observatory Table*, 2005, <http://www.us-vo.org/VOTable/>.
- [84] W3C, *The World Wide Web Consortium*, 2005, <http://www.w3.org/>.
- [85] Nicholas Walton, Andrew Lawrence, and Tony Linde, *Scoping the UK's virtual observatory: Astrogrid's key science drivers*, Astronomical Data Analysis Software and Systems XII (Harry Payne, Robert Jedrzejewski, and Richard Hook, eds.), The Astronomical Society of the Pacific, December 2003, pp. 25–28.

- [86] WFCAM, *Wide Field Infrared Camera For UKIRT*, 2005, <http://www.roe.ac.uk/atc/projects/wfcam/>.
- [87] IVOA WS, *Web services for the virtual observatory*, 2006, <http://voservices.org/>.
- [88] WS-Talk consortium, *Web services communicating in the language of their community*, European project supported under the CRAFT program. COOP-CT-2004-006026.
- [89] WSDL, *Web Service Definition Language*, 2005, <http://www.w3.org/TR/wsdl>.
- [90] X11, *X.Org Foundation*, 2005, <http://www.x.org/>.
- [91] XMM-Newton, *ESA XMM-Newton*, 2006, <http://xmm.vilspa.esa.es/>.
- [92] George Kingsley Zipf, *Zipf Law*, 2006, http://en.wikipedia.org/wiki/Zipf's_law.

Appendix A

NodeMap UML diagrams

NodeMap client UML diagrams

NodeMap client UML diagrams included in this appendix are the following:

- Figure A.1 shows the TreeDataBase Java class
- Figure A.2 shows the NodeMap Java class
- Figure A.3 shows the Node Java class
- Figure A.4 shows the BinaryTree Java class
- Figure A.5 shows the OntoTableModel Java class
- Figure A.6 shows the DrawingPane Java class
- Figure A.7 shows the GZip Java class
- Figure A.8 shows the Config Java class
- Figure A.9 shows the PorterStemmer Java class
- Figure A.10 shows the Util Java class

