COBrA –
A Concept Ontology Browser for Anatomy

Roman Korf

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2003

Abstract

In developmental biology ontologies are used amongst other things to describe the anatomy of different species in meaningful ways. These ontologies developed independently for different species. An interesting task is now to establish homology links across these different species to use the knowledge of one organism for other ones.

In this MSc project I developed and implemented an application that displays two ontologies simultaneously and enables biologists to make homology links across species. Two different visualisation techniques have been developed and implemented to evaluate how different perspectives of the ontology can help in exploring ontologies and support the biologist in making homology links.

An evaluation has shown the potential of the different visualisations of the application and the functionality in achieving its supposed purpose.

Acknowledgements

First of all I want to thank my supervisor Stuart Aitken and my co-superviser Bonnie Webber. They have always been a great help in technical and organizational questions.

I also want to thank Stephen Potter. He has been a great help in the last days of writing this dissertation and made very helpful and constructive suggestions for improvement.

Then I want to thank Jonathan Bard and Duncan Davidson for taking some time to help me with the my project and the experiments.

Many thanks also to Fran, Vicky, Giorgos, Gissela, Jay and Charles for spending their valuable time helping me with my experiments.

Special thanks to Fran, Vicky, Giorgos, Alexandros and Jordi for always being great friends, in good and in not so good times.

At last but not at least I want to thank Rainer Stozka and Tim Müller. Without their help I would not have be able to come to Edinburgh.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Roman Korf*)

Table of Contents

1 Introduction **1**

1.1 Thesis Structure 2

2 Background **4**

2.1 What are Ontologies 4

2.1.1 Use of Ontologies 5

2.1.2 Representation of Ontologies 6

2.1.3 Visualisation of Ontologies 8

2.1.4 Ontology Editors and Browser 11

2.2 Ontologies in Biology 15

2.2.1 Representation of Biological Ontologies 16

2.2.2 The GO File Formats 16

2.2.3 DAG-Edit 16

2.3 Software Evaluation 17

2.3.1 What is software evaluation 17

2.3.2 What are the Goals of Evaluation? 19

2.3.3 Evaluation Methods 20

3 COBrA – Ontology Browser **25**

3.1 Background Data Structure 25

3.1.1 Concept 26

3.1.2 Design 27

3.1.3 Realisation Issues 27

3.2 Visualisation of the Ontology 28

3.2.1 Tree Visualisation 30
3.2.2 Node Based Visualisation 31
3.2.3 Connection between Tree and Node Based Visualisation . . . 33
3.2.4 Selecting the Visualisation Type 34
3.2.5 Reducing the Visible Complexity of the Ontology 35
3.3 Editing the Ontology 36
3.3.1 Menu Driven Editing in the Tree 36
3.3.2 Direct Editing in the Tree 36
3.4 Making Homology Links 37
3.4.1 Concept 37
3.4.2 Design 38
3.4.3 Realisation Issues 39

4 Experiments and Results **41**
4.1 Preparation and Realisation of the Evaluation 41
4.1.1 Aim of the Evaluation 41
4.1.2 The Participants 43
4.1.3 The Experimental Methodology 43
4.1.4 The Tasks 45
4.1.5 The Evaluation Methods 49
4.2 The Results of the Software Evaluation 53
4.2.1 User Experience 53
4.2.2 Issues Identified while Performing the Tasks 54
4.2.3 Issues Identified through the Questionnaire 57
4.2.4 Additional Issues Raised during the Evaluation 59
4.3 Analysis of the Results 60
4.3.1 Issues Identified while Performing the Tasks 60
4.3.2 Issues Identified through the Questionnaire 62
4.3.3 Testing the Hypotheses 63

5 Conclusions and Future Work **66**
5.1 Conclusions 66

5.2 Future Work 67

A Results **69**

B Usability Questionnaire **71**

C GO Flat-File **75**

D GO XML-File **78**

Bibliography **81**

Chapter 1

Introduction

With the growing amount of biological information it has become increasingly important to describe biological objects in meaningful ways. Ontologies provide a powerful instrument to structure the terms and describe relationships between these terms.
An interesting question in developmental biology surrounds which genes play which roles in the development of organisms. The research in this area is done mainly on distinct species and the structured vocabulary that describes the anatomy of these different species has been developed independently. Currently the most investigated species are the fruit fly (Drosophila melanogaster[1]), the round worm (Caenorhabditis elegans[2]) and, for mammalian development, the mouse[3]. (See Figure 3.5)

(a) (b) (c)

Figure 1.1: (a) shows the fruit fly, (b) the round worm and (c) the mouse.

[1]http://www.flybase.org/
[2]http://genex.hgu.mrc.ac.uk/
[3]http://www.wormbase.org/

One particular task in developmental biology is now to establish homology links between ontologies of different species. (Here homology link refers to the similarity between two tissues.) This enables biologists to use the knowledge about one organism for other organisms. With the knowledge of how drugs act in one organism they could, for instance, make assumptions how these drugs would act in other organisms.
The aim of this MSc Project is to develop and implement an easy to use tool that enables developmental biologists to explore and edit ontologies and to establish homology links between different species.
To make the homology links the biologist will need to access multiple ontologies. First the biologist will select two tissues that shall be mapped in the two ontologies of the anatomies of different species. Than the biologist will select one or more cell types from the cell-ontology which describes the different cell types. The cell type defines the homology between the two tissues.
The biologist will often need to access different information or different perspectives on the ontology. Sometimes she might need to display this information simultaneously. For this some efficient ways to present the information need to be developed and implemented. The homology information has to be stored efficient and needs to be easily accessible by the biologist.
At the end of the design and implementation the application will need to be evaluated with the aid of evaluation methods that involve users. For this different evaluation methods will be combined to gather data from different sources. This is done to compare the results of the different methods to get more reliable results. The evaluation will test the applicability of the two visualisations for the task they are supposed to help. It will also test the understandability of the functionality and it will gather the subjective opinion of the participants about the application.

1.1 Thesis Structure

In chapter 2 I will first introduce ontologies, their applications and some issues about representing ontologies. Then I will introduce different visualisation techniques to present ontologies on the screen. This is the background that I consider as the vital

knowledge to understand the design and implementation issues of the work. After this I will give a short introduction to software evaluation and introduce the methodology of the evaluation methods I applied during the software evaluation.
The design and implementation issues will be discussed in chapter 3. First the underlying data structure to store the ontologies will be described. Then I will describe the implemented visualisation methods. After this I will address the issues of the editing functionality and at the end I describe the implementation and realisation of the homology mapping.
In chapter 4 I will explain the evaluation performed with the users to evaluate the usability and adequacy of the application for the task. Then I present the results of the evaluation and attempt an analysis of the results.
The 5th chapter will present the conclusions about the work and the relevance of the project. I will explain what issues have been raised and what this means for the adequacy of the application. At the end I will propose some future work and briefly discuss some features that might increase the usability of the application.

Chapter 2

Background

This chapter presents definitions and explanations about the background of the project that are important to know. First I will shortly describe what ontologies are, what they are used for, how they are represented and how they are visualised in graphic computer applications. Then I will give a short introduction to software evaluation and describe some of the methods used in this field.

2.1 What are Ontologies

An efficient communication between two or more agents[1], in order to exchange knowledge, needs a *"common understanding"* of the concepts described in the world[2] (Noy & McGuinness, 2001). This means that agents not just need to agree about the terms used for the concepts described, but also need to have an agreement about the meaning of these terms and the relationships among those terms. One major problem that might emerge is that two agents use the same term but with a different semantics. The two agents would believe they talk about the same thing, but they do not. Similar problems might emerge when two agents use different terms that have the same meaning. The agents would not recognise that they actually talk about the same thing.
The aim of ontologies is to improve the efficiency and consistency in communication between agents. They do this by defining hierarchical organised vocabularies which

[1] Agent here means either software agents or agents of natural kind like humans.
[2] The term *world* here refers to a domain of interest that the two agents share.

describe the concepts of the domain of interest, the relationships among these concepts and their properties. The concepts are sometimes also called classes or "*categories* [that are] *used to model the world.*" (Guarino & Giaretta, 1995). These categories and their relationships are clearly defined in the domain. Properties, also sometimes referred to as slots, describing the *"various features and attributes of the concept"* (Noy & McGuinness, 2001).

Typical relationships between concepts are *is-a*– and *part-of*–relationships. The first can be compared with *generalisation* and *inheritance* from object-oriented programming, the later is sometimes referred to as the *part-whole*–relationship. An example of the *is-a*–relationship from biology is: a *vacuolar membrane is-a membrane*. The concept *vacuolar membrane*, which is not further specified here, is a sub concept of the concept *membrane* and has the properties of the concept *membrane*. In addition the concept *vacuolar membrane* would have additional properties, otherwise the concept *vacuolar membrane* would be the same as the concept *membrane* and therefore redundant. An example for the *part-of*–relationship is: a *vacuolar membrane* is *part-of vacuole*. Here the concept *vacuolar membrane* is a component of the concept *vacuole*. The concept *vacuole* also consists of other components, that are *part-of* it. If it would just consist of the component *vacuolar membrane,* then the concept *vacuolar membrane* and the concept *vacuole* would be the same and therefore one of them would be redundant.

Instantiation is another important relationship between ontologies. This relationship describes the relation between a concept and the instances of that concept. Instances are *"entities of the world"*. These can be physical objects, events or regions for example. (Guarino & Giaretta, 1995).

In the literature there are various cited definitions about ontologies (e.g. Gruber (1993), Guarino & Giaretta (1995), Guarino (1997), Borst et al. (1997)).

2.1.1 Use of Ontologies

Ontologies are used in different research areas: *"knowledge representation, natural language processing, information retrieval, databases, knowledge management, on line database integration, digital libraries, geographic information systems, visual in-*

formation retrieval or multi agent systems." (Nieto, 2003)
In e-commerce ontologies are used to enable communication between buyer and seller. In search engines ontologies are used to find web pages containing semantically similar words provided by the user.
Knowledge about the application domains is very important in order to have successful software projects. This knowledge is usually elicited from domain experts that know about business processes, customer relations, other relevant processes, products and relationships. In this case a powerful ontology for modelling the domain knowledge can be very helpful. Even for the elicitation of the domain knowledge an ontology can structure the process and prevent ambiguity. (Knublauch, 2003)
In general ontologies are used in the research on *data-interchange*, for *data-integration*, *data-querying* and *data-verification*. (Frank van Harmelen et al., 2001)

2.1.2 Representation of Ontologies

In order to represent ontologies, there must be a structure to represent them. This can be either in memory or in other structures like files. In the recent years XML[1], RDF[2] and extensions of these representation techniques have been widely used to represent ontologies. This chapter does not cover all aspects of representing ontologies and it is not meant to be complete. But it covers the ones that are important for this dissertation.

2.1.2.1 XML

XML is a meta-language to define individual markup languages like HTML[3]. It was recommended by the W3C[4] in 1998.
XML uses tags to build the structure of documents, and attributes that describe the information stored between the tags. The tags divide an XML-document into its parts, and identify the different parts of the document (Canfora & Cerulo, 2002). The XML-document does not contain any semantic information since the tags are not predefined.

[1]Extensible Markup Language : http://www.w3.org/XML/
[2]Resource Description Framework : http://www.w3.org/RDF/
[3]Hyper-Text Markup Language : http://www.w3.org/MarkUp/
[4]World Wide Web Consortium : http://www.w3.org/

The semantics of an XML document can either be defined by an application that processes the document or a style sheet. (Walsh, 1998)
The markup-languages in XML-documents are described in DTDs[1]. DTDs describe the structure of XML-documents and the valid order in which the tags can appear. An XML-document can be validated against a DTD.
Like DTDs XML-Schemas[2] describe the content of XML-documents. Elements in XML-Schema files are described in XML itself. XML-Schemas provide a wider range of different data types then DTDs. Therefore they provide a better control over the information stored in XML-documents. In XML-Schemas someone can specify own data-types.

2.1.2.2 RDF

The Resource Description Framework (RDF) is a model to represent meta-data in particular on the World Wide Web. RDF is specified in a W3C recommendation (Lassila & Swick, 1998) from 1998. The aim of the model is to describe resources domain and application independent without defining a semantic on the resources. The data model distinguishes the following objects:

Resources: Everything described in RDF is a resource. This can be any web-page on the World Wide Web, but also things that are not on the web, like a book. Each resource is specified by a unique URI[3] that guarantees the unambiguous identification of objects.

Properties: Properties are aspects, characteristics, attributes, or relations that describe resources. Properties are also resources. Therefore they can be described by properties themselves.

Statements: A statement consists of the three parts: subject, predicate, and object. The subject is the resource about which a statement is made. The predicate is the property and the object is the value of the property. The object can be a resource itself or a literal (plain text/string).

[1] Document Type Definitions.
[2] http://www.w3.org/XML/Schema
[3] Uniform Resource Identifier

In the RDF-specification the W3C introduces a XML-based syntax to describe RDF-documents.
RDF is a restricted language and does not provide a mechanism for defining meta-information to structure information. For this reason the simple schema language RDF(S)[1] (Brickley, 2000) has been developed. RDF(S) allows one to model further primitives like:

- classes and subclass relationships which allow the definition of class hierarchies
- domains and ranges for properties to restrict the possible combinations of classes and properties and
- the type statement to declare resources as instances of classes

An RDF Schema, unlike XML Schemas or DTDs, does not describe the syntactical appearance of the RDF document. Rather it gives a semantic description of the statements made in the RDF-document (Broekstra et al., 2000).
DAML+OIL[2] is a language that is based on RDF and RDF(S) but which provides a richer vocabulary to model primitives. The Web Ontology language OWL[3] is based on DAML+OIL. It is a candidate recommendation of the W3C for web-based ontologies (underpinning the Semantic Web[4]). I will not further discuss these approaches since they are not relevant for this dissertation.