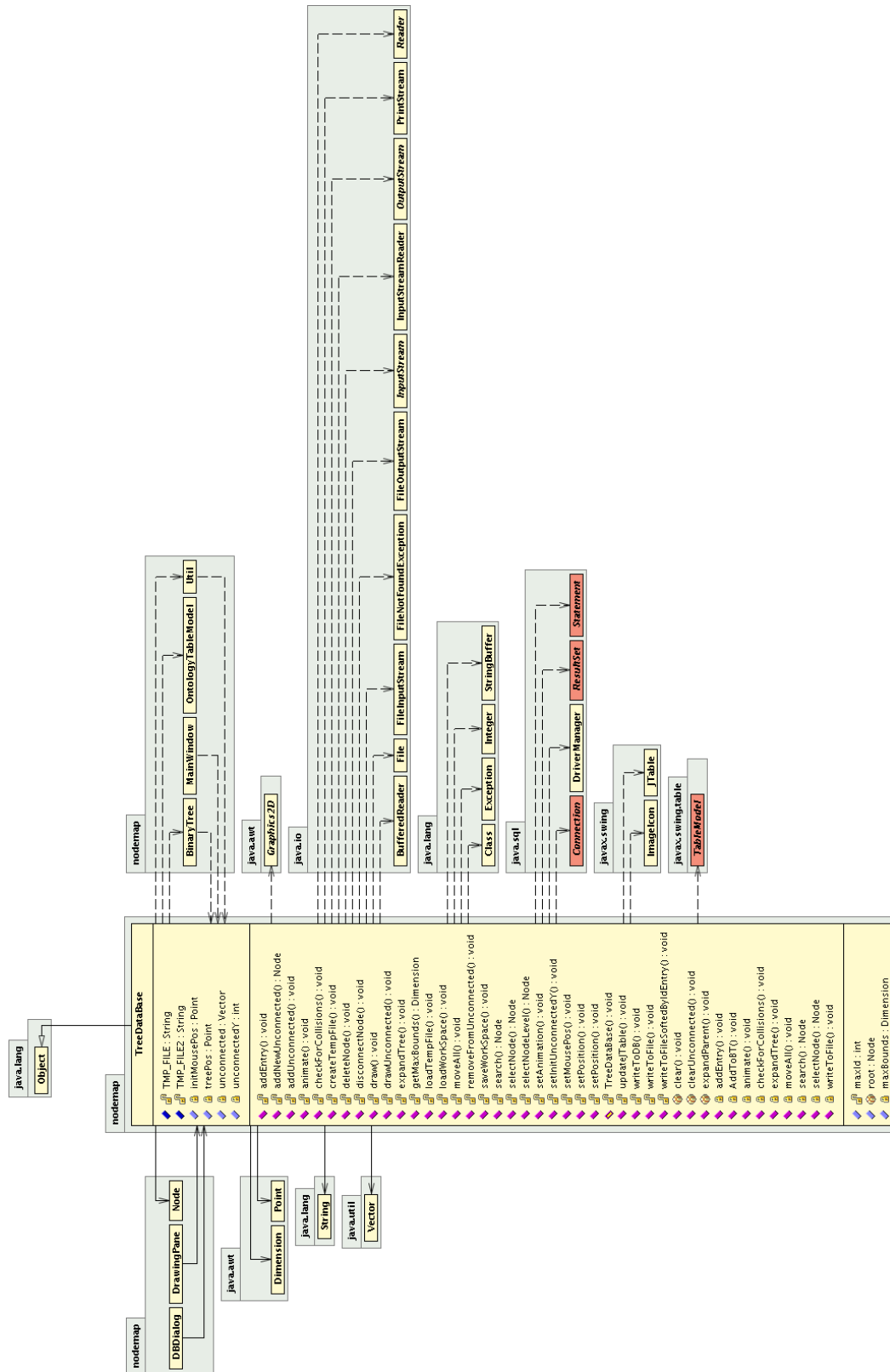


Figure A.1: TreeDataBase Java class

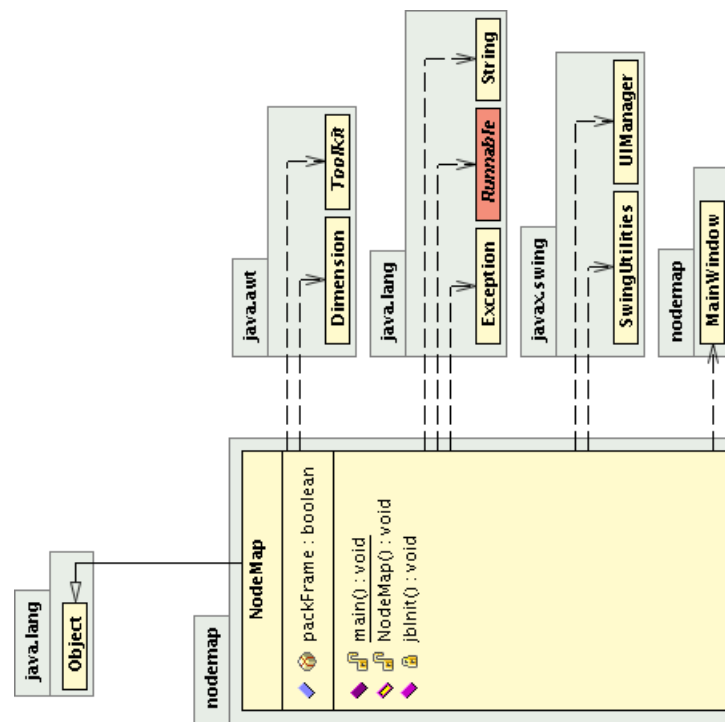


Figure A.2: NodeMap Java class

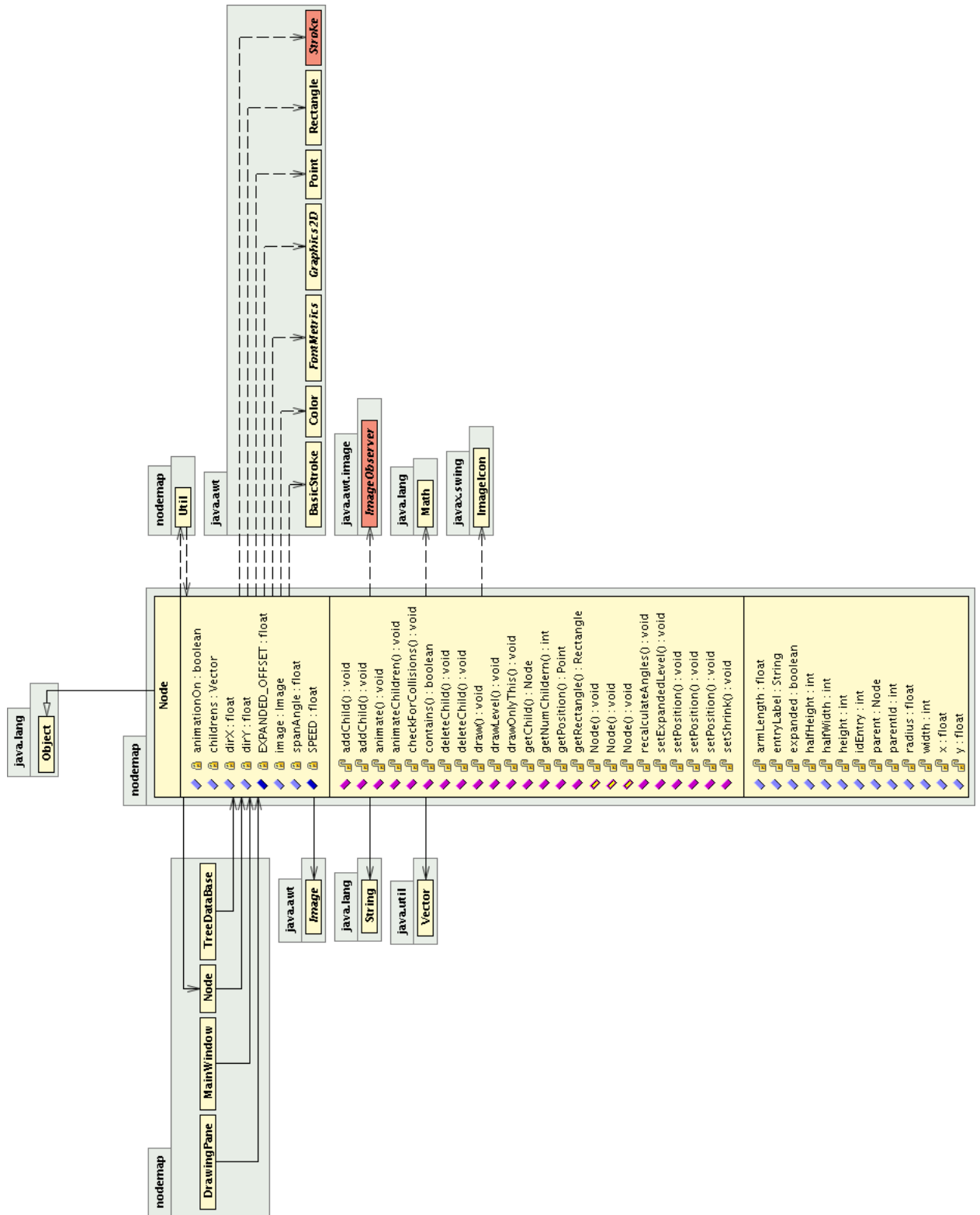


Figure A.3: Node Java class

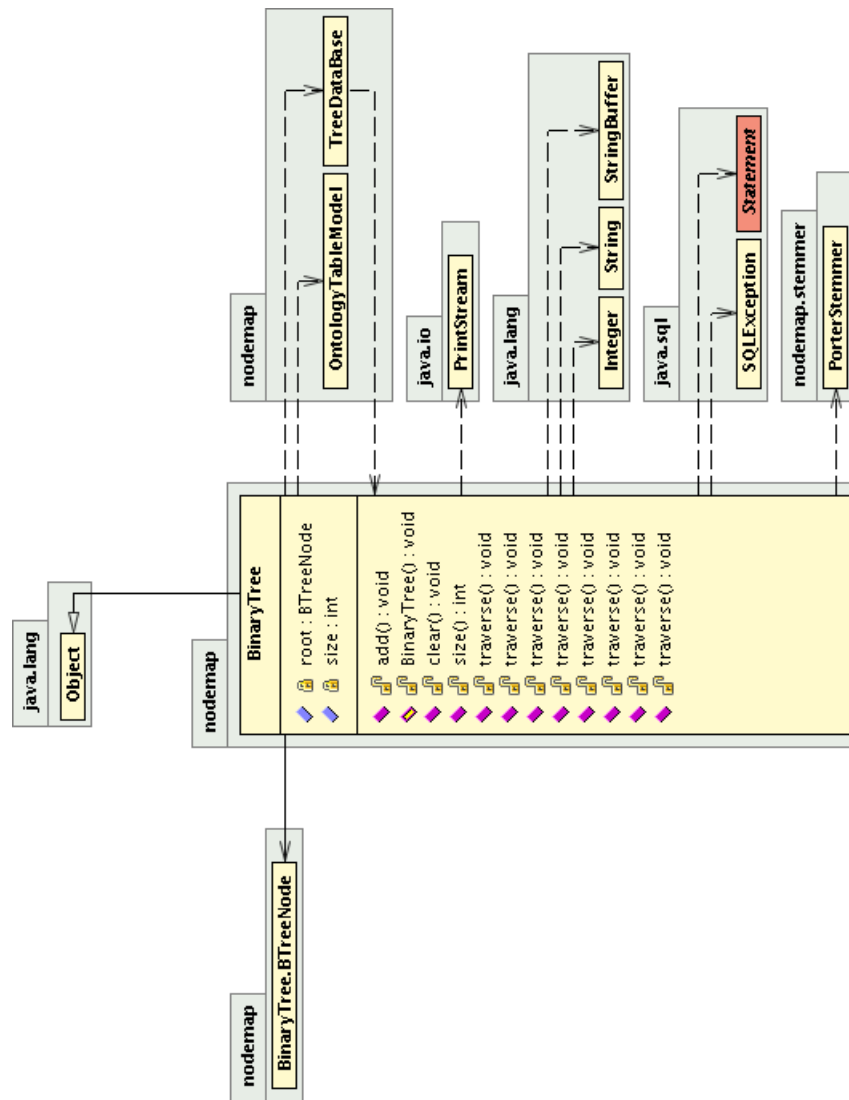


Figure A.4: BinaryTree Java class

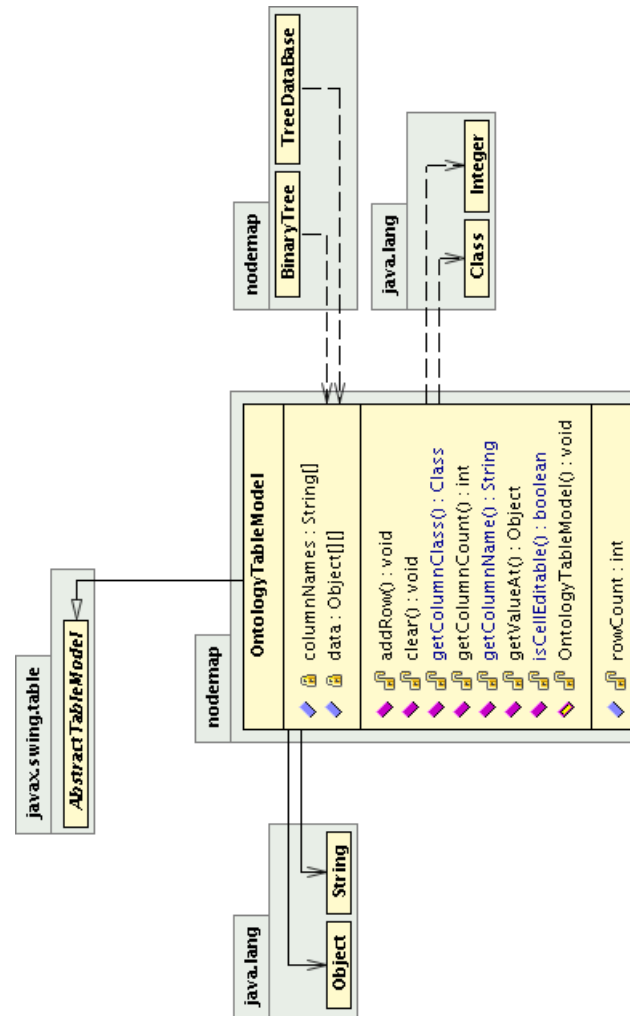


Figure A.5: OntoTableModel Java class

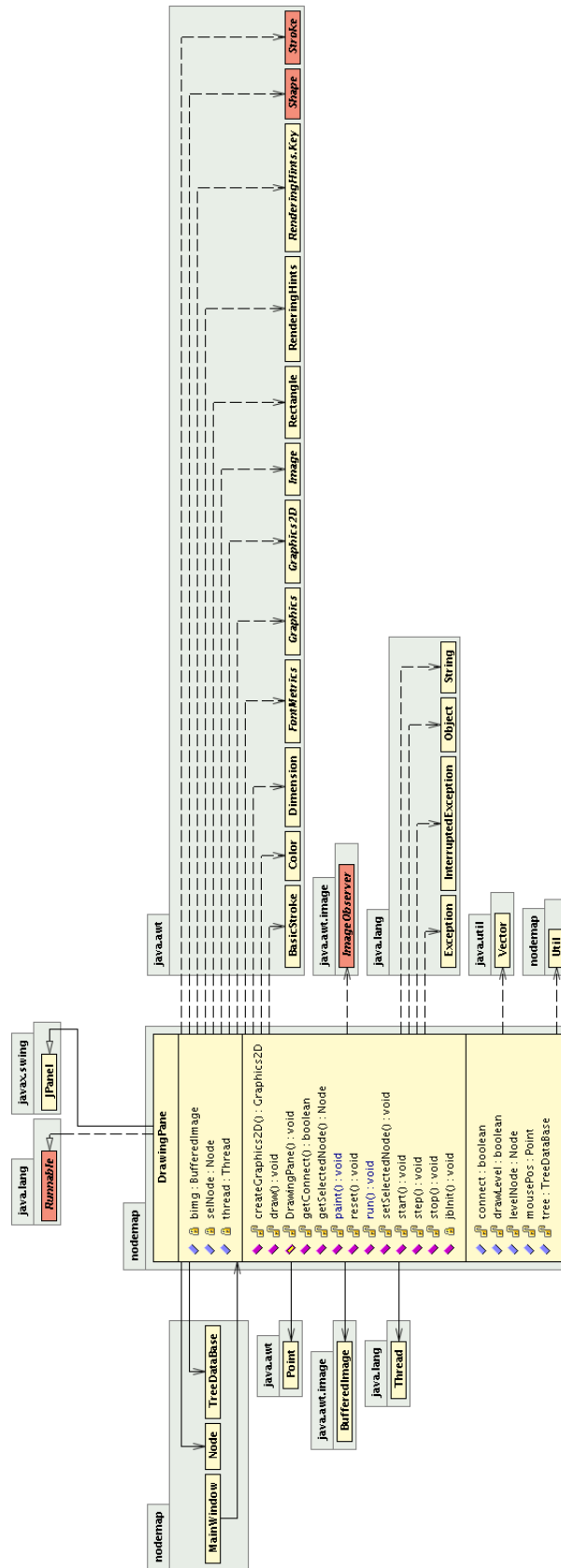