2.1.3 Visualisation of Ontologies

The previous section gave a short survey about the representation of ontologies. This chapter gives a short overview of different techniques to visualise ontologies in particular on the computer screen.

[1] Resource Description Framework Schema
[2] http://www.w3.org/TR/daml+oil-reference
[3] http://www.w3.org/TR/2003/CR-owl-features-20030818/
[4] http://www.w3.org/2001/sw/

2.1.3.1 The Tree Visualisation

A tree is a data structure that is widely used in compiler design, artificial intelligence and graphics. It consists of hierarchically organised nodes and edges which connect the nodes. An example can be seen in Figure 2.1 (a). An edge can have an attribute attached that can describes the relationship between two nodes and a 'direction' for the relationship. A tree has exactly one root node, which is the node highest in the hierarchy. Each node can be reached from the root node. All nodes can have several children but just one parent node. Consequently there is just one path to each node.
Trees are widely used to visualise the file and directory structure of file systems, the structure of XML documents, or hierarchies in companies. Because of the restricted size of the screen and the complex nature of the data represented in these trees techniques have to applied to keep a consistent visualisation. One possibility is scrolling. When the size of the tree exceeds the size of the screen, parts of the tree are not visible. With scrolling this parts can be made visible but other parts will be hidden. When the tree is very complex even this might not be enough. Then other techniques have to be applied. One approach is to expand and collapse sub trees. In this case the children of the node are hidden. Instead of seeing the whole tree just specified sub-trees are displayed.
Another way to reduce the complexity of a tree is to hide nodes with particular properties. In this case nodes with particular properties are not displayed on the screen. This approach is used in file systems to hide system files.
For ontologies, trees can be used to visualise the hierarchical structure of the concepts and relationships between the concepts. The relationship: *vacuolar membrane is-a membrane* could be visualised by displaying the node *vacuolar membrane* and the node *membrane* which are connected to each other with the directed edge *is-a*. This can be seen in the tree in Figure 2.1 (a). The arrow at the edge represents the direction of the relationship. The relationship: *vacuolar membrane* is *part-of vacuole* has to be represented in a different branch even though the concept *vacuolar membrane* is still the same concept. The reason for this is that a node can have just one parent in the tree. Here the edge connects the concept and its components. The arrow represents the direction of the relationship. This can be seen in Figure 2.1 (a).

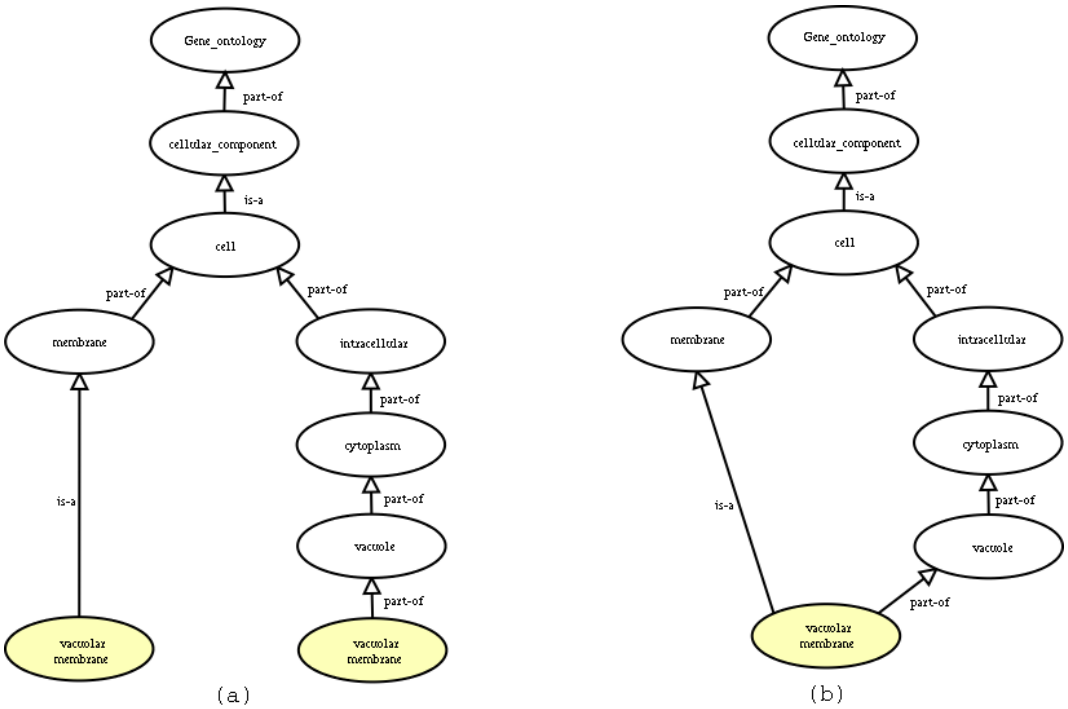


Figure 2.1: shows the (a) tree and (b) graph visualisation of concepts from the GO ontology (2.2) with the cellular-components. In (a) the tree visualisation the single concept *vacuolar membrane* has to be represented by two nodes since in a tree a node can have just one direct parent. In (b) the graph visualisation multiple parents for the concept *vacuolar membrane* is no problem.

To visualise an ontology as a tree has some disadvantages. One can see all children of a concept but not all parent nodes directly. For each parent there exists a separate path to the node. Some mechanisms have to be defined to visualise the fact that a concept has multiple parents. Another problem is that only limited information can be displayed for a concept in the tree. Usually the name of the concept is displayed or sometimes a unique identifier if this is expressive enough. All the other properties can not be displayed easily. As mentioned before, out of complexity it is not always feasible to display the whole tree of the ontology at once. Some ways have to be found to reduce complexity of the visualisation. In this case it has to be accepted that the displayed information is not complete and important facts can be missed.

2.1.3.2 The Graph Visualisation

In graphs, like in trees, information is represented via nodes and their connections. In contrast to trees, nodes in graphs can have more than just one parent node. In this case all the relatives of a node can be displayed directly. This solves one of the problems of the tree visualisation because all the hierarchical information and the relationships can be displayed at once. Graphs are widely used to represent network structures, circuit design or in planning tasks like route planner. An example of concepts and their relationships in a graph visualisation can be seen in Figure 2.1 (b).
In comparison with the tree visualisation, the visualisation and the arrangement of the nodes in graphs gets very complex quickly. Especially for large ontologies it is impossible to visualise all concepts with all the relationships on a computer screen. The edges and the nodes of the graph would overlap and make it difficult to distinguish the different concepts and their relationships. As in the tree visualisation some information has to be hidden to reduce the complexity of the problem. An example of this can be seen in section 2.1.4.2.

2.1.3.3 The Node-Based Visualisation

The node-based visualisation uses a different approach than the two approaches introduced above. By node-based visualisation is meant an approach that displays all the information of a concept at once. This means for a node all the properties and relationships can be seen. Usually this is done in a document-style manner. The navigation in the ontology can be realised via hyper-links to the relatives and properties. This is well known from HTML.
A main disadvantage here is that the hierarchical information can not be displayed in the node-based visualisation. For each node just a limited number of predecessor and successors can be displayed. An example for this approach can be seen in Figure 2.4.

2.1.4 Ontology Editors and Browser

This section gives a short overview of some ontology editors and browsers available at present. They are presented not only in order to introduce different approaches of

visualising ontologies but also to introduce other features of these applications.

2.1.4.1 Protégé-2000

Protégé-2000[1] is a platform for developers and domain experts to build conceptual domain models and knowledge bases using a graphical user interface. Figure 2.2 shows the graphical user interface of Protégé-2000.
With the plug-in facility Protégé-2000 can be extended with new plug-ins for file import and export in different formats or new visualisation techniques. The models can also be accessed from stand alone applications via the Protégé API. Protégé-2000 is

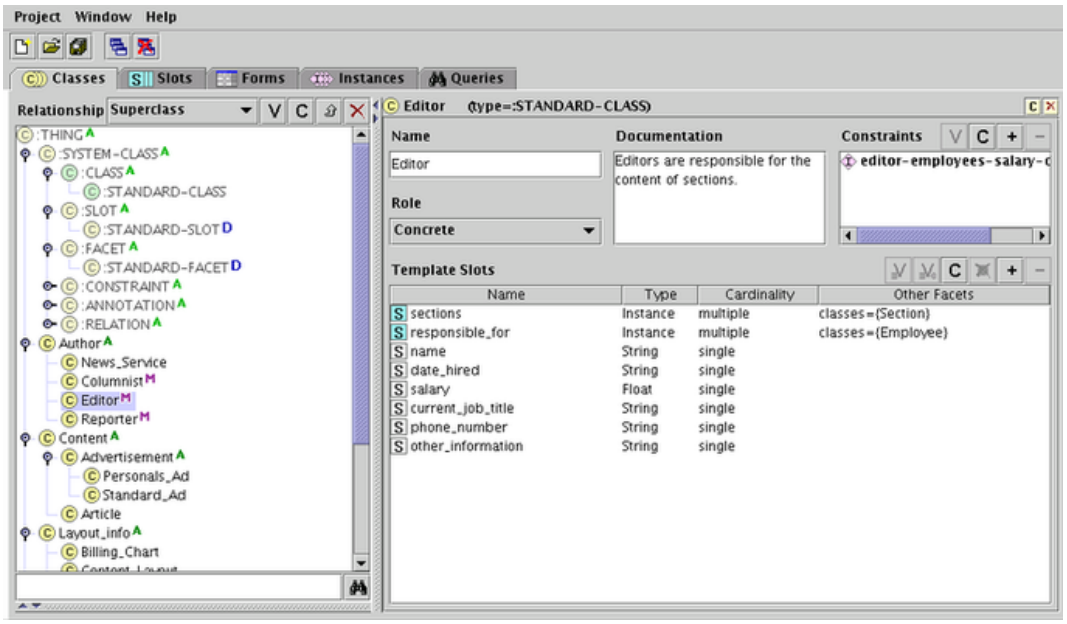


Figure 2.2: shows the *classes tab* of ontology and knowledge base editor Protégé-2000. On the left side the hierarchy is shown. On the right side the properties of the selected concepts are shown. The ontology used is the newspaper ontology provided with the tool.

typically used to model classes (domain concepts), their properties and relationships. It is used to create instances of these classes to build knowledge bases with defined semantics and logical behaviour and to ask questions of these. It uses its own data

[1] http://protege.stanford.edu/

model to represent the ontology internally. Filters are provided to import ontologies from and export to other representations languages like RDF, UML and DAML+OIL. Protégé-2000 supports multiple inheritance so that a concept can have more than just one parent. The concepts can be concrete or abstract. Like in object oriented programming instances can be created only from concrete concepts, not from abstract concepts. The properties are called *slots* and have a name and a value.
Protégé-2000 without additional plug-ins provides four different tabs that display different parts of the ontology. The *classes tab* (see Figure 2.2) displays the concepts in an inheritance hierarchy in a two dimensional tree and the properties and relationships for a selected concept in the tree. The properties and relationships of the concepts can be edited in the *slot panel*. The *instance tab* shows the concepts in a two dimensional tree view like in the *classes tab* and in addition to this the instances of the selected concept are displayed. When selecting an instance the properties of it are displayed and can be edited. In the *forms tab* the user can edit the forms presented when editing the properties of the concepts. (Knublauch, 2003)

2.1.4.2 KAON

KAON[1] is an *"ontology management infrastructure"* that provides a graphical user interface for ontology engineering. The ontology language used in KAON is an extension of RDF(S). The extension includes new relationships, and the reuse of other ontologies in ontologies and meta concepts. Concepts can be treated as instances of meta-concepts. (oim, 2002)
KAON uses *OI-model*[2] for the representation of ontologies. The *OI-model* contains the concepts of the ontology with their properties, relationships and instances. An *OI-model* can also contain other *OI-models* and therefore other ontologies. To edit and browse through ontologies a graphical user-interface is provided. It is displayed in Figure 2.3. The ontology is visualised as a graph[3]. Each node displays the name of the concept it represents. For a visible node in the graph the immediate sub-concepts, the super-concepts, the properties and the instances can be expanded individually. This

[1] http://kaon.semanticweb.org/
[2] ontology-instance model
[3] based on the graph-library TouchGraph: http://www.touchgraph.com/

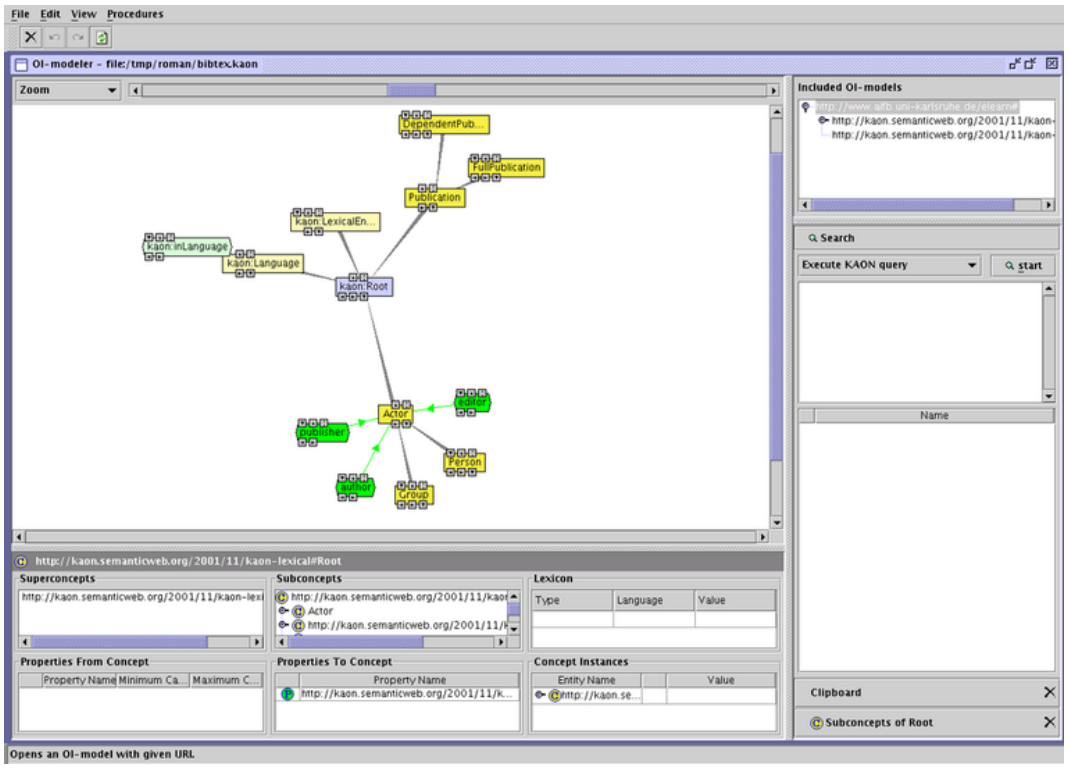


Figure 2.3: shows the KAON OIModeler (oim, 2002). The top left part shows the ontology displayed as a graph. Different colours at the nodes represent different relationships.

keeps the complexity of the graph at a manageable level. The nodes provide editing functionality. (oim, 2002)

2.1.4.3 Ontolingua

Ontolingua[1] is an environment to *"browse, create, edit, modify and use ontologies"* that can be accessed via the World Wide Web with any browser. The underlying representation of the ontology is realised via an extended version of KIF[2]. (Rice et al., 1996) The tool provides a *"hierarchical class browser"* to display the ontology and browse through it. It shows the concepts in a two dimensional tree visualisation. In addition the user can browse through the ontology by a node-based visualisation (shown

[1] http://www.ksl.stanford.edu/software/ontolingua/
[2] Knowledge Interchange: http://logic.stanford.edu/kif/kif.html

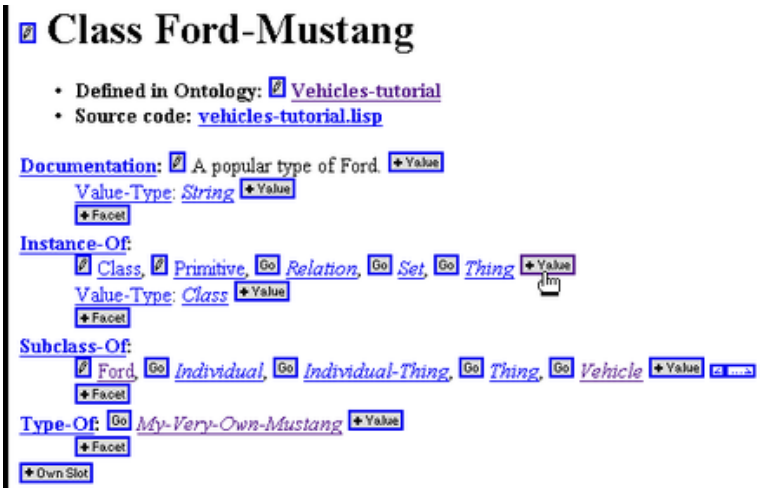


Figure 2.4: shows the class definition of the concept Ford-Mustang in the ontology editor of ontolingua. It was taken from the ontolingua tutorial (ont, 1997).

in Figure 2.4) and reach the related concepts by hyper-links. (Rice et al., 1996)

2.2 Ontologies in Biology

The Gene Ontology Consortium was formed to provide a structured vocabulary to describe elements from molecular biology that are shared across different life forms (Consortium, 2001). The vocabulary is called Gene Ontology (GO). It describes gene products *"in terms of their associated biological processes, cellular components and molecular functions"* in a species-independent way. The terms in the vocabulary are used as annotations for genes in databases for a wide variety of species. The ontologies for these species can be found on the OBO[1]-homepage.

The Gene Ontology and the Open Biological Ontologies are not the only resources on ontologies for biology. Other approaches like the Sequence Ontology[2] (SO) or BioCyc[3] are not further described since they are not relevant for the project.

[1]Open Biological Ontologies: http://obo.sourceforge.net/
[2]http://song.sourceforge.net/
[3]http://www.biocyc.org/

2.2.1 Representation of Biological Ontologies

In the Gene Ontology concepts are referred to as 'terms'. Each term has a unique identifier. The reason for this is that the name of the term may not be unique. These terms are represented in form of an directed acyclic graph (DAG) sometimes referred to as a network. This allows a GO term to have more than one parent term, as described in section 2.1.3.2. The permitted relationships between terms are *is-a*– and *part-of*– relationships. The meanings of these relationships are as described in Section 2.1. An additional used relationship type in ontologies of species is *lineage*. This relationship can refer to the developmental stage of an organism. This can be for instance different embryonic stage in the development of embryos.

2.2.2 The GO File Formats

To store the ontologies the GO Consortium developed the GO flat-file format. The files are pure text files. Each line represents a GO term. Indentation is used to represent parent-child relationships. In front of each term a special symbol indicates the relationship to the parent term. A fragment of a GO flat-file can be seen in Appendix C.
In recent years the consortium has developed an XML version to store the ontologies. This format uses constructs from RDF to formulate the relationships between terms, and to specify the unique identifier of the GO terms. A fragment of the XML version of GO can be seen in Appendix D.
A more detailed description of the file formats is given on the web-page of the GO Consortium.

2.2.3 DAG-Edit

DAG-Edit[1] is an ontology editor and browser specially developed to edit the GO file formats. The editor displays the hierarchical structure of the ontology on the left side in a two dimensional tree. Different relationship types are identified by different icons. Multiple parent relationships are not directly visible in the tree. On the right side

[1] http://www.geneontology.org/doc/GO.tools.html

additional information about the currently selected term is displayed. The environment is displayed in Figure 2.5.

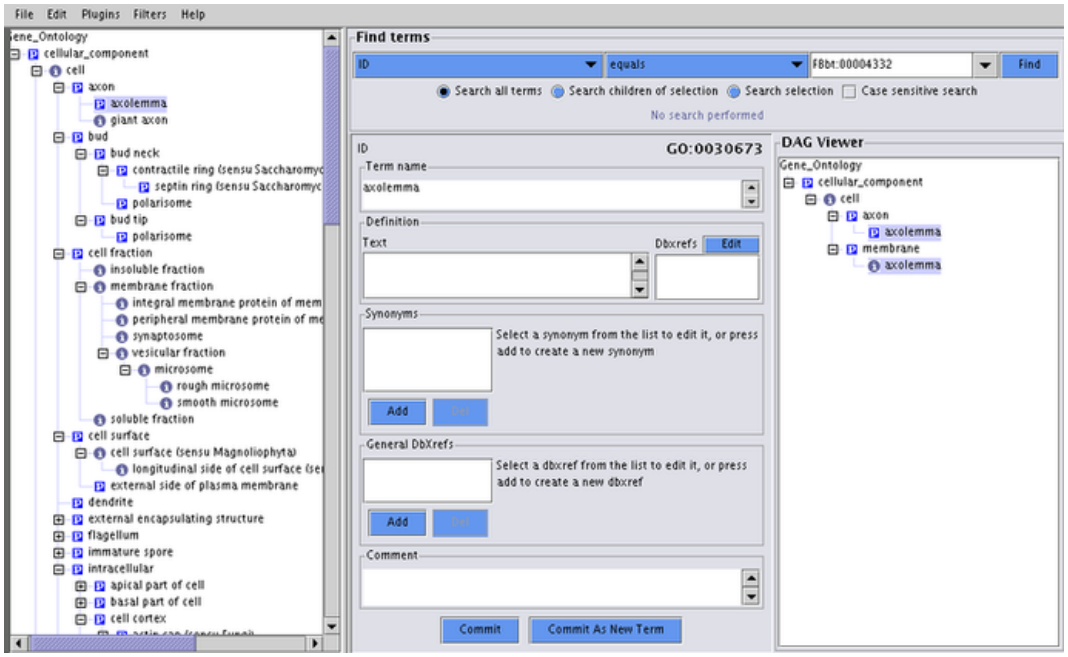


Figure 2.5: shows the Gene_Ontology in the DAG-Editor. On the left side the ontology is displayed in a tree visualisation. On the right side the properties of the selected GO term are displayed.

2.3 Software Evaluation

The aim of this section is to introduce the evaluation techniques used to perform software evaluation. First some of the terms used here are clarified and then some methods for evaluation are introduced.

2.3.1 What is software evaluation

In (Preece et al., 1994) the authors define evaluation as follows:

> *"Evaluation is concerned with gathering data about usability of a design or product by a specified group of users for a particular activity within a specified environment or work context."*

The definition says that evaluation is done by a *specified group*. Not just every one should attend evaluations. Usually a product or design is made to fulfil the requirements of specific group of users. They have a specific vocabulary that should flow into the design or product, and specific ways to do their tasks. The product should be adapted to these requirements. And therefore, the evaluation should be done with participants of this group to see how the product can support these people in their work.
The definition also says that the evaluation is done for a *particular activity*. The product is usually meant to support the users in their work. The tasks done during the evaluation should be similar to the tasks user would actually use the product for.
It is also important to take the *environment* of the evaluation into account. It can make a difference to do the evaluation in an completely unfamiliar environment for the user, for instance.
In the definition the authors also say that the evaluation is about measuring the *"usability of a design or product"*. The international standard ISO 9241 (Part 11) defines usability as follows:

> *"Usability of a product is the extent to which the product can be used by specific users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specific context of use."* (ISO, 1998)

ISO 9241 (Part 11) defines *effectiveness* as *"the accuracy and completeness with which users achieve specified goals"*. This covers the quality of the solution and the errors made while using the product. *Efficiency* is defined as *"the resources expended in relation to the accuracy and completeness with which users achieve goals achieved"*. This may include the time spent to do a task, the time spent to learn to use the product, and the physical effort spent. And *satisfaction* is defined as *"the comfort and acceptability of use"*, which are the subjective experiences of the user of the product.
Nielsen (Nielson, 1993) defines usability on the basis of:

learnability: The system should be easy to learn so that the user can rapidly start getting some work done with the system.

efficiency: The system should be efficient to use, so that once the user has learnt the system, a high level of productivity is possible.

memorability: The system should be easy to remember, so that the casual user is able to return to the system after some period not having used it, without having to learn everything all over again.

errors: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur.

satisfaction: The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

2.3.2 What are the Goals of Evaluation?

In (Preece et al., 1994) the authors distinguish between two kinds of evaluation. *Summative evaluation* is a goal-oriented evaluation. It judges the finished product or design on the basis of previously defined criteria. These results can be used to compare the quality of different products. *Formative evaluation* is concerned with improving the product or design. Qualitative data are gathered to detect usability problems and quantitative data are gathered to test the progress of the realisation of the usability goals.
There are different reasons why an evaluation is done. Typical questions that may be answered with the evaluation are:

Which one is better? Alternative products or designs are compared. Reasons for this can be to choose the best solution from the alternatives.

How good is it? The quality of an product or design is measured. This can be used to test if it conforms to the usability goals or for certification.

Why is it bad? The aim is to find weaknesses of the product or design to generate suggestions for further development.

The first two goals can be seen as *summative evaluations*, the third one as *formative evaluation*.

2.3.3 Evaluation Methods

Evaluation methods are tools that can be applied to measure if a product or design achieves the given goal. These methods can be classified in various ways. A very common way is to classify them into *user testing* and *usability inspection* (see also Nielson & Mack (1994)).
In the *user testing* representative users participate in the test. These methods are applied to find problems users might experience. *Usability inspection*, on the other hand involves methods that can be applied without the involvement of users. Important considerations in choosing usability methods is always the time spent to do the evaluation and analyse the data and the cost for equipment and material. There are numerous introductions available that cover a wide range of evaluation methods (e.g. Preece et al. (1994), Dix et al. (1998), Wixon & Willson (1997) and Riihiaho (2000)).
To do the evaluation the users should do a predefined task to have comparable results from the evaluation with different users. The task should be representative of the tasks the product was intended for, and cover most of the important functionality of the product. (Nielson, 1993)
In the following sections exclusively methods with user involvement are described since they are relevant for this thesis. The methods introduced here are those relevant for this dissertation.