Figure A.6: DrawingPane Java class

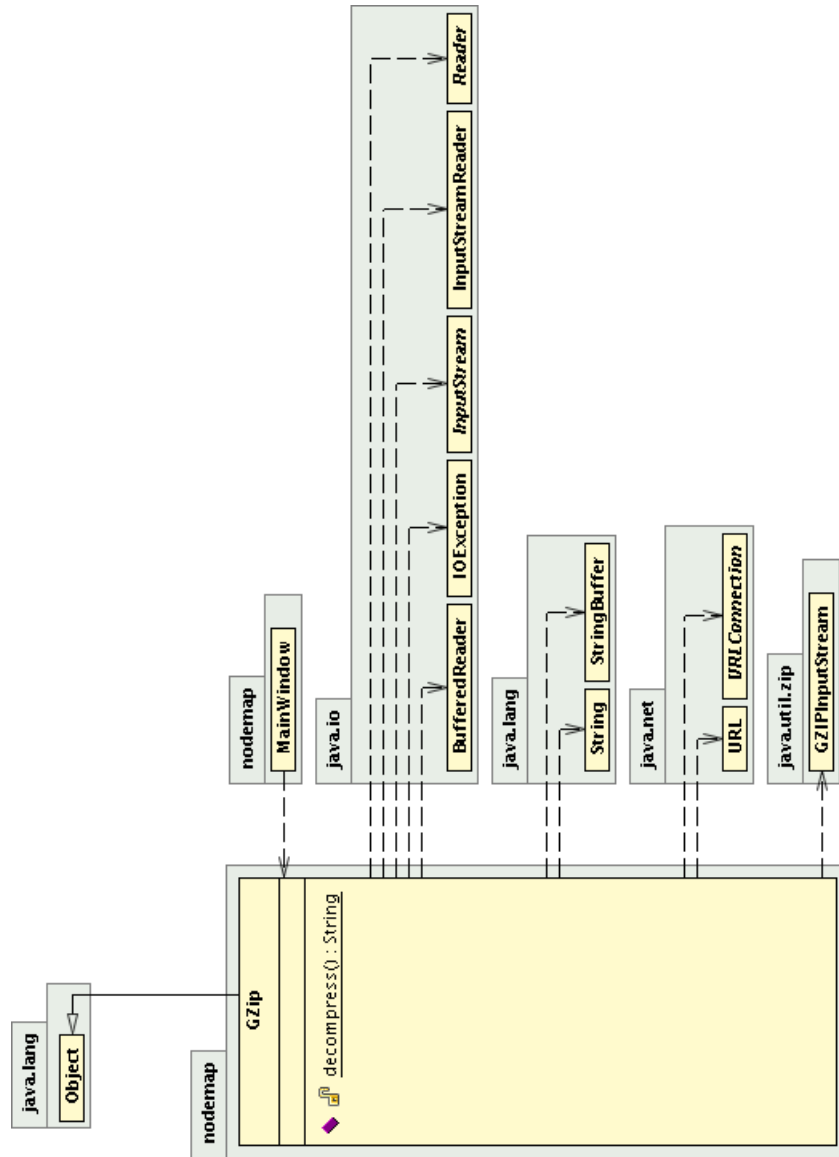


Figure A.7: GZip Java class

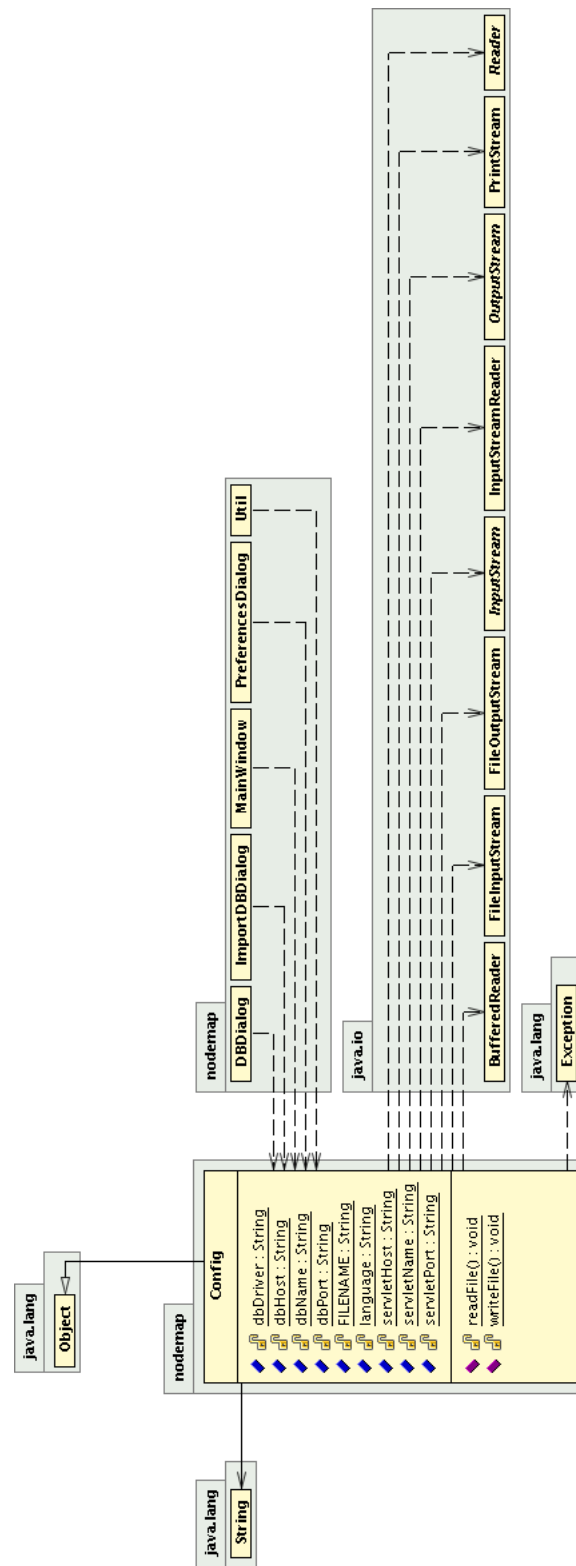


Figure A.8: Config Java class

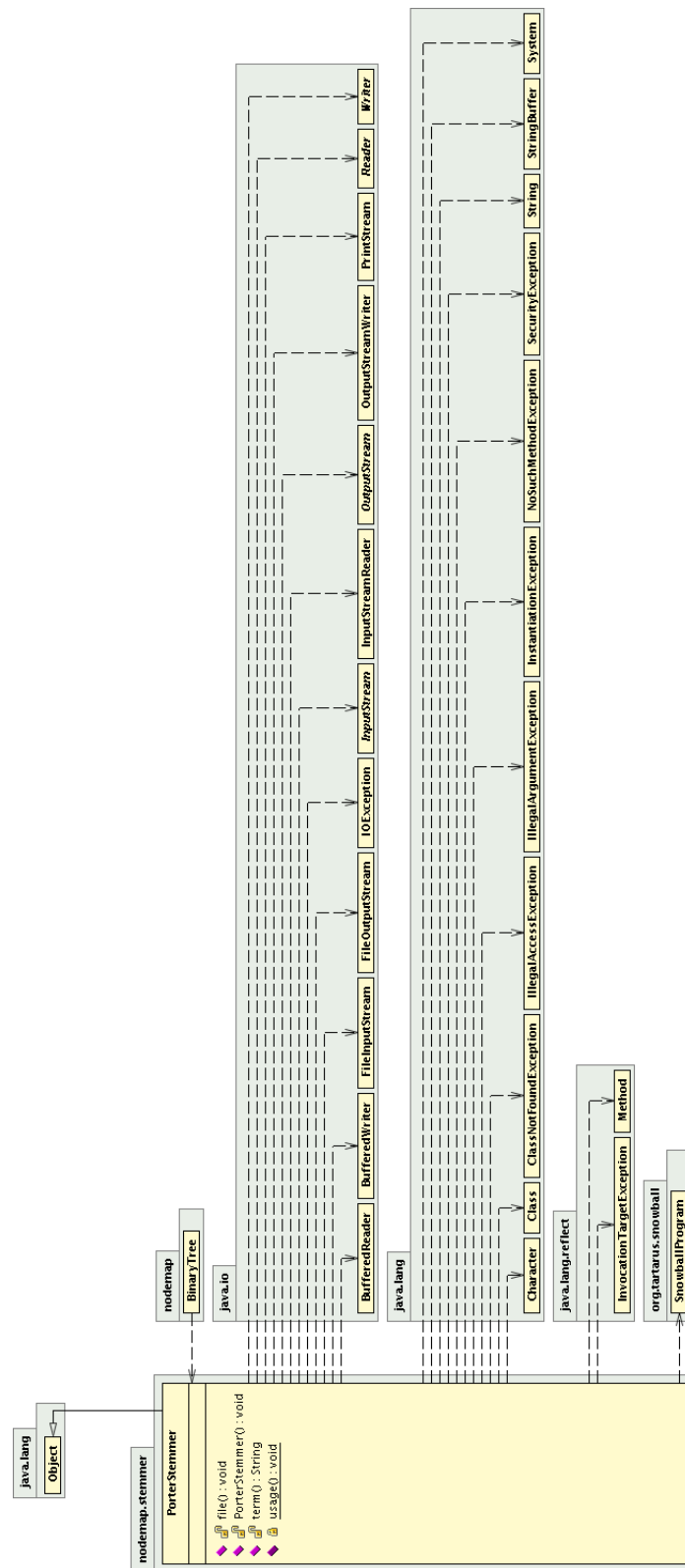


Figure A.9: PorterStemmer Java class

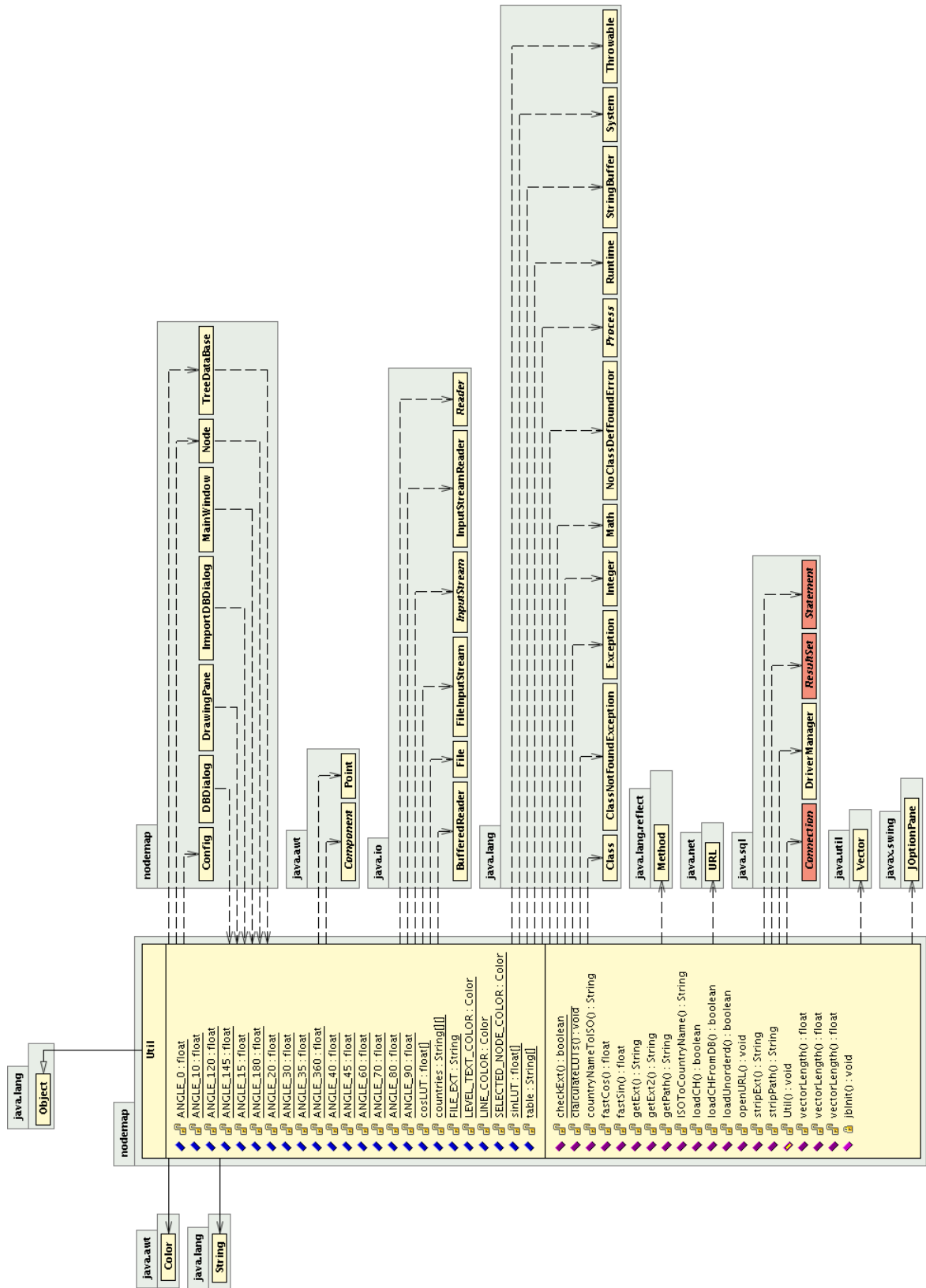


Figure A.10: Util Java class

NodeMap servlet UML diagrams

NodeMap servlet UML diagrams included in this appendix are the following:

- Figure A.11 shows the Cluster Java class
- Figure A.12 shows the MoreTerms Java class
- Figure A.13 shows the Matching Java class
- Figure A.14 shows the TextProcessing Java class
- Figure A.15 shows the Interpreter Java class
- Figure A.16 shows the Query Java class
- Figure A.17 shows the Search Java class
- Figure A.18 shows the Connect Java class
- Figure A.19 shows the TestServletConnection Java class
- Figure A.20 shows the ServletUtilities Java class