2.3.3.1 Observing

Usually users are observed to see how they do a specified task, to see how they behave, what problems they experience and what difficulties they have. Like in all evaluation methods, it is important to think about what shall be the purpose of the observation. It might be of interest to know how a user does a specific task, in other cases the observer is more interested to know how the user would use the product in their own work environment. Another point that has to be considered is what shall be observed. An observer might not just be interested in observing the screen, but also in observing the user's behaviour or even the use of keyboard and mouse. The observer then needs to observe not just the screen but also the body and face of the user and the keyboard as well. (Preece et al., 1994)

Depending on the desired outcome of the evaluation and the available equipment different observation methods can be applied. During *direct observation* the observer is next to the user and takes notes about interesting behaviour of the user. One major problem with this method is that the user could feel disturbed. This could influence the results of the evaluation. Another major problem is that the observer cannot deal with the amount of information and might leave out some important facts. On the other hand the method is a immediate way of recording user behaviour and therefore often applied. (Preece et al., 1994)
Indirect observation/video recording is a way in which users are observed without a direct observer. Usually cameras are positioned in order to record the screen, the face, the body and/or the keyboard. Often video recording is synchronised with software logging (see section 2.3.3.3) to get contextual information. A major trade-off of video recording is the time used to analyse the data. (Preece et al., 1994)

2.3.3.2 Thinking-Aloud Protocol

While applying the *thinking-aloud protocol* method the user is asked to articulate her thoughts, feelings and opinion while she is interacting with the product. The evaluator records the comments of the user. This can be done with pencil and paper, video or audio recording. During the evaluation the evaluator is not supposed to guide the user. (Preece et al., 1994)
With this method the evaluator gets qualitative data about the attitude of the participants. These are data about how they understand the product and how they interpret the product (Kato, 1986). In the *thinking-aloud protocol* session many problems are shown the user would not remember after the evaluation when filling out a questionnaire (section 2.3.3.4), for instance. In combination with software logging (see section 2.3.3.3) the evaluator has the chance to interpret the result using contextual information.
The *thinking-aloud method* is an obtrusive evaluation method. The users' articulation of her thoughts can affect the performance of processing the task because it is unnatural to most of the users and might make the task feel harder. Sometimes user think they have to fulfil some expectations. This can influence the comments they make and

therefore the results.
The collection of data is a qualitative evaluation technique and can be done just for a small group of users. Nielsen suggests that a small group of five users is sufficient to find the major problems (Nielsen, 1994). But the evaluation of the results requires interpretation from the evaluator.
This method is one of the most used methods in the evaluation process. Some reasons for this are: it is straightforward to apply, can be done with just a few participants, and gives fast results. (Nielsen et al., 2002) For additional information see also Boren & Ramey (2000) and Lewis (1982).

2.3.3.3 Software Logging

Software logging is a process where the interactions between the user, the application and the operating system are logged into a file while participants use a software product. These methods are applied in order to get data about patterns of system usage, the time taken to perform tasks, the error rates and the frequency with which the user accesses the online help. In (Preece et al., 1994) the authors describe two different approaches of *software logging*. In the first approach the key-strokes and mouse-clicks are logged together with time stamps. In the second approach the whole interaction with the system is recorded in real time. The evaluator can see the users' interaction with the system in real-time.
Usually these methods are combined with other evaluation methods like video recording (see section 2.3.3.1) or thinking-aloud protocols (see section 2.3.3.2) to add contextual information to these methods.
One of the main advantages of *software logging* is that it is unobtrusive. The user is not disturbed by the questions of the evaluator or by verbalising her thoughts like in the thinking-aloud method (see section 2.3.3.2). However for ethical reasons the user should be informed that she is being monitored. Another advantage is that the evaluator does not have to be present during the test. The analysis of these software logs can be partly automated. (Preece et al., 1994)
The main disadvantage of this approach is that it gathers a large amount of data that have to be analysed. This can be very time consuming.

A good introduction and discussion about how to extract *"usability information from user interface events"* can be found in (Hilbert & Redmiles, 2000)

2.3.3.4 Questionnaires

Questionnaires are methods to test the users' opinion and attitude about the product (Preece et al., 1994). Usually questionnaires are given in paper form. In recent years web forms have been used more frequently for questionnaires. This provides a better tool to reach a larger number of people.

In (Preece et al., 1994) the authors distinguish between two main types of questions. *Closed questions* are questions where the user chooses the answer from a set of different possible answers. Usually a fixed number of answers in a rating scale. For the scale usually values like *yes, no* and sometimes *strongly agree* or values from *agree, neutral* to *don't agree, strongly don't agree* are used. An example can be seen in Figure 2.6. The results can be easily analysed with statistical methods since the answers are restricted and usually complete. A problem with this type of questionnaires is, that the answers are restricted and predefined. The user can not express her thoughts in her own words.

With *open questions* on the other hand the user can give a free response (see Figure 2.6). She can articulate her own thoughts and attitudes without being restricted. Sometimes these answers raise issues the evaluator had not thought of before. The main problem with these type of questions is that they are time consuming to analyse. Because the answer leave space for the users opinion the answers can be very long and complex. If they are filled out by hand it might be also difficult to read the answers.

The questionnaires should be answered shortly after the evaluation. The more time that passes between evaluation and answering the questionnaire the more important facts the user can forget. In this case video recording (see section 2.3.3.1) might helpful the evaluator to fill the gaps.

Often the evaluator is not present when the user fills in the questionnaire. The participant might not know what to answer or does not completely understand the question. The challenge in preparing the questionnaire is to make the question as unambiguous as possible and as easy as possible. It is better to restrict the number of questions;

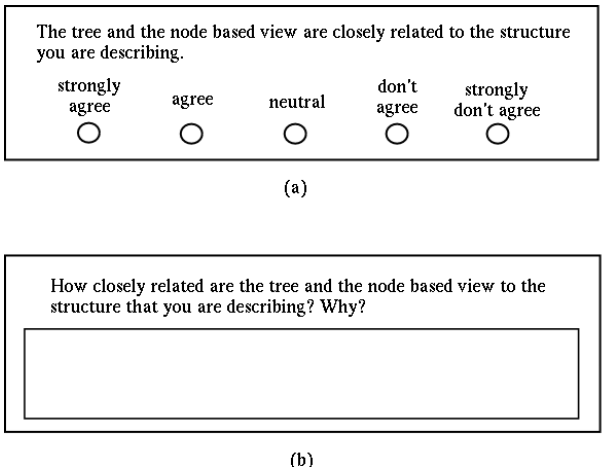


Figure 2.6: shows and example of a *closed* question in (a) and an example of an open question in (b).

users might get bored and lose interest in filling out the questionnaire when there are too many questions.

For further reading see Oppenheim (1992), Converse & Presser (1986),Sudman & Bradburn (1982) and Human Factors Research Group (2000).

Chapter 3

COBrA – Ontology Browser

This chapter will summarise the work I have done. At the beginning the concepts, the design and the implementation issues of the implemented graphical user interface COBrA (Concept Ontology Browser for Anatomy) are summarised. A screen-shot of the application can be seen in Figure 3.1
The purpose of the program is to provide a tool for Developmental Biologists to easily explore two ontologies in Gene Ontology format about anatomies of different species and make annotations about relationships between the anatomies. Therefore the ontologies had to be presented to the user in visual form. The functionality to browse through the ontology and to make the annotations had to be provided. In addition the ability to edit the ontologies was desired.

3.1 Background Data Structure

This section describes the underlying data structure for the ontologies. The main concern was to cover the expressiveness of the Gene Ontology file formats and provide a data structure that describes the ontology as closely as possible. At first the idea behind the representation of the ontology is described, then relevant and important design issues are presented and at the end the most important issues in the realisation are discussed.

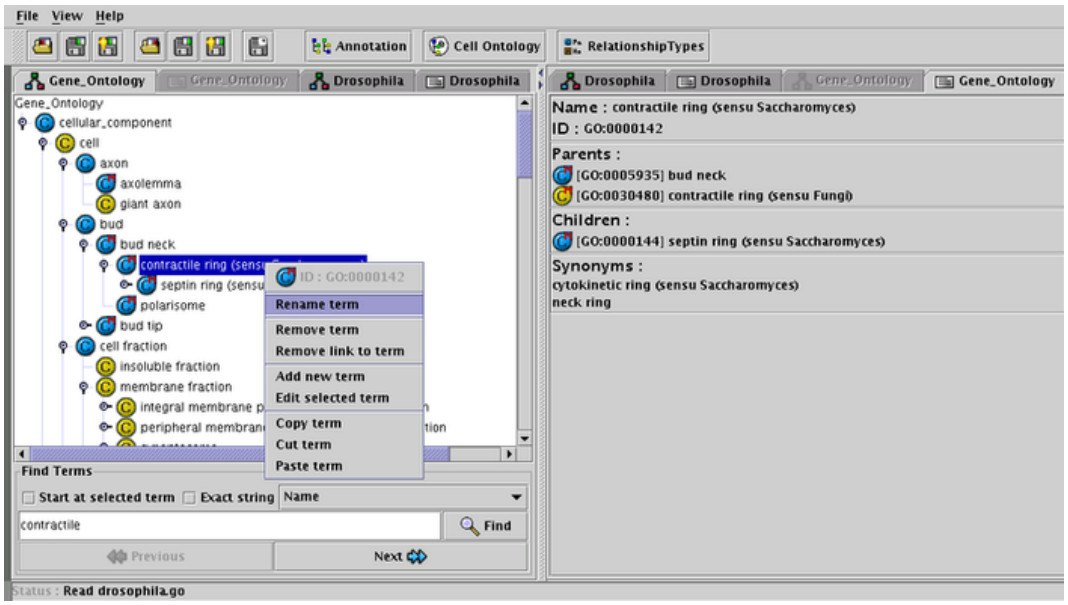


Figure 3.1: shows the COBrA tool. On the left side the hierarchical structure of the Gene_Ontology is displayed in the tree visualisation. The menu of the selected GO term is displayed in the tree. The search facility can be seen below. The right side of the application shows the node based visualisation of the selected GO term.

3.1.1 Concept

As described in section 2.2.1 the ontology is represented in form of a directed acyclic graph (DAG). The nodes of this graph are the GO terms and the edges between two nodes in the graph represent the relationship between these GO terms. Each GO term has properties in addition to the relationships. The graph has exactly one root node, which represents the domain of the ontology.

The two different file formats, flat-file format and XML, briefly introduced in section 2.2.2, provide different levels of details in representing the GO terms. Since both formats will be supported this has to be considered in the design. (A fragment of the flat file version of GO can be seen in Appendix C and a fragment of the XML version of GO can be seen in Appendix D.)

3.1.2 Design

To represent a GO term in memory a class had to be designed that contains all properties of the GO term. Beside the name of the GO term, GO id and other properties, the relationships to and from other GO terms had to be represented. Relationships to other GO terms can be seen as child relationships or parent relationships depending on the direction of the relationship. Parent and children relationships are properties of the GO term with an attached relationship type.
A relationship property can therefore be represented with a reference to a GO term and a relationship type. Depending on the relationship property (parent or child) the direction of the relationship is clear.
The ontology can then be represented as a class with the root term as attribute. From the root term each GO term in the ontology can be reached. In addition the ontology has a version number to keep track of different versions, and some additional attributes for further identification.

3.1.3 Realisation Issues

The structure of the directed acyclic graph is realised with double-linked lists. Each link consists of a reference to the relative GO term and the relationship type from or to the relative. Through the reference the relative can be reached. This provides the basic functionality to browse through the ontology and visualise the tree. The class for the ontology contains a reference to the root term. From this term all GO terms can be reached.
Figure 3.2 shows UML-diagram of the relevant classes for the data model of COBrA. Not all classes are displayed and not all attributes in the classes are displayed. This has been done to reduce the complexity of the diagram. The diagram shows the class *GoOntology* which has among other attributes the attribute *rootTerm* which is a *GoTerm*. A *GoTerm* can have *children* and *parents* which are stored as instances of the class *GoRelationship* in a *GoRelationshipVector*. A *GoRelationship* consists of a *relative*, which is the relative-*GoTerm* and a *GoRelationshipType* which specifies the relationship between the GO term and the *relative*. The *GoRelationshipType* contains

the relationship name, the icon displayed for the relationship type and the icon that is displayed when the GO term has multiple parents.

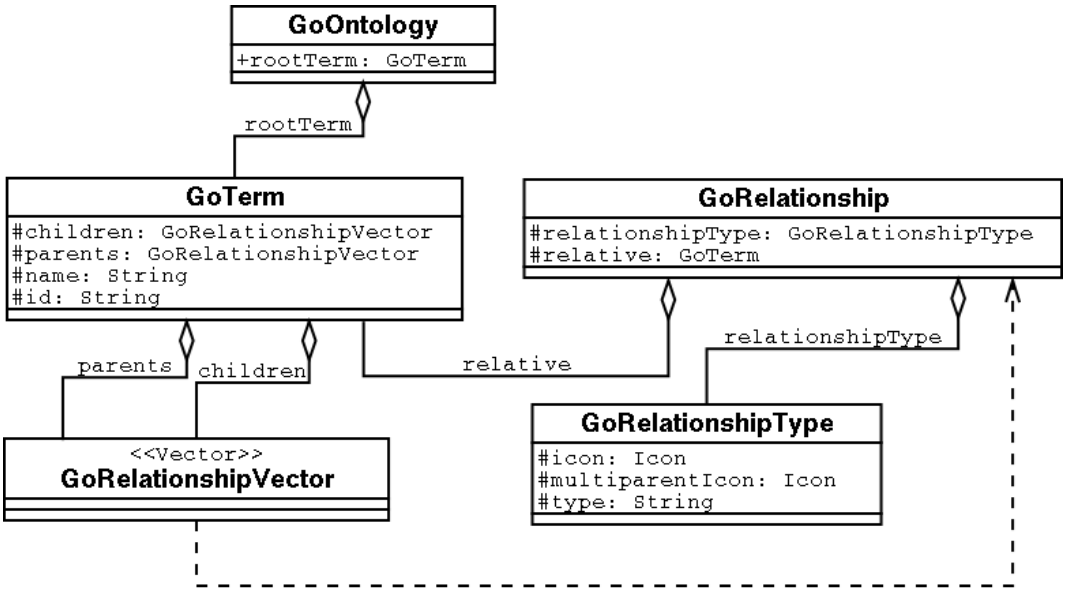


Figure 3.2: shows the UML-diagram of the data-model for COBrA. Just the attributes relevant for the relationships among the classes are displayed and no methods are shown to keep the diagram simple.

3.2 Visualisation of the Ontology

In this section I describe the different approaches I considered to visualise ontologies on the screen. Then I will justify my choices and describe the design and realisation issues of these approaches in more detail. A good discussion about visualisation in software is given in (Petre et al., 1997).

Following the suggestion in (Preece et al., 1994), I examined other programs with similar aims in order to get ideas about possible approaches and compare different approaches to the visualisation of ontologies. In section 2.1.4 and section 2.2.3 I introduced different ontology editors and browsers with different visualisation techniques. In *Protégé-2000* and *DAG Edit* the ontology is displayed as a tree. Other approaches can be integrated in Protégé with the plug-in functionality. *KAON* provides a graph

visualisation in the *OIModeler*, and in *Ontolingua* the ontology is visualised in a node-based visualisation with hyper-links to the related concepts. The visualisation techniques and some of their advantages and disadvantages have been discussed in section 2.1.3.

In (Storey et al., 2001) and in (Frank van Harmelen et al., 2001) two different approaches to the visualisation of ontologies are introduced. In (Storey et al., 2001) a *"nested interchangeable view"* is used to explore the ontology at different abstraction levels. Concepts can be zoomed in and properties can be displayed and edited. Through hyper-links other concepts can be explored. In the approach in (Frank van Harmelen et al., 2001) concepts that are *"semantically close"* are grouped together and visualised in *"cluster maps"* (Christiaan Fluit, 2002). These *"cluster maps"* are connected with edges like in the graph visualisation. Both of these approaches are in my opinion not feasible for the project. First of all the approaches need extensive pre-processing for the visualisation. The implementation of this is very time consuming. And second the programming primitives for the visualisation and the arrangement of the visualised objects needs to be implemented. In the available time for the project I thought I can not implement either of these visualisation techniques. For this reasons I did not implement them.

For a similar reason I also decided not to implement the graph visualisation. For this approach nodes have to be displayed, edges have to be drawn and everything has to be efficiently arranged on the display. As it can be seen in Figure 2.3 in section 2.1.4.2 the number of nodes that can be displayed at the same time is very limited.

The reasons that led to the decision to implement the tree visualisation are:

1. The tree visualisation for ontologies is widely accepted and often used as can be seen in Protégé and DAG-Edit. The fact that the user can employ well known methods to do her work and does not have to adopt to the tool is a vital argument to decide for the tree visualisation. It would shorten the time needed to learn to use the application. (Nielson, 1993)

2. The JAVA API[1] provides the basics for tree visualisation with the *JTree*.

[1] http://java.sun.com/

In the tree visualisation limited information can be visualised for each GO term. To provide the user with additional information like properties of the GO terms, I also decided to implement a node based approach to visualise the ontology, in addition to the tree visualisation. It should be easy for people who are familiar with hyperlink technologies like HTML to make the transition to the node based visualisation.

3.2.1 Tree Visualisation

This chapter introduces important issues that raised while designing and implementing the tree visualisation. On the left side of Figure 3.1 the tree visualisation is displayed.

3.2.1.1 Concept

As mentioned in section 2.1.3.1, the tree provides a hierarchical visualisation of the ontology. One aim was to provide basic editing functionality for the ontology in the tree. It should be possible to add new GO terms to the ontology, edit the properties of existing GO terms and edit the relationships between the terms.

3.2.1.2 Design and Realisation Issues

To display the hierarchical structure of the ontology the directed acyclic graph has to be represented as a tree. Each node in the tree represents a GO term. Multiple parent relationships can not be displayed directly in the tree. For each parent there must be a separate path to the GO term. This approach is similar to the approaches used in Protégé and DAG-Edit. The root node in the tree is the domain name. All other GO terms are direct or indirect children of this node. For each node the name of the GO term it represents is displayed. An icon at a node represents its relationship to its parent. Different relationships are represented by different icons. When a GO term has multiple parent relationships the icon at the node that represents the GO term contains additional information that indicates this. Since the name of the GO term is not unique the GO id has to be displayed as well. To avoid displaying overly complex nodes, I decided to provide the GO id via tool tips.
Figure 3.3 shows the UML-diagram of the classes that are related to the tree visu-

alisation. Not all classes and attributes are included to reduce the complexity of the diagram. The tree visualisation in the class *OntologyTreePanel* consists of the tree that is represented by the class *DragAndDropTree* and the class *SearchPanel*. The GO terms and their relationship type to the direct parent are represented as *GoTermNodes* in the tree. It encapsulates the class *GoRelationship* (see also Figure 3.2). The tree displays the name and the icon that represents the relationship type between the GO term and its parent.

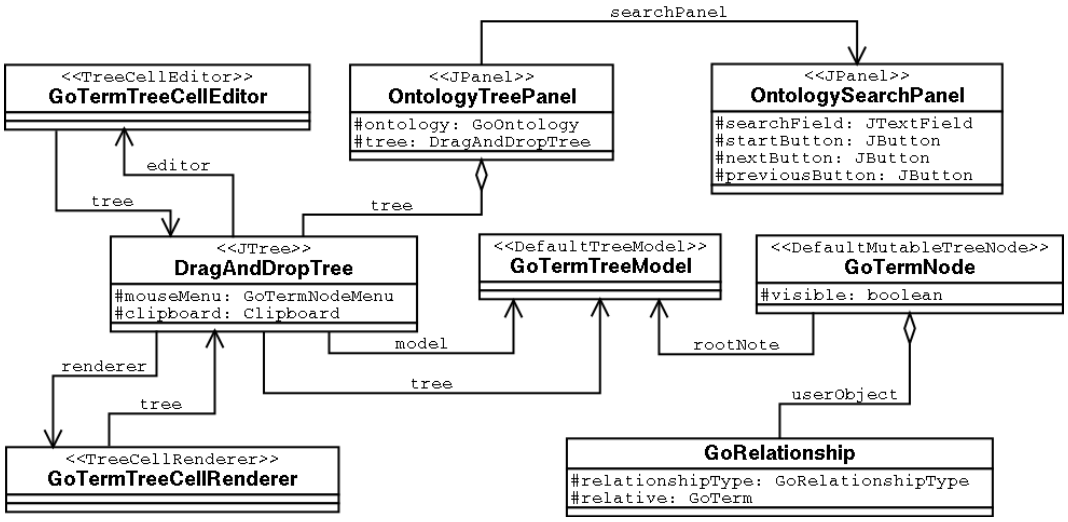


Figure 3.3: shows the UML diagram of classes related to the tree visualisation. Not all classes and attributes are displayed in this diagram to reduce the complexity.

3.2.2 Node Based Visualisation

This section introduces the concept, design and realisation issues for the node based visualisation of the ontology. The ideas for this approach come from HTML and from the approach in Ontolingua (section 2.1.4.3). The right hand side of Figure 3.1 shows the node based visualisation in COBrA.

3.2.2.1 Concept

In the node based visualisation the main attention is drawn to the properties of one selected concept in the ontology and not the hierarchical structure of the ontology. The

hierarchical information in this visualisation technique is limited. In order to reach other concepts in the ontology a browsing mechanism is desirable.

3.2.2.2 Design

To visualise the information of a GO term in the node-based visualisation a document style is used. It is displayed on the right hand side in Figure 3.1. The information is represented as a document page divided into different sections. Each section displays different properties of the GO term. On top of each section the name of the property is displayed. Below the property name the values are displayed. If a property has no values the section for this property is not displayed in the page.
In the sections for the parent and children properties the relatives are indicated by hyper-links. The ontology can be explored through the hyper-links. By clicking on one of the links to a relative the page for the relative will be created and displayed instead of the page of the GO term that was displayed before. This is a very similar concept to the hyper-links in HTML.

3.2.2.3 Realisation Issues

For the realisation I considered two different approaches. The first uses the HTML functionality of Java. HTML provides an easy way to display the properties of a GO term and realise the hyper-links to the related GO terms. For the second approach the basic Swing[1] components of in the Java API are used. I decided for the second approach in order to keep the design uniform with the JAVA Swing design. The hyper links are realised though buttons.
Figure 3.4 shows the UML-diagram of the node-based visualisation. Not all attributes of the classes and not all classes are displayed to reduce the complexity of the diagram. The *OntologyNodeBasedPanel* is created with the actual selected *GoTerm* in the *GoOntology*. This is the same GO term that is selected in the tree visualisation. The properties of the GO term are displayed in panel which are arranged underneath each other. The *HeadPanel* displays the name and the GO id of the actual selected GO term and, if available, the description. If the GO term has children they are displayed

[1] http://java.sun.com/docs/books/tutorial/uiswing/index.html

in the *childrenPanel* which is an instance of *RelativePanel*. The parents are displayed in the *parentPanel* which is also an instance of *RelativePanel*. The displays of the children and parents are realised with *JButton* which provide a link to the relative. If the *GoTerm* contains additional properties like *synonyms* they are also displayed in a panel.

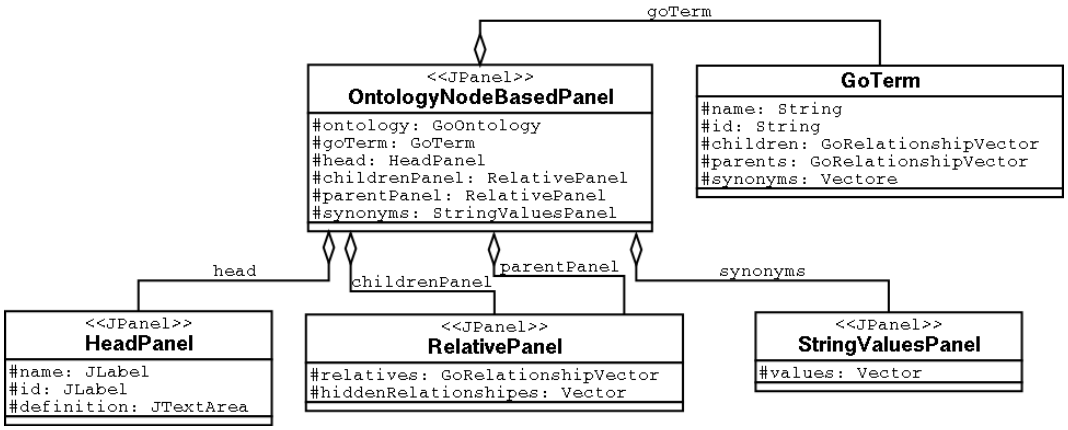


Figure 3.4: shows the UML-diagram of the node-based visualisation for COBrA. Just the attributes relevant for the relationships among the classes are displayed and no methods to keep the diagram simple.

3.2.3 Connection between Tree and Node Based Visualisation

In order to keep consistency in the application the different visualisation techniques have to be synchronised. When a GO term is selected in the tree visualisation, the same GO term should be visualised in the node based visualisation. Otherwise the user has to keep track of different concept in the same ontology. This would also complicate the process of making the annotations (section 3.4). It would be not clear which GO term is chosen for the link.

To prevent these problems and provide consistency, I connected both visualisation techniques. To do this I implemented a listener that listens for changes in the alternative visualisation. A listener is a class that looks for particular interface events. When a GO term is chosen in one of the alternatives it is also selected in the other visualisation.

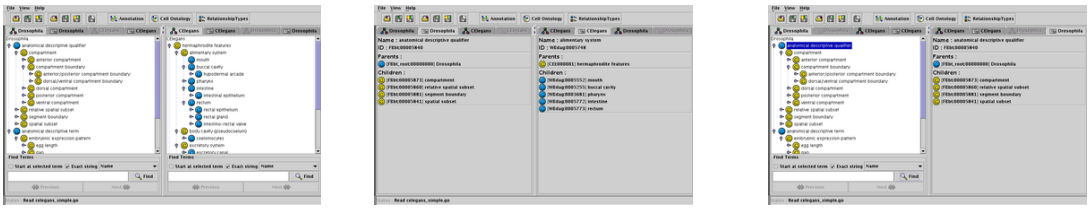
3.2.4 Selecting the Visualisation Type

As described in the previews sections, I implemented two different techniques to visualise the ontology with different methodologies. The user should be able to choose which visualisations of the ontology she wants to see. For this I considered different scenarios in which the user uses the different views.