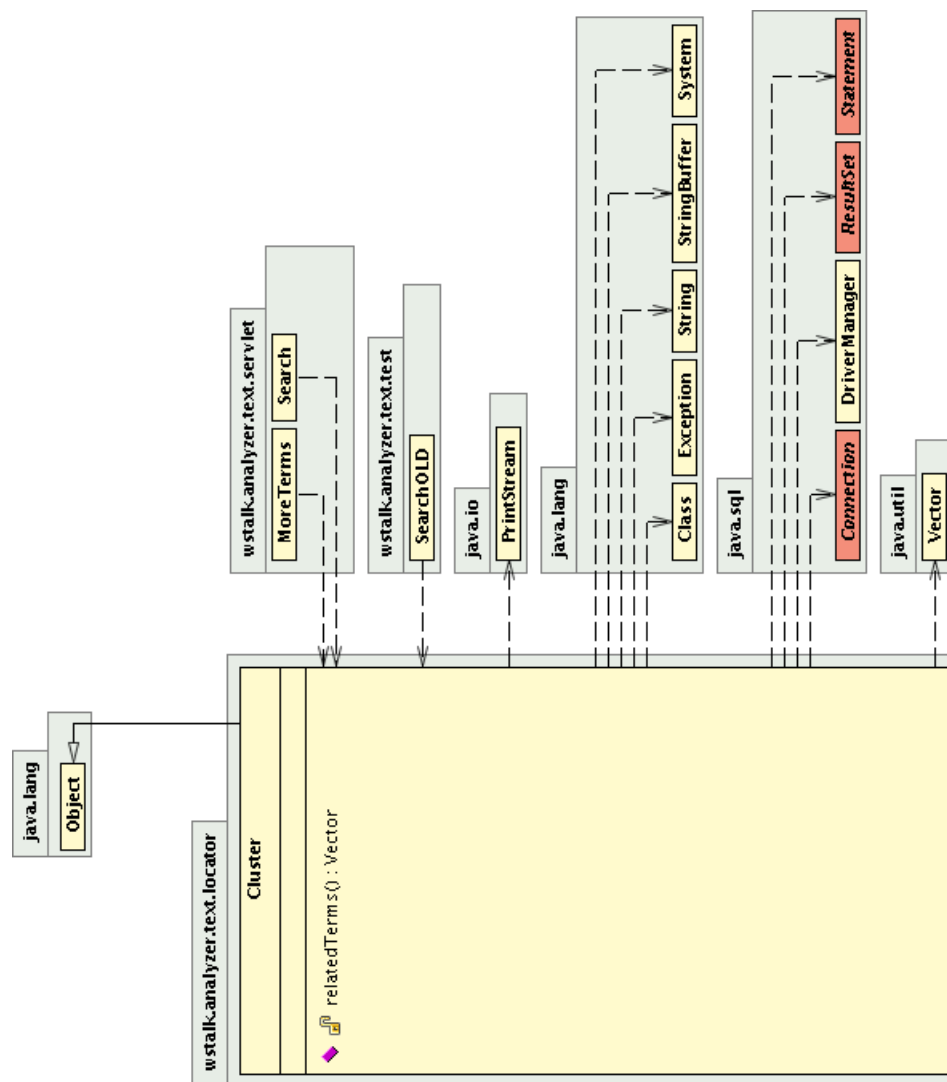


Figure A.11: Cluster Java class

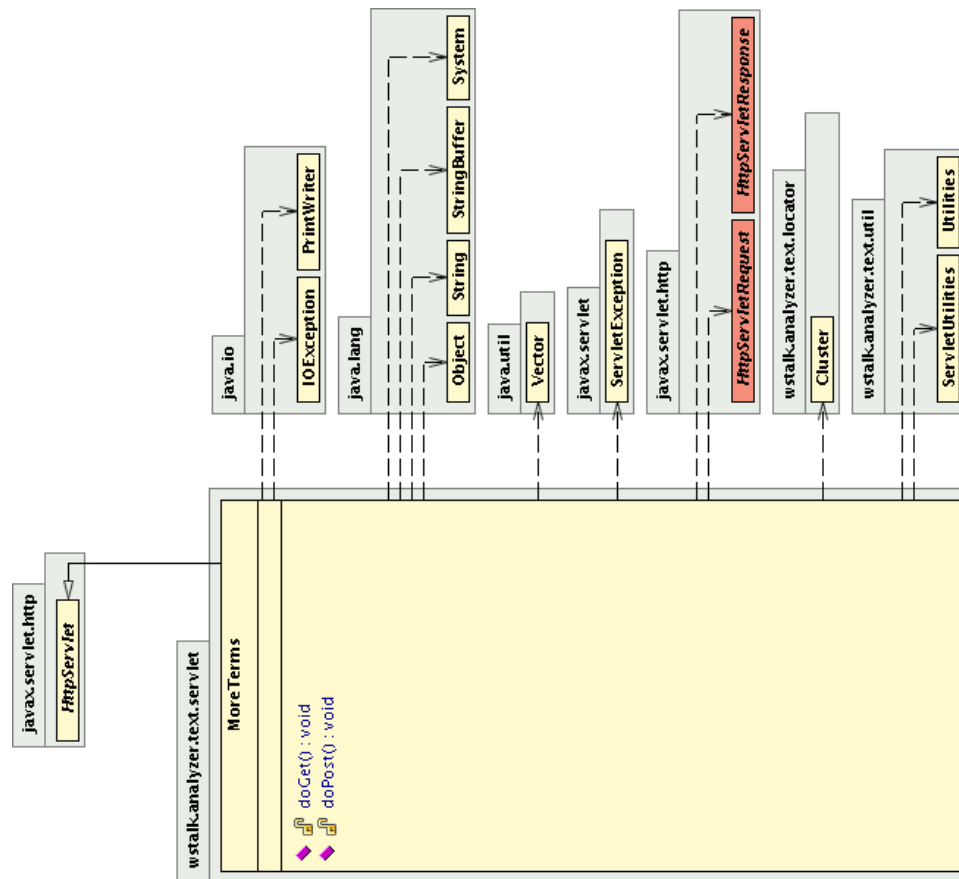


Figure A.12: MoreTerms Java class

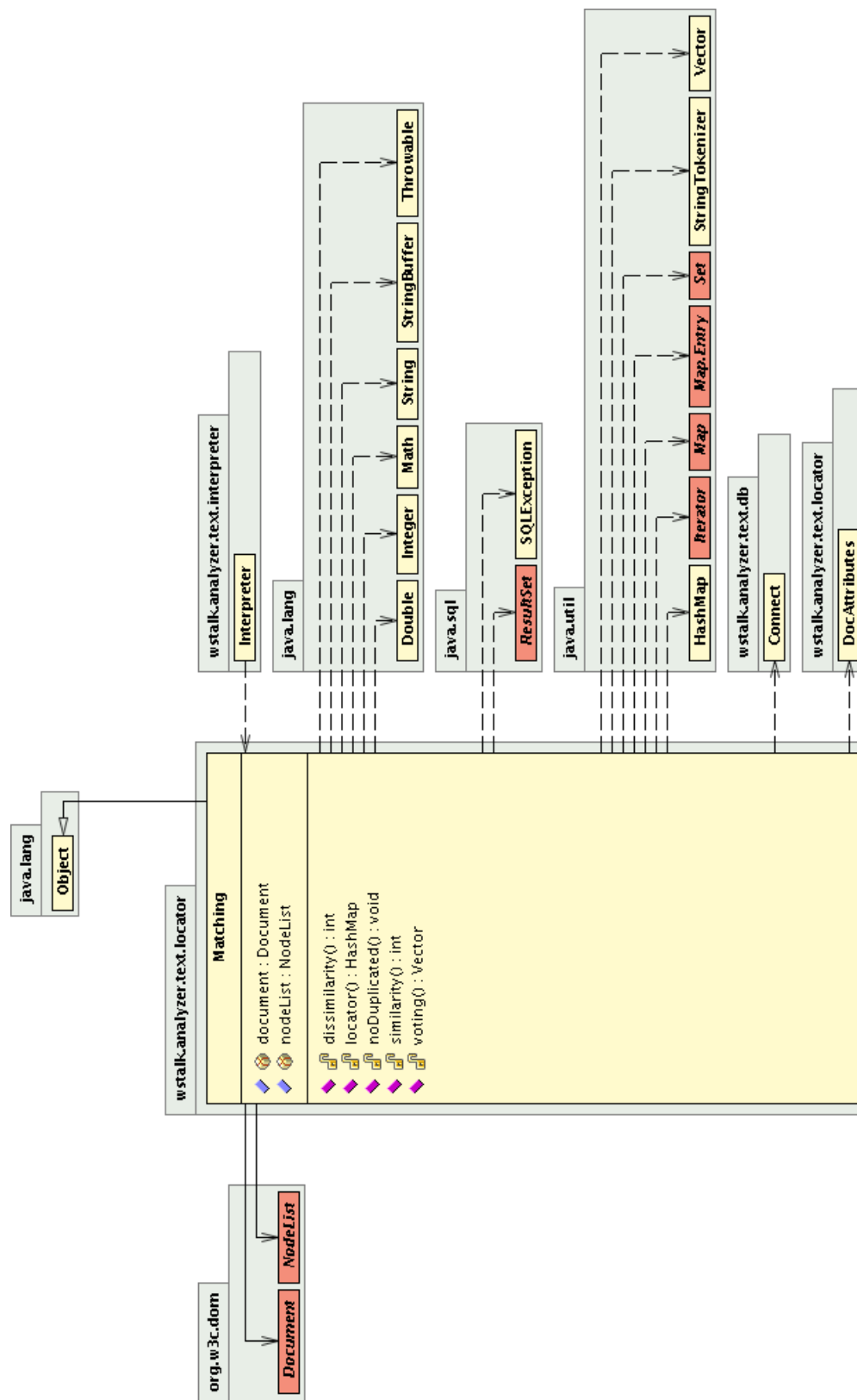


Figure A.13: Matching Java class

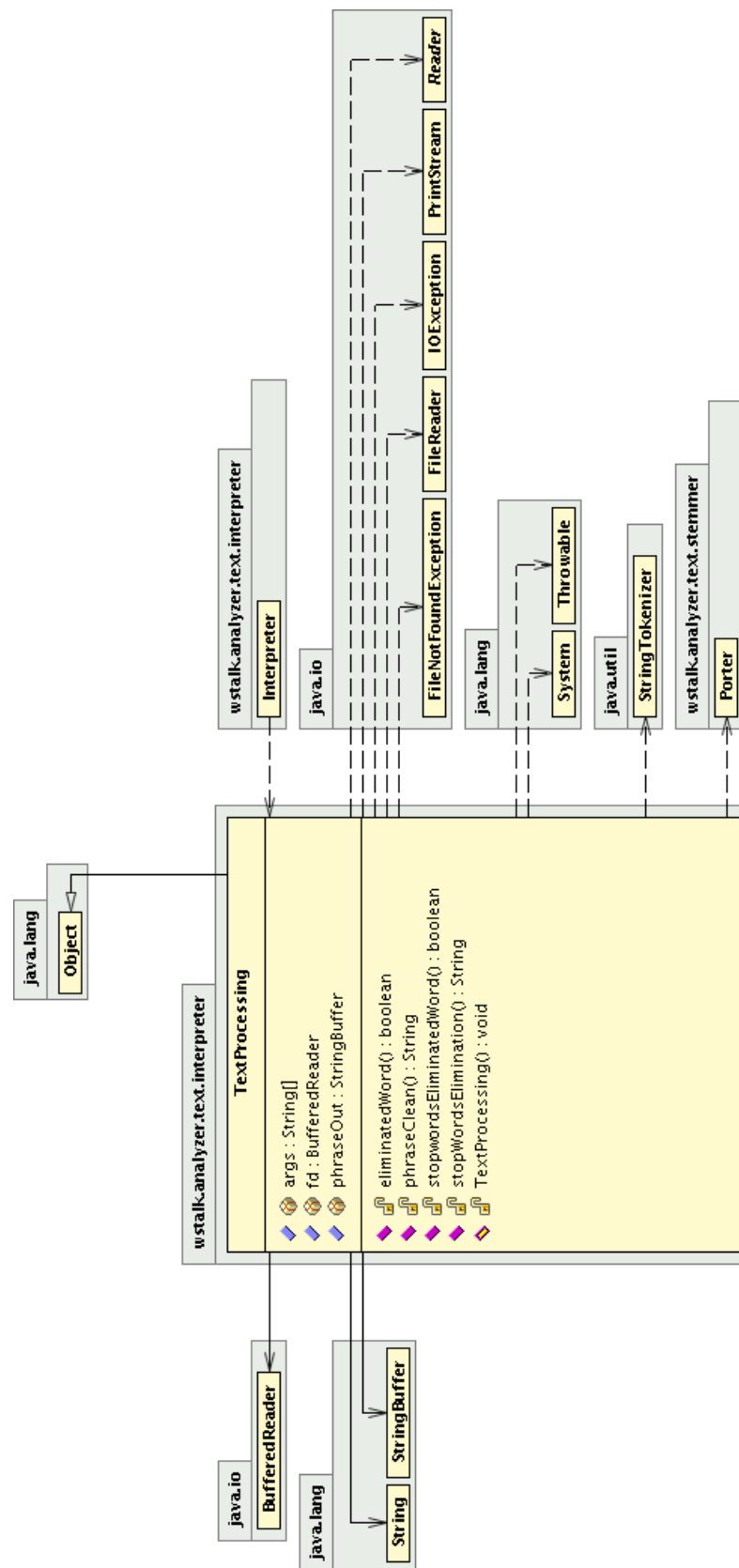


Figure A.14: TextProcessing Java class

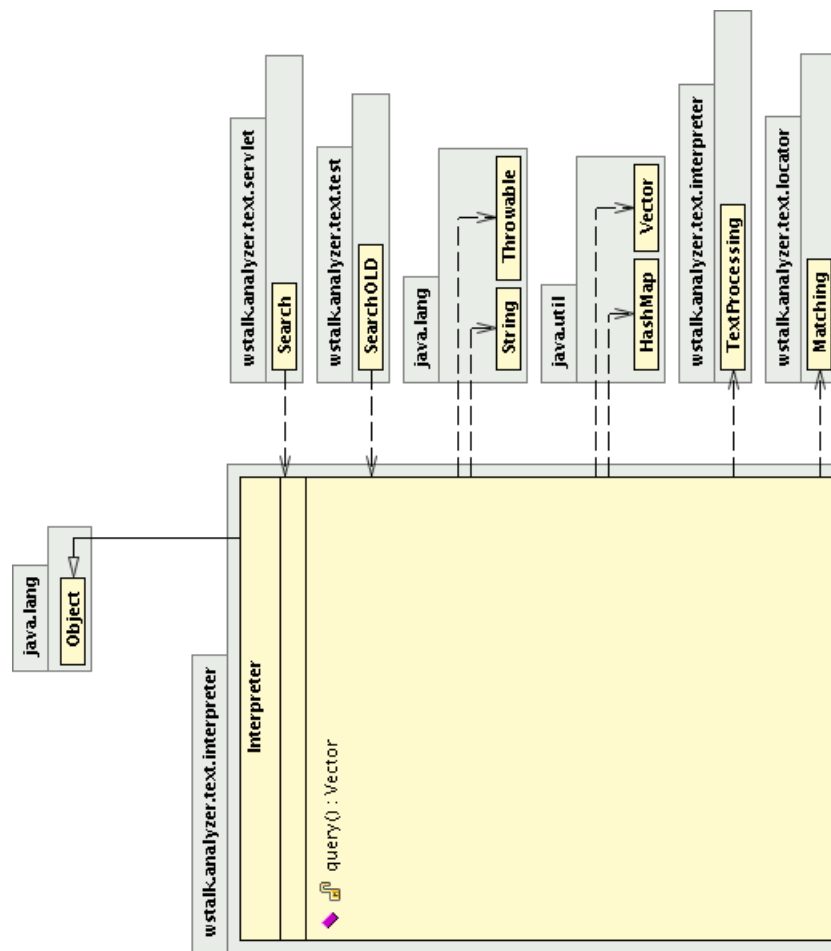


Figure A.15: Interpreter Java class

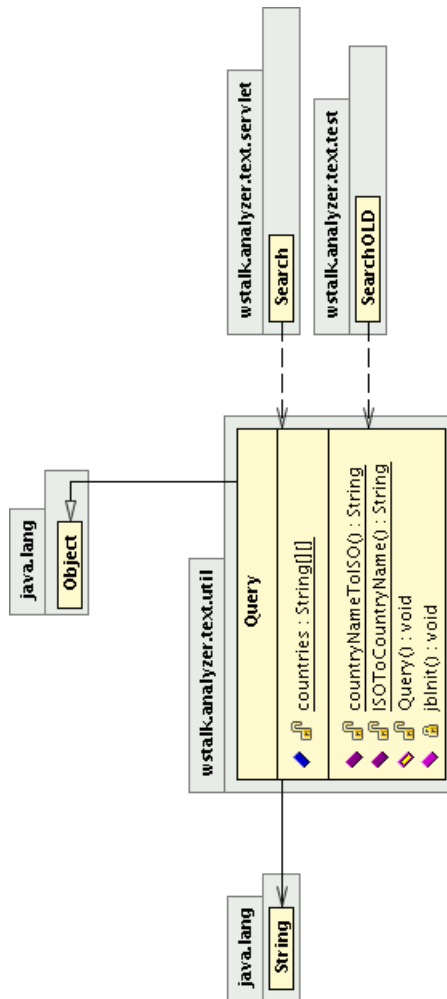


Figure A.16: Query Java class

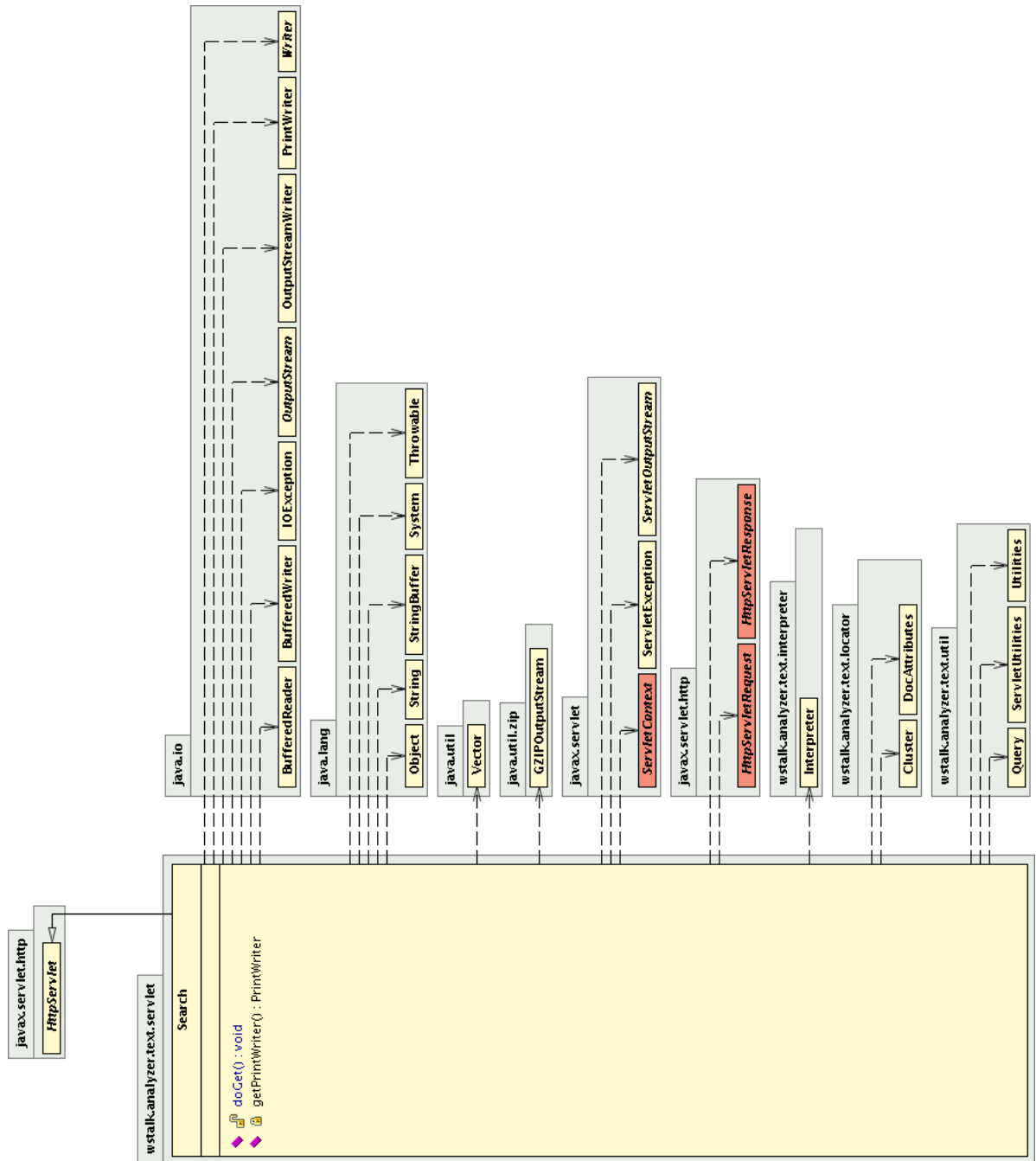


Figure A.17: Search Java class

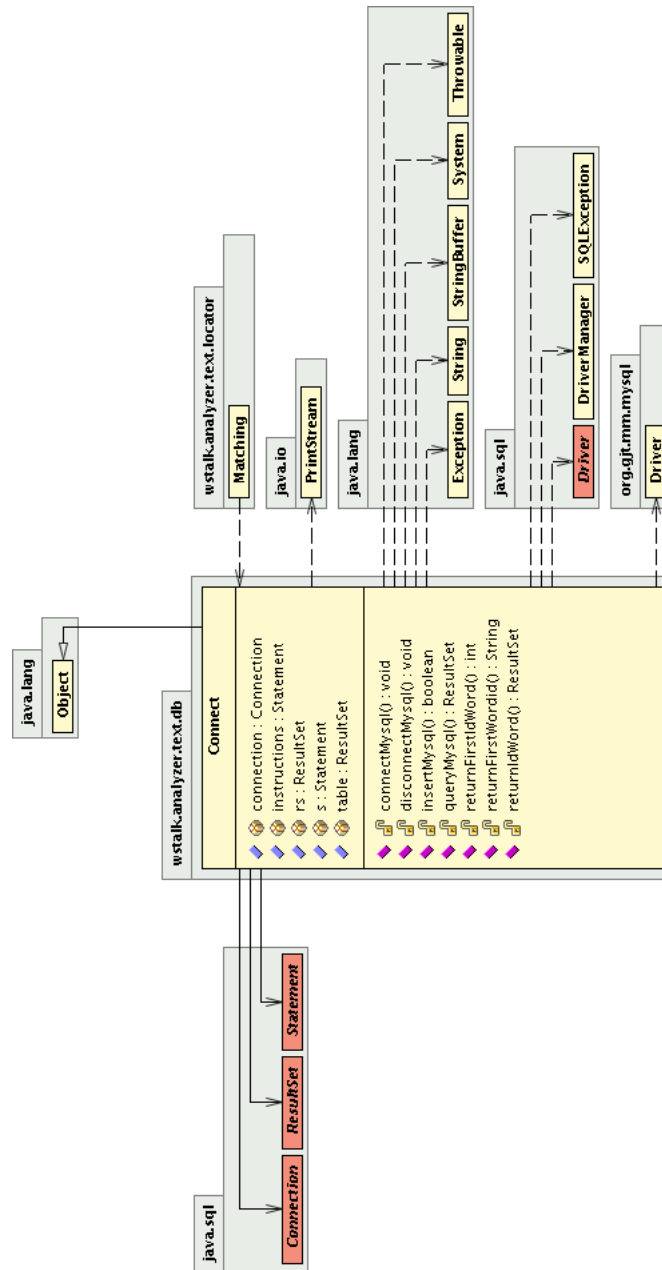


Figure A.18: Connect Java class