In the first scenario the user chooses the tree or node-based visualisation to browse and edit two ontologies and create links in the ontologies (see section 3.4). The two alternatives are displayed in Figure 3.5 (a) and (b). This is what the authors in (Petre et al., 1997) call *"the simple case"* of multiple representation. The different visualisations are used separately. In this case the user would have either of the visualisations for the two different ontologies on both sides. The user should be able to select the visualisation she prefers on either side.

In the second scenario the user wants to explore one of the ontologies in more detail and wants to see the two different visualisations of the ontology at the same time (As in Figure 3.1). This is displayed in Figure 3.5 (c). In this case the user should be able to choose one visualisation (e.g. the tree visualisation) of an ontology on the left side and the other visualisation (in this case the node based visualisation) for the same ontology on the right side. On one side the user would see the hierarchical information of the ontology in the tree, and on the other side enhanced information about the selected GO term. The authors in (Petre et al., 1997) call this *"bridging representation"* where the two representations support each other.

The application provides four different tabs on either side to select the two different visualisations for the two ontologies. The domain names of the ontologies are displayed in the tabs to distinguish the two ontologies. Different icons are displayed in the tabs to distinguish the different visualisations when they are hidden from view. To avoid inconsistency in displaying the ontologies the visualisation of an ontology that is displayed can not be displayed on the other side.



(a) (b) (c)

Figure 3.5: shows different scenarios COBrA can be used for. In (a) the tree views of both of the two ontologies are displayed. In (b) the node-based views of both of the two ontologies are displayed. In (c) the tree and node-based view of one of the ontologies are displayed.

3.2.5 Reducing the Visible Complexity of the Ontology

As mentioned in section 2.1.3.1 the tree visualisation can get very complex. One solution discussed in section 2.1.3.1 hides particular GO terms. In the case of the ontology this can be applied for particular relationship types. All GO terms that have a particular relationship to its parent can be hidden. This can be applied for the tree visualisation as well as for the node based visualisation.

A problem this raises here is that GO terms with the relationship type that shall be hidden can children that have other relationships to the term. In this case the term can not be hidden. Otherwise the logical structure of the ontology gets lost. In section 2.1.3.1 Figure 2.1 shows different relationships between GO terms. When hiding the relationship *part-of* the GO term *cellular_component* can not be hidden since it has the child *cell* that is related to the GO term with an *is-a*–relationship. The same applies for the GO term *membrane* and *vacuolar membrane*.

This approach in implemented in COBrA. The user can choose which relationships she wants to see. If some relationships are not relevant for the task the user is doing she can hide them.

3.3 Editing the Ontology

This section gives a short overview of the approaches that have been implemented to edit the ontologies.

3.3.1 Menu Driven Editing in the Tree

To provide basic editing functionality for the ontology a menu is provided for each node. The menu displays the GO id of the GO term that is selected and the icon for the relationship to its direct parent in the tree. This information is provided to always keep track of the selected node. The menu provides items to rename the selected GO term, remove the GO term completely from the ontology and to remove the relationship link to its direct parent. When renaming a term the name is checked and if the name already exists in the ontology, a warning is displayed. Since the name of a GO term need not be unique the user can still change the name. When a GO term is removed from the ontology, all its relationship links are removed. When the term has children, the user can choose to remove the links to the children, add the children to a new parent or cancel the action. When a link is removed from its direct parent in the tree and this is the only parent the GO term had, the term is removed from the ontology.
In addition to the editing functionality mentioned above, a new GO term can be added as a child of the selected GO term or the selected GO term can be edited. For this a dialogue is displayed, where the user can edit the different properties of the GO term. When copying a GO term and then paste it to another GO term in the tree a new relationship link from the selected GO term to the other GO term will be created and the other GO term will become a parent of the copied one. Cutting a GO term and pasting it to another GO term will remove the link from the direct parent of the cut GO term and create a new link to the GO term where the GO term was pasted.

3.3.2 Direct Editing in the Tree

In recent years drag-and-drop techniques have been extensively used in graphical user interfaces. In the tree visualisation drag-and-drop can be used to edit links between GO terms. In general there are two different possibilities to edit nodes with drag-and-drop

in a tree. The first possibility is to remove the node from the actual parent and then add it to a new parent and the second is to just add an existing node to a new parent. In the ontology the first approach would mean deleting the link between a GO term and the parent that is the direct parent in the tree and adding a new link between the GO term and its new parent. The second approach would mean adding a new link from the GO term to a new parent. The link to the other parents would remain. For the drag-and-drop functionality I adopted an existing approach that I found in the Java Technology Forums[1]. This approach provides a *JTree* with basic drag-and-drop functionality. This does not include the functionality for ontology editing like updating the underlying data structure. I had to make this adoptions for my approach.

3.4 Making Homology Links

The main reason for which COBrA was developed was to make *homology* links between two tissues of different species. In the following section I first describe the concept of creating a homology link and then I describe the design and implementation issues.

3.4.1 Concept

The Virtual Paleobotany Laboratory Glossary[2] defines homology as: *"likeness and correspondence in structure between parts of different organisms, due to common ancestry of the organisms"* and analogy is defined as: *"correspondence in function between anatomical parts of different structure and origin; analogous"*.
Homology links are relationships between two tissues that appear in different species that are related to each other. The relation is specified by a cell type. In order to make a homology link the tissues of the two different species and a cell type that relates the two tissues have to be selected.

[1] http://forum.java.sun.com/thread.jsp?forum=57&thread=416442
[2] http://www.ucmp.berkeley.edu/IB181/VPL/Glossary.html

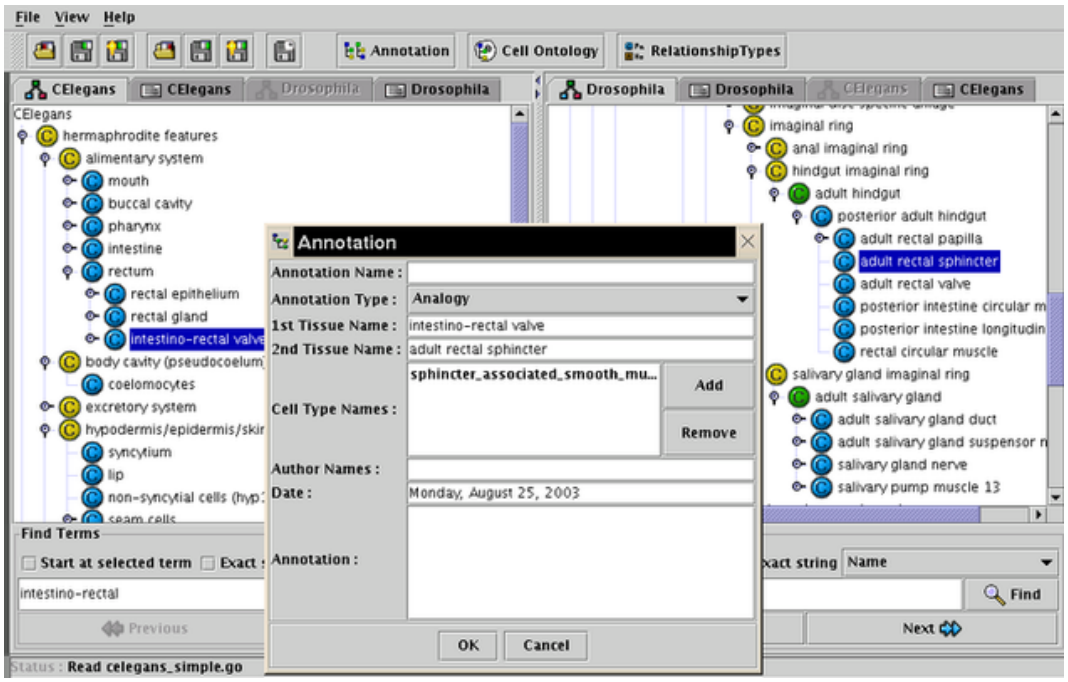


Figure 3.6: shows the COBrA tool. In each of the two tree visualisations of the ontologies a GO term is selected. These GO terms are tissues for annotation. In front of the application the annotation dialogue is open. A cell type was chosen for the analogy.

3.4.2 Design

To store the homology links (hereon referred to as annotations) different approaches can be used. One possibility is to store the annotations in an ontology similar to the biological ontologies. The advantage of this approach is that the annotations can be processed like all the other terms in the biological ontologies. The user selects a tissue on either ontology. Then she has to specify one or more cell-types that relates the two tissues. To store the annotations in the ontology as a GO term a name has to be specified and a GO id. The GO id is assigned by the application. This also prevents the user from assigning an existing id. The user is also asked to select an annotation type among: *analogy*, *cell-function homology*, *tissue homology* or *association*. The first three annotations refer to *analogy* and *homology* defined above. *Association* is selected, when none of the others can be applied.

Figure 3.7 shows the dialogue that can be used to retrieve the homology links.

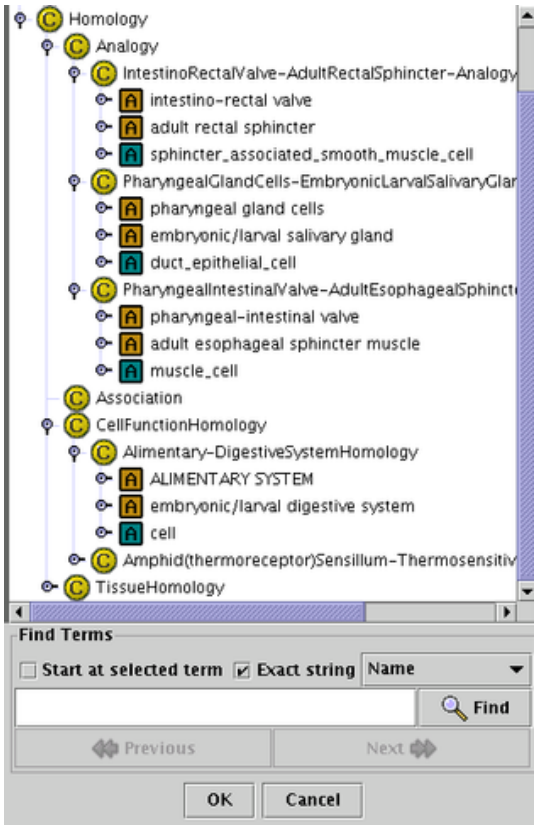


Figure 3.7: shows the homology links stored in the cell-ontology. For each homology link first the tissues and than the cell-types are displayed.

3.4.3 Realisation Issues

To make the annotations two tissues are selected in the two ontologies. After pressing the “Annotation”-button or selecting the corresponding item in the menu a dialogue is opened. Here the user specifies the name of the annotation and selects the annotation type. To select the cell-types a new dialogue is opened that displays the cell-ontology in a tree visualisation. This is the same approach as used for the tree visualisation for the two species ontologies. A search facility can be used in addition to find the cell type. The annotation dialogue is displayed in Figure 3.6.

After confirmation of the annotation it is stored in the same ontology as the cell-ontology. This approach was chosen to prevent the maintenance of too many ontology files. The two tissues can not be linked to their ontologies since they are usually stored

in different ontologies. In the flat-file version there is no mechanism to refer to GO terms in other ontology files. The cell-types can be referred to the cell-ontology since the annotations are stored in the same file as the cell-ontology.

Chapter 4

Experiments and Results

In this chapter I will explain the experiments I have done to evaluate the software system, then I will present the result of these experiments and the analysis of these results.

4.1 Preparation and Realisation of the Evaluation

In this section I describe the software evaluation performed in order to evaluate the application. First the aim of the test is explained, then the group of participants is introduced. After this the experimental setting and the tasks the participants had to perform are described. At the end the evaluation methods used during the evaluation are explained in more detail.

4.1.1 Aim of the Evaluation

The aim of the test was to find out if the tool is easily to use (ergonomic), if the user can do the things she wants to do and if it is an aid in the every day work of the user (suitable for the task) and if it fulfils the user's expectations (adequacy).
Some hypotheses have been defined to be tested by the evaluation:

H1: The tree view is more often used to browse through the hierarchical structure and find terms then the node-based visualisation.

H2: The node-based visualisation is used for additional information about the GO terms and to see multiple parents.

H3: The functionality of the menu-items are well understood.

H4: The functionality can be easily accessed.

H5: Drag-and-drop provides an easy way to edit the relationship links between GO terms.

H6: The application allows biologists to make homology links and explore them.

In the tree visualisation the user can see the hierarchical structure of the ontology. The hypothesis *H1* claims that this is a good method to browse through the ontology and find terms and that the user will prefer it to the node-based visualisation. The hypothesis *H2* claims that the node-based visualisation is used to provide the user with additional information about the terms. In particular it is claimed that the multiple parents are better recognised in the node-based view. This hypothesis can be tested by evaluating which view was used for which purpose. This can be done with the direct observation, video recording and with software logging.
In the hypothesis *H3* it is supposed that the items in the menus are understood. This means that the functionality provided by the application is understood by the user. In addition to this the hypothesis *H4* claims that the tool is easily to use. This can be measured by error rates, capturing the comments of the user and recognising errors the user does. Methods for capturing these data can be software logging, observation and questionnaires.
The hypothesis *H5* claims that the additional functionality of drag-and-drop provides an easy way to edit the ontology. In this case it has to be found out if and when how often the drag-and-drop functionality was used. This can be captured via log-files or observation techniques.
The last hypothesis *H6* claims that the biologists can use the application to make homology links. This was the original purpose of the project.

4.1.2 The Participants

As mentioned in section 2.3.1, it is important to test a product with a group of participants that are typically users for the product. The application described in this dissertation is an application developed specially for Developmental Biologists who work on cross-species mapping of anatomy models. Typical participants who attend the evaluation therefore would be Developmental Biologists who work in this specialised area.
In Edinburgh there are just a small group of people who fulfil the requirements. Two people accepted an invitation to attend the evaluation procedure. Both of them work in Developmental Biology and are experts in the area the application was designed for. To do the evaluation with experts from other cities was not feasible since the time and costs for doing this are too high.
An evaluation with just two attendants is not very extensive. Nielsen (Nielsen, 1994) recommends having 3 to 5 participants for the *thinking-aloud protocol* evaluation. More participants would be desirable for the evaluation. As described above it was difficult to find participants that had the required experience. For this reason I decided to do an evaluation with a small group of non-experts. Six postgraduate students from Informatics and Artificial Intelligence attended the evaluation. The reason for this was that they are familiar with the concept of ontologies. Since they don't have the expertise in Biology it could not be assumed that they understand all of the tasks. In particular the task of making homology links is not suitable for non experts. All of them have been familiar with tree visualisation and hyper-linked documents. Therefore the browsing and editing functionality of the application could be tested also with them.
However it has to be considered that the second group of attendants are not experts in Developmental Biology but in Informatics and therefore the results have to be critically analysed.

4.1.3 The Experimental Methodology

The evaluation of the application COBrA, consisting of three parts, is described here. None of the participants had used the application before the evaluation. Before the

evaluation the functionality of the application was explained to the participant. This was done with the aid of a description that explains the functionality explained to the participants and the order in which it shall be explained. This warranted a uniform initial training state for each user. The following functionality was introduced to the participants:

- different menu items to read and write ontologies.
- the different possibilities for visualising the ontology and different scenarios described in section 3.2.4.
- different ways to browse through the ontology with the different views.
- the editing of the tree with the tree-menu and with drag-and-drop.
- how to make annotations.

After the introduction the participants had time to ask questions and try the functionality of the application. The complete introduction was about 30 minutes for each individual.
The participants were given three different tasks during the evaluation. Non experts were asked to perform just two of these tasks because of the expertise needed for the last task. Each task was designed to test a different part of the functionality of the application. The tasks were further divided into sub tasks. While performing the evaluation the participants were asked to verbalise their thoughts and feelings. This may affect the performance of the participants and has to be considered during the analysis of the results. The comments were recorded with pen and paper. In the first two cases also video recording was used to gather data. The interaction with the tool was recorded into a log-file. While performing the task minimum assistance was given and when it was given, it was noted. The specific tasks given to the participants are described in more detail in section 4.1.4.
At the end of the session each participant was asked to fill in a usability questionnaire with 16 questions. This was used to gather the opinion of the user about the application. The questions are based on the cognitive dimension questionnaire of Blackwell

and Green (Blackwell & Green, 2000) as described in 4.1.5.5. They can be found in Appendix B.

4.1.4 The Tasks

For the evaluation tasks were chosen that the users would expect to do when they are using the application. These are the activity-types: *"adding new things, modify existing structure, copying content of a structure to another one, explore further characteristics and finding things"*, which are discussed in (Green & Blackwell, 1998).
Since the users did not have much time to get familiar with the application, the tasks were designed with increasing difficulty. The first task uses the different visualisation approaches of the application to browse through the ontology. In the second task the user is asked to edit the ontology in order to test the editing functionality. To do this the user has to apply the browsing techniques used in the first task. For the last task the user is asked to make homology links. This task is performed only by the participants that are familiar with the topic.
During the tasks a 'wizard' was used to lead the user through the evaluation. A wizard is a software agent that guides the user through some tasks. This wizard first describes the task the participant should perform and then it asks the user to press the "Start Task"–button to begin the task. After performing the task the participant were asked to press the "End Task"–button. These are used to indicate the start and end of each task in the log-file (see section 4.1.5.4). The wizard is used for the evaluation only: it is not an integral part of the application. During the evaluation some of my spelling mistakes in the task description of the wizard had to be clarified to the participants.
For the task the ontology for drosophila (fruit fly), the mouse ontology and the ontology for C. elegans (round worm) were used.
At the beginning of the evaluation an introduction text was displayed that described the procedure:

> You are in task mode. This is a special mode of the tool. Some additional items are displayed, like this wizard. You are asked to complete a number of tasks. All tasks are logged in a specified log file for later revision of the evaluator. Before each task there will be a short task description and then you are ask to press the 'Start Task' button. When you are finishes you are

ask to press the 'End Task' button.
Press "Start Task", when you are ready to perform the task.

Before each task a short description was given to the participant that explains the main objective of the following sub tasks that have to be performed:

1st Task: Browsing
You are now asked to complete the first task. This task tests in general the browsing through the ontology.
Press "Start Task", when you are ready to perform the task.

The first sub-task of the first task (1.1):

1st Task: Browsing
1. Open now the 'abstract mouse' ontology from the file 'abstract_mouse.go' on the left side of the tool.
Press "Start Task", when you are ready to perform the task.

The second sub-task of the first task (1.2):

1st Task: Browsing
2. Click through the tree view of the ontology on the left side and find the term 'compacted morula' which is a child term of the term 'mouse'.
Press "Start Task", when you are ready to perform the task.

The third sub-task of the first task (1.3):

1st Task: Browsing
3. Use now the search facility at the bottom of the left ontology panel to find the term 'compacted morula' and then use the 'Previous' and 'Next' button to click thought the found terms.
Press "Start Task", when you are ready to perform the task.

The fourth sub-task of the first task (1.4):

1st Task: Browsing
4. Open now the 'drosophila' ontology from the file drosophila.go on the right side of the tool.
Press "Start Task", when you are ready to perform the task.

The fifth sub-task of the first task (1.5):

1st Task: Browsing
5. Use the different views on the right and left side of the tool to click through the different ontologies.
Press "Start Task", when you are ready to perform the task.

In the first task the participant was asked to open first one ontology. By opening just one ontology, the tree view is displayed on one side of the application and the node-based view is displayed on the other side of the application. The intention by doing this was that the participant recognised the connection between the tree view and the node-based view while browsing through the ontology. Then the user was asked to find a specific term in the ontology. At first it had to be found in the tree visualisation and then with the aid of the search-facility. This sub-task was given to show the user different ways to find terms in the ontology and to see if this is understood by the user. After this the participant was asked to open the second ontology on the other side of the application and browse through the two different ontologies with the different views. This was done to introduce the participants to the different opportunities to browse through the ontologies and the different scenarios (3.2.4) the different views can offer. After this the second task was explained to the user by the wizard:

> **2nd Task: Editing**
> You are now asked to complete the second task. This task tests in general the editing in the ontology.
> Press “Start Task”, when you are ready to perform the task.

The first sub-task of the second task (2.1):

> **2nd Task: Editing**
> 1. You think now that the term ’one-cell stage’ in the ’abstract mouse’ ontology is also ’part-of’ ’4-8 cell stage’. Edit this new relationship.
> Press “Start Task”, when you are ready to perform the task.

The second sub-task of the second task (2.2):

> **2nd Task: Editing**
> 2. You found out that this was a wrong decision. Remove now the link ’one-cell stage’ from the parent term ’4-8 cell stage’.
> Press “Start Task”, when you are ready to perform the task.

The third sub-task of the second task (2.3):

> **2nd Task: Editing**
> 3. You now found out that the term ’embryo’ is no longer used in the ontology. Delete the term from the ontology and attach its children to the term ’first polar body’.
> Press “Start Task”, when you are ready to perform the task.

The fourth sub-task of the second task (2.4):

> **2nd Task: Editing**
> 4. Add the new term 'third polar body' to the term 'mouse' in the 'abstract mouse' ontology.The new id for the term is 'EMAPA:00035000'.
> Press "Start Task", when you are ready to perform the task.

In the second task the user was asked to edit some of the relationship links, remove a term from the ontology and add a new term to the ontology. For this it was necessary to use the mouse-menu or the drag-and-drop functionality where applicable.
The second task was the last task done by the students. The last task was not given to them since they don't have the expertise to do this.
For the two experts the following and last task was given:

> **3rd Task: Annotation**
> You are now ask to complete third task. This task tests in general the functionality to make annotations.
> Press "Start Task", when you are ready to perform the task.

The first sub-task of the third task (3.1):

> **3nd Task: Annotation**
> 1. Open now the 'celegans simple' ontology from the file 'celegans_simple.go' on the left side of the tool.
> Press "Start Task", when you are ready to perform the task.

The second sub-task of the third task (3.2):

> **3nd Task: Annotation** 2. Find term 'intestine' with the id 'WBdag:5772' in the 'celegans' ontology.
> Press "Start Task", when you are ready to perform the task.

The third sub-task of the third task (3.3):

> **3nd Task: Annotation** 3. Find term 'embryonic fat body' with the id 'FBbt:5710' in the 'drosophila' ontology.
> Press "Start Task", when you are ready to perform the task.

The fourth sub-task of the third task (3.4):

> **3nd Task: Annotation** 4. Create the new Annotation 'Intestine-FatBodyHomology' with the id 'CL:0010006'. It has the cell type 'metabolising_cell' which has the id 'CL:0000181'. The homology type is 'CellFunctionHomology'
> Press "Start Task", when you are ready to perform the task.

The fifth sub-task of the third task (3.5):

> **3nd Task: Annotation** 5. Find the Annotation 'Intestine-FatBodyHomology' in the cell ontology.
> Press "Start Task", when you are ready to perform the task.

The last task asked the participant to find a term in each of the two ontologies and then make a homology link. At the end the participant was asked to find this link in the *cell-ontology*.
To prevent the participant from using the menu or the tool bar before she pressed the "Start Task"-button, the menu and the tool-bar were disabled. This is done to log the interactions with the task information. Without preventing the user from using the functionality the log entries would not contain the task information. For some of the tasks it was not necessary to use the tool-bar or the menu. In this case the users sometimes forgot to press the "Start Task"-button since the tree or the node-based view could still be used. The participants were reminded to do it in this case. To keep the evaluation unaltered for the two experts, this was not changed for the first two evaluations. For the student evaluations the tree and node-based visualisation were disabled to avoid this problem.

4.1.5 The Evaluation Methods

In (Holleran, 1991) the author recommends gathering different types of data to increase the validity of the data through agreement and consistency among the different results. Additional methods can also be used to support other methods. This is done e.g. in video recording with software logging.
For the different tasks, observation methods have been applied to gather the direct response of the participant while using the application. To capture the opinion and the feelings of the participants and the problems they had while using the application they were asked to articulate their thoughts. These responses were noted. For the evaluation with the two experts a video camera was used in addition to video record the screen and the comments the participant made.

4.1.5.1 Direct Observation

For the direct observation I was sitting next to the participant. This was close enough to recognise the screen but far enough to try not disturb the participant while she was doing the tasks. During the evaluation the comments of the users and recognised problems have been noted. This was done separately for each sub-task with the task and sub-task number to identify the different tasks while analysing the record.

4.1.5.2 Video Recording

For the evaluation with the two experts, in addition to the other methods, a digital video camera was used to capture the screen and the comments of the participant. The equipment was just available for a limited time and therefore just used for these two evaluations and not for the evaluations with the students.
Before the start of the evaluation the camera was placed on the left side of the screen. The left side was used because the participants are right-handed. On this side the camera disturbed the participant less while she was doing the tasks. But it could still record the screen in a useful way. The camera also recorded the verbal comments the participants made. These can be used in addition to the notes made during the direct observation.

4.1.5.3 Thinking Aloud

During the evaluation the participants were asked to verbalise their thoughts, to articulate their problems and feelings while doing the tasks. The comments were noted on paper during the direct observation and, during the evaluation with the two experts, additionally with the video equipment.

4.1.5.4 Software Logging

In (Hilbert & Redmiles, 2000) the authors deeply discuss the extraction of *"Usability Information from User Interface Events"* and of the analysis of this information. The analysis of these data is extremely time consuming since the amount of data is *"extremely voluminous"*. For the short time of the project with implementation and

evaluation this was not feasible. I decided to log the user events with additional information about the task performed to prevent gathering too complex data.
As described above the participants for the evaluation are led through the tasks by a wizard. Before each task the user has to press the “Start Task”-button and after finishing the task she has to press the “End Task”-button. These have been included in order to log the time when a task has been started and stopped. For the tasks I thought of the relevant user events to log.
For the first task I decided to log the following events:

- reading an ontology (inclusive cancelling of this action).
- changing the tab and therefore the views.
- searching in the ontology.

I did not log the browsing through the ontology on either visualisations. I thought it will generate too much data which I can not analyse in the available time.
For the second task more information was needed since the participants had to edit the tree. Here I also logged the following interactions:

- editing in the tree (remove link to term, remove term, rename term, drag and drop, copy, cut and past).
- add a new GO term or edit the properties of an existing GO term in the tree.
- when hiding relationships as described in section 3.2.5.
- when the dialogue with the description of the relationships is displayed.