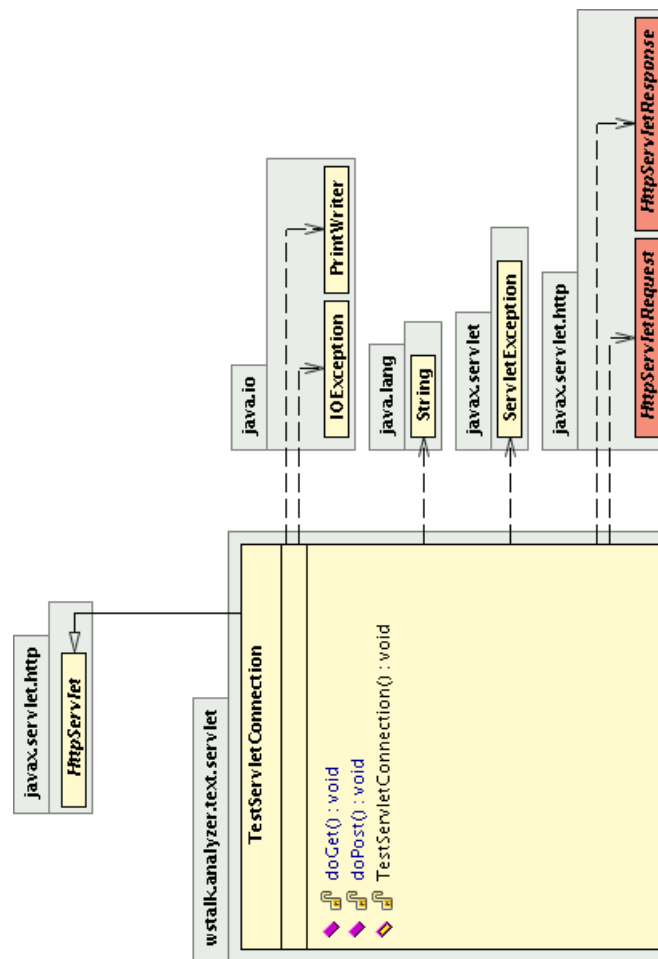


Figure A.19: TestServletConnection Java class

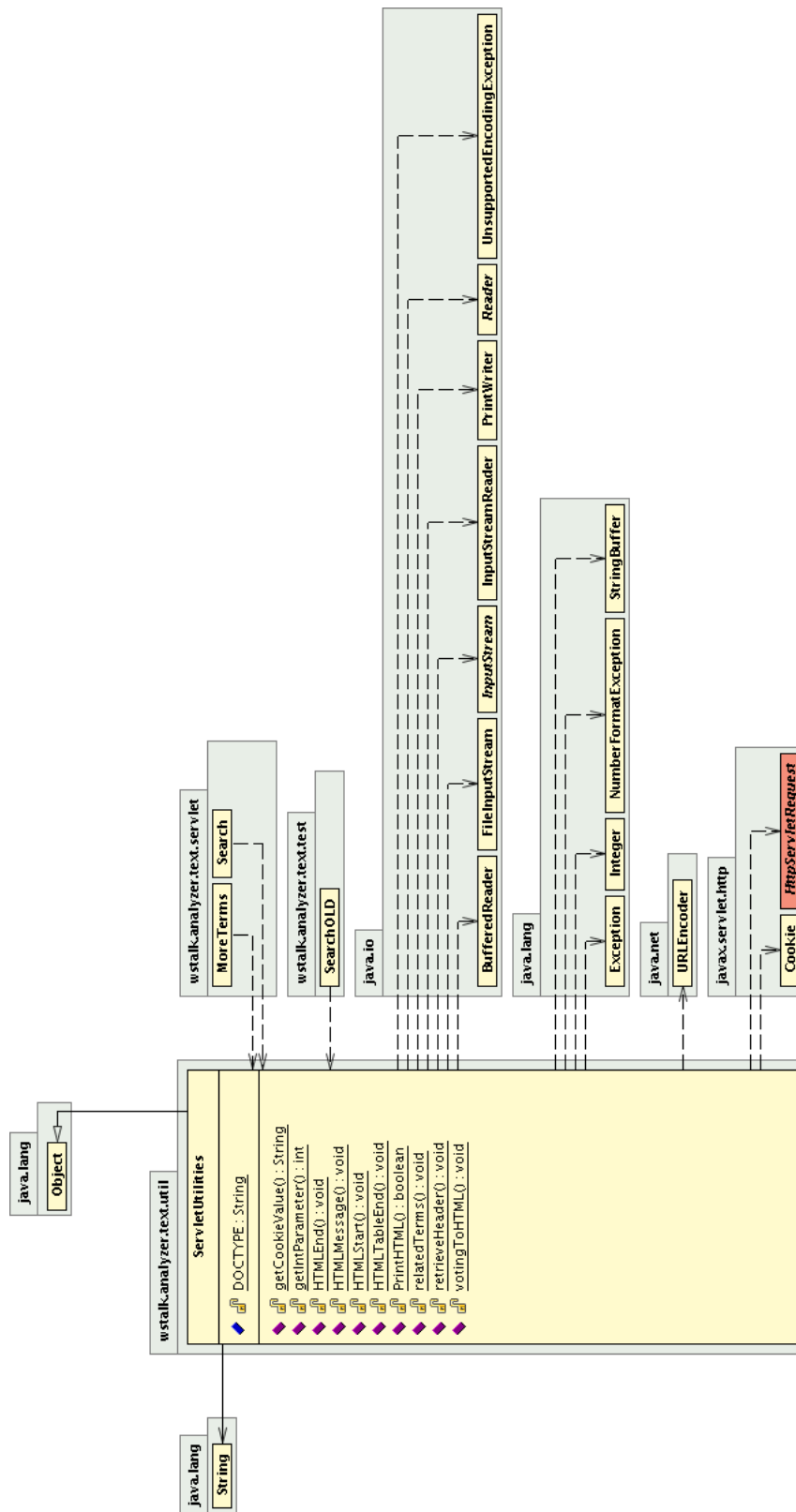


Figure A.20: ServletUtilities Java class

Appendix B

Astronomy ontology

The following presents an astronomy ontology that shows a Nebulae hierarchy obtained from the International Astronomical Union at <http://www.mso.anu.edu.au/library/thesaurus/english/hier-NEBULAE.html>. The first column presents a sequential number, second column presents the level within the ontology and finally the third column presents the label or node name.

1 0 Stars	10 9 Diffuse Nebulae	19 18 Emission Nebulae
2 1 Nebulae	11 9 Dust Nebulae	20 19 H II Regions
3 2 Bipolar Nebulae	12 2 Emission Nebulae	21 20 Compact H II Regions
4 2 Bright Nebulae	13 12 H II Regions	22 19 Herbig Haro Objects
5 2 Compact Nebulae	14 13 Compact H II Regions	23 19 Planetary Nebulae
6 2 Dark Nebulae	15 12 Herbig Haro Objects	24 2 Planetary Nebulae
7 6 Bok Globules	16 12 Planetary Nebulae	25 2 Reflection Nebulae
8 2 Dense Clouds	17 2 Filamentary Nebulae	26 2 Ring Nebulae
9 2 Dense Matter	18 2 Gaseous Nebulae	

Figure B.1 shows a section of the above ontology once loaded into the NodeMap software.

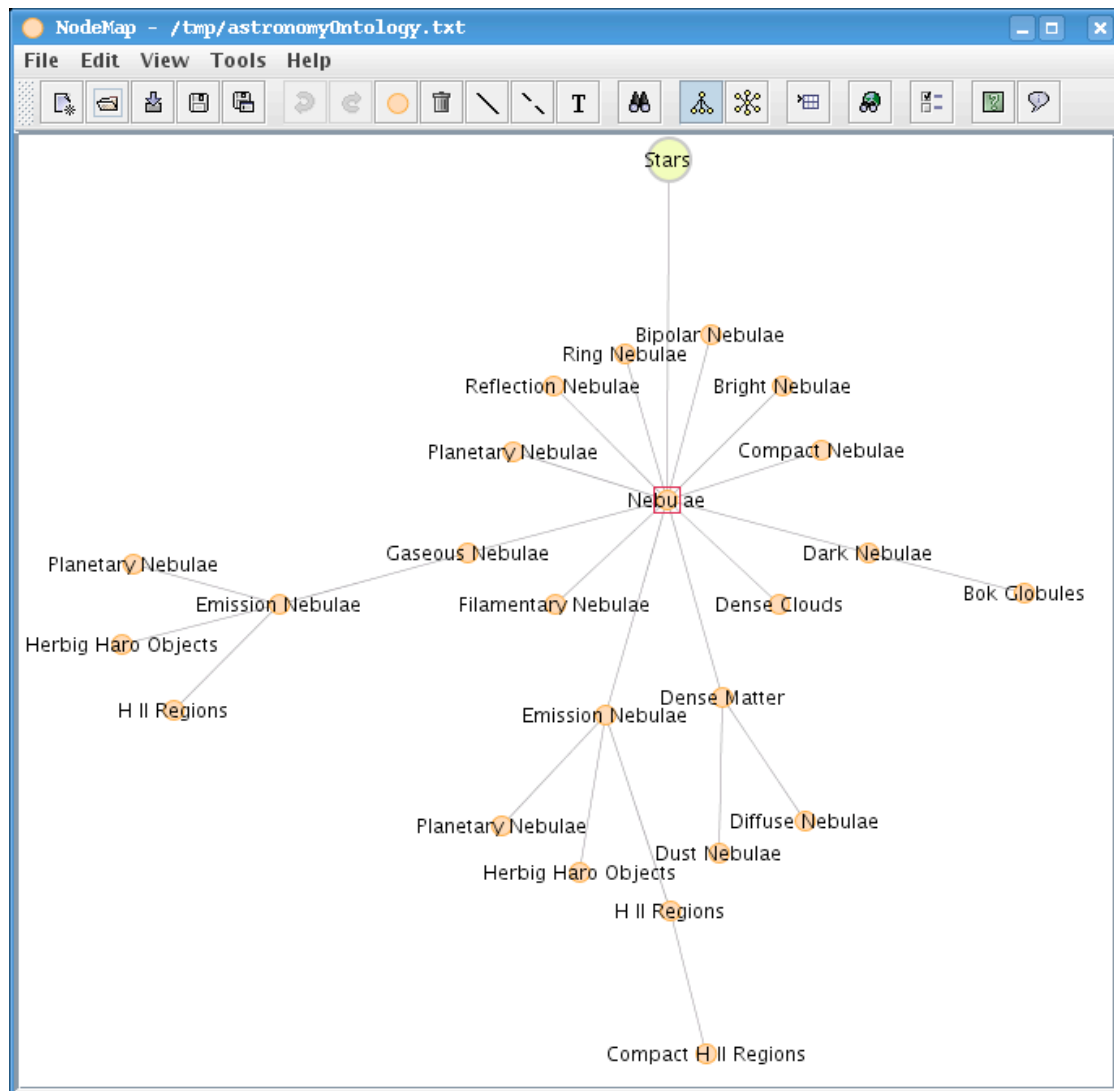


Figure B.1: Astronomy ontology displayed with NodeMap software

Appendix C

Animal ontology

The following presents an animal ontology, where the data is store in three columns. The first column presents a sequential number, second column presents the level within the ontology and finally the third column presents the label or node name.

1 0 animals	19 15 tuna	37 29 caimans
2 1 vertebrates	20 15 salmon	38 29 gharial
3 1 invertebrates	21 15 trout	39 7 carnivora
4 2 fish	22 5 salamanders	40 7 artiodactyla
5 2 amphibians	23 5 frogs	41 7 afrosoricida
6 2 reptiles	24 5 toads	42 39 felidae
7 2 mammals	25 23 arrow frog	43 39 mongooses
8 2 birds	26 23 bull frog	44 39 canidae
9 3 worms	27 6 snakes	45 42 tigers
10 3 molluscs	28 6 lizards	46 42 cats
11 3 arthropods	29 6 crocodilia	47 42 lions
12 3 corals	30 27 anaconda	48 43 civet
13 3 echinoderms	31 27 rattle snake	49 43 meerkats
14 4 sharks	32 27 asp	50 44 coyote
15 4 bony fish	33 27 cobra	51 44 wolf
16 14 great white shark	34 28 gecko	52 44 fox
17 14 tiger shark	35 28 chameleon	53 8 paleognathae
18 14 blue shark	36 29 crocodiles	54 8 neognathae

55 39 mustelidae	71 54 pigeons	88 11 myriapoda
56 55 weasels	72 54 parrots	89 11 crustacea
57 55 martens	73 54 owls	90 87 spiders
58 55 badgers	74 54 cuckoos	91 87 scorpions
59 55 otters	75 10 cephalopods	92 87 ticks
60 41 afrosoricida	76 10 gastropods	93 88 centipedes
61 41 paenungulata	78 75 octopus	94 88 millipedes
62 40 suina	79 75 squid	97 13 starfish
63 40 tylopoda	80 75 cuttlefish	98 13 sea urchins
64 40 ruminantia	81 76 snails	99 13 sea cucumbers
65 53 ostrich	82 76 slugs	100 89 crabs
66 53 emus	83 76 limpets	101 89 shrimps
67 53 kiwis	84 9 annelida	102 89 lobsters
68 54 eagles	85 9 nematomorpha	
69 54 hawks	86 9 onychophora	
70 54 falcons	87 11 chelicerata	

Figure C.1 shows a section of the above ontology once loaded into the NodeMap software.

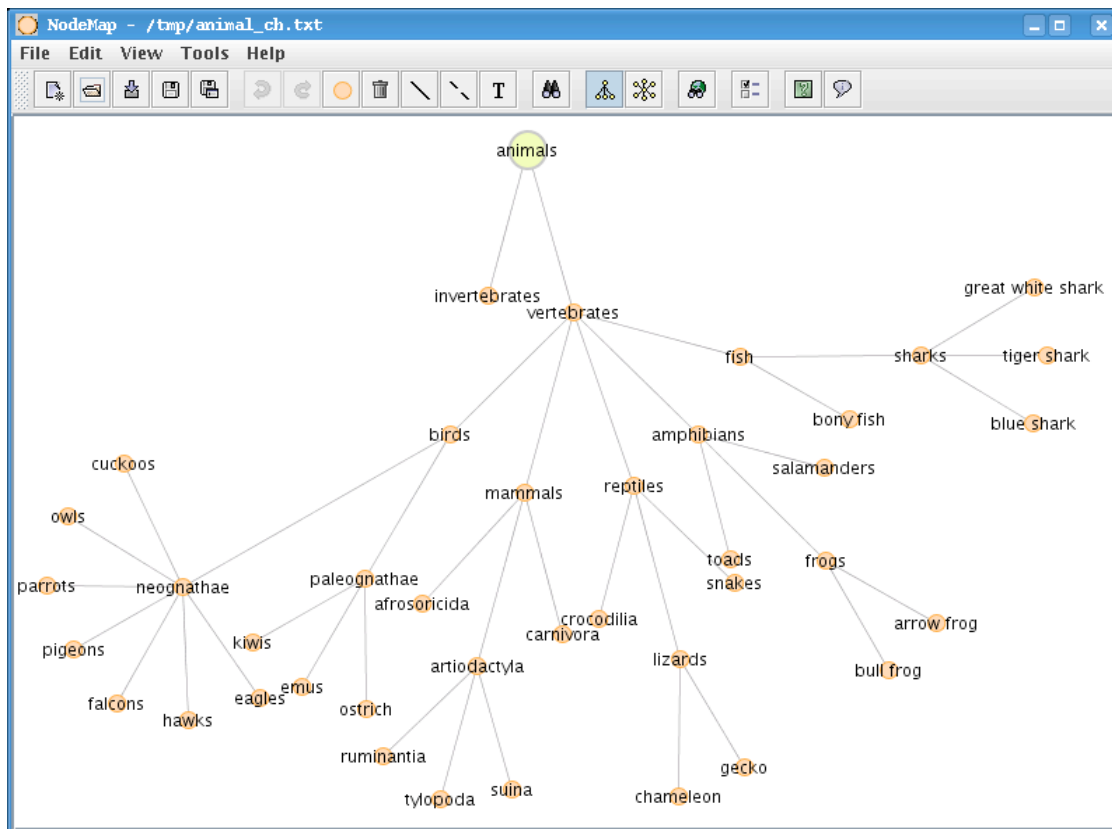


Figure C.1: Animal ontology displayed with NodeMap software