This are in general all the editing possibilities. The problem is that without additional information the analysis is very difficult and time consuming. The video records, the notes of the observation and the information in the log-file about the task that is performed are essential aids in the analysis.
For the third task the homology annotations had to be created and additional information had to be logged:

- the annotation created (name, tissues and cell-types).

- problems when creating annotation (missing part of the annotation like the name or cell-type).
- cancel when making the annotation.

The synchronisation with the video and the notes of the observation were done manually. In general the log-information was taken to give additional information to the notes made and to analyse the time used for the tasks.

4.1.5.5 Questionnaire

For the questionnaire the cognitive dimensions questionnaire proposed by Blackwell and Green in (Blackwell & Green, 2000) was used as a basis. The questions are formulated in general terms and can be used for many information structures. Since some of the dimensions are not essential for the evaluation not all questions have been used. Of the original 14 dimensions 7 have been chosen. It covers the dimensions:

visibility and juxtaposability: the ability to view components easily and the ability to place any two components side by side.

viscosity: resistance to change (to find out if the system is hard to modify).

error proneness: notation invites mistakes.

closeness of mapping: closeness of representation to domain.

hard mental operations: high demand on cognitive resources.

role expressiveness the purpose of a component is readily inferred.

consistency similar semantics are expressed in similar syntactic forms.

defined in (Blackwell & Green, 2000). It does not cover the dimensions: diffuseness, hidden dependencies, progressive evaluation, provisionality, premature commitment, secondary notation and abstraction management.
The questionnaire provides different question proposed in (Blackwell & Green, 2000) for the different cognitive dimensions. All questions are open questions.

4.2 The Results of the Software Evaluation

This section summarises the results of the user evaluation. The evaluation took place in the time from the 13th of August to the 21st of August in 2003. The evaluation with the two experts took place on the 13th and the 14th of August in a conference room of the School of Informatics at the University of Edinburgh. The evaluation with the students took place on the 20th and 21st of August at the MSc computer labs at the School of Informatics. During the evaluation the experts performed three tasks and were video recorded in addition to the other evaluation methods described in 4.1.5. The students did just the first two of the tasks. They were not video recorded while performing the task.
None of the participants had any experience with the COBrA application before the evaluation took place. Some of them had used other applications with similar purposes for editing and browsing ontologies before. All of the participants have been familiar with the concept of ontologies, although the students were not familiar with the domain.

4.2.1 User Experience

This section shortly introduces the group of participants that attended the evaluation. The Table 4.1 shows the summary of the answers about the user experience in the questionnaire. The questionnaire is listed in Appendix B.
All participants did an introduction of 30 minutes before they had to perform the tasks. Just one of the participants felt proficient with the use of the application. All the others thought they were not proficient. As it can be seen in Table 4.1 most of the students had not use an ontology editor and browser before. Both of the experts used browser and editors for ontologies before.

[3]http://oiled.man.ac.uk/index.shtml
[3]http://www.aiai.ed.ac.uk/project/xspan/
[3]http://genex.hgu.mrc.ac.uk/intro.html

Participant	Length of Use	Proficient	No. of Similar Tools Used
1	30min	yes	0
2	30min	no	1 (OILed[1])
3	30min	no	0
4	30min	no	0
5	30min	no	0
6	30min	no	0
7	30min	no	2 (DAG-Edit, XSPAN Acquisition Tool[2])
8	30min	no	1 (anatomy editor[3])

Table 4.1: shows the answers to the user experience-questions from the questionnaire.

4.2.2 Issues Identified while Performing the Tasks

While performing the task most of the participants made positive comments about the application. These comments are not taken into account here. This section presents the issues that have been identified during the observation of the participants while doing the tasks. With issues I mean some aspect of the system that caused some problem or situations which hindered the participant in her task.

Table 4.2 summarises the time taken for each task. The last column in the table summarises the time taken for the first two tasks. Since the students did not perform the last task, there are no values in this column for the students. The last three rows show the average time, the minimum time, and the maximum time taken to perform the task. The time required by the two experts (participant 7 and 8) to perform the tasks are very close. However the time taken to finish the tasks for the students differ much.

The participant 3 completed the task in the shortest time. I recognised in the notes of the observation that she did not experience any problems while performing the tasks (compare Table A.1 in Appendix A). Subject 2 took the longest time to complete the two task. She experienced some problems in sub-task 2.1 and 2.2 (compare Table A.1 in Appendix A). In this cases I had to give advice. Subject 4 took the longest time to complete the first task. In the notes of the observation I recognised some problems in understanding the causal connection between the different views and the fact that two

Participant	Task1	Task2	Task3	Task1 + Task2
1	4:40 min	10:19 min	-	14:59 min
2	5:55 min	12:29 min	-	18:24 min
3	4:06 min	4:17 min	-	8:23 min
4	7:11 min	7:00 min	-	14:11 min
5	3:48 min	6:08 min	-	9:56 min
6	5:29 min	10:12 min	-	15:41 min
7	5:18 min	10:36 min	7:11 min	15:54 min
8	6:04 min	9:52 min	7:12 min	15:56 min
average	5:19 min	8:52 min	7:12 min	14:11 min
minimum	3:48 min	4:17 min	7:11 min	8:23 min
maximum	7:11 min	12:29 min	7:12 min	18:24 min

Table 4.2: shows the time needed by the participants for each task. It also shows the average, minimum and maximum time needed for each task calculated over all participants.

different ontologies are displayed. Participant 5 performed the first task in the shortest time and participant 3 performed the second task in the shortest time. No particular problems were noted for these participants in these tasks. The two experts finished the last task in nearly the same time.

The Table 4.3 shows the summary of the notes taken while observing the performance of the tasks. For each sub-task the number of participants that completed the sub-task is shown. All eight participants performed the first two tasks and two participants performed the last task. The third column shows the number of issues raised during the task. The fourth column shows the number of times advices was given to the participants. The last column shows the description of the nature of the issues identified. The numbers in parentheses in this column refer to the numbers in Table A.1 in Appendix A.

It can be seen in the Table 4.3 and Table A.1 that the second task raised most issues. Especially in sub-task 2.1 where the participants were asked to create a new relationship link between two GO terms and in sub-task 2.2 where the participants were asked

Task	No. of Completed Tasks	No. of Raised Issues	No. of times advice given	Description
1.1	8	0	0	
1.2	8	1	1	(1) se: wrong spelling of GO term
1.3	8	0	0	
1.4	8	1	1	(2) a: open ontology on the right side
1.5	8	2	2	(3) a: problems with understanding the different tabs for the ontologies
2.1	8	3	3	(4) a: described how to use edit menu (5) a: problems with the search options (6) a: how to choose term in *edit menu*
2.2	6	3	2	(7) se: could not remove children in *edit term* dialogue (8) m: removed term instead of link (9) a: reminded user to use *remove link*
2.3	7	1	0	(10) se: dialogue to attach children did not come up
2.4	8	2	2	(11) a: there are two edit fields for id (12) a: use *add term* instead of *edit term*
3.1	2	1	1	(13) a: does not have to close ontology to open new
3.2	2	0	0	
3.3	2	0	0	
3.4	2	1	1	(14) a: does not have to load cell-ontology to find the cell-type
3.5	2	1	1	(15) a: does not have to load cell-ontology to find the homology annotation

Table 4.3: shows the summary of the observation of the tasks. The first columns shows the task number. The second column shows the number of participants that completed the task. The third column shows the number of issues that were raised. The fourth column shows how many times advice had to be given. In the last column a description is given to the problems. *m* stands for user mistake, *a* for advice given, usually after a user mistake or when the user is stuck, and *se* for system error.

to remove the link to a GO term many issues were notified. Most of the problems identified here happened while using the editing menu as it can be seen in the last column of Table 4.3. Tow participants could not complete this sub-task. One reason for this was a software problem during the first evaluation. The problem was eliminated for the next evaluations. In the other case an incorrect solution was chosen. A software problem also raised an issue during the first evaluation in sub-task 2.3.

4.2.3 Issues Identified through the Questionnaire

I now present the participants' subjective opinion of their experience with the application. Table 4.4 shows the number of issues identified in the answers of the usability questionnaire. The questionnaire does not refer to the cognitive dimensions explicitly. Since each question is related to a cognitive dimension, the answers can be related to these dimensions. By issue an aspect of the application is meant that caused problems or that could be improved according to the opinion of the user.

Cognitive Dimension	No. of Issues as Questionnaire	No. of Issues Reclassified
visibility and juxtaposability	5	5
viscosity	3	2
hard mental operation	4	4
error proneness	4	3
closeness of mapping	2	2
role expressiveness	3	6
consistency	1	0

Table 4.4: shows the number of issues raised for each cognitive dimension. The first column shows the cognitive dimension. The second one shows the issues raised like they appear in the questionnaire. And the last column shows the issues raised after reclassifying them. (see text)

In Table 4.4 it can be seen that the cognitive dimension *visibility and juxtaposability* raised most of the issues according to the answers of the questionnaire followed by

No.	Cognitive Dimension	Issues Raised	from CD
1	visibility and juxtaposability	- homology annotations are not easy to find (5) - multiple parents in tree view not always obvious (2),(17) - difficult to see relationship dialogue (3) - facility to specify depth of expanded level (1)	1, 6, 1,5
2	viscosity	- needs undo functionality (6),(8)	
3	hard mental operation	- difficulties to keep track of what is displayed in the tabs (9) - node-based view: can get lost in hierarchy (11) - can get lost in fairly large ontologies in tree (19),(20)	6
4	error proneness	- easy to forget to unmark the check box *exact string* for searching (4),(13) - can give the same id to different concepts (15)	
5	closeness of mapping	- nodes without children should be recognised (18) - labels in the tabs are not expressive enough (10)	3
6	role expressiveness	- different relationships distinguished just by colour (21),(22) - the menu items to edit the tree are not always obvious (12), (14) - the menu items edit and add are misunderstood (7),(16)	4,3, 2
7	consistency		

Table 4.5: shows the issues raised for the different cognitive dimensions. Some of them were raised more than once. The number in brackets behind the issue refers to the numbers in Table A.2. Some of the issues had to be reclassified. The cognitive dimension (CD) for which it was answered originally is displayed in the last column.

the cognitive dimensions *hard mental operation* and *error proneness*. This can be seen in Table A.2 in Appendix A in more detail. However, after analysing the answers I found out that some of the issues appeared also in the answer to other questions. Some

of the questions were not correctly understood by the participants since the answers given referred rather to other questions. For this reasons I had to reclassify the given answers.
The numbers of issues identified for each cognitive dimension after the reclassification are shown in the last column in Table 4.4. The actual issues raised are presented in table 4.5 where the last column shows the cognitive dimension where the answers belonged before they were reclassified.
After the reclassification of the answers it can be seen that the cognitive dimensions*role expressiveness* and *visibility and juxtaposability* raised the most issues followed by *hard mental operation* and *closeness of mapping*. Other issues were identified for the dimensions: *error proneness* and *viscosity*. Some of the issues raised can be seen as belonging to more than just one dimension. For instance the problem with the distinction of different relationships mentioned in the dimension *role expressiveness* could also appear in *visibility*.
The most frequently mentioned issues are problems while using the menu items to edit the tree. Other issues that were raised frequently are: multiple parents are not obvious in the tree visualisation, and the relationship is just displayed by the colour of the icon.

4.2.4 Additional Issues Raised during the Evaluation

In addition to the issues that were identified while performing the task and the once that were raised in the answers of the questionnaire some additional comments were made during the evaluation:

- In many comments a undo functionality was desired to annul mistakes.
- Some participants mentioned that it was difficult to recognise if the editing task was executed. A better way to show that changes were made is desired.
- Some of the participants also said that it would be good to confirm changes in the tree to prevent the user from doing things she did not want to.
- Two of the participants asked for a mechanism to specify the depth of the expansion so that an expanded tree can be easily collapsed in the specified depth.

- Some participants said that the tree visualisation is more helpful in browsing than the node-based visualisation. But the node-based visualisation provides additional information like multiple parents that is helpful.
- One participant also said that it would be good to limit the number of entries in the editing fields.

4.3 Analysis of the Results

In this section I give possible interpretations of the reasons for the issues identified during the evaluation.

4.3.1 Issues Identified while Performing the Tasks

As mentioned in the section 4.2.2 participants 3 and 5 finished the task in a much shorter time than the average. One reason for this is that they had no big difficulties in completing the tasks. This can be seen in Table A.1. On the other hand participant 2 needed the longest time in performing the tasks. She also did not have significant problems in completing the tasks. A possible explanation for that is that she probably considered all possibilities and thought more carefully about each task. A video record or more detailed notes of the observation could provide a more detailed explanation.
From the Table 4.3 it can be seen, that the second task raised more issues than the first task. The third task will be discussed separately because it was performed just by two participants. The tasks that caused problems are examined in more detail below.
In the first task sub-task 1.2 and in the second task the sub-tasks 2.1 and 2.2 caused problems during the first evaluation because of software problems. These were remedied for the next evaluations.
In sub-task 1.4 one participant wanted to open the ontology on the wrong side. The same participant had problems in understanding the different tabs during the sub-task 1.5. (This participant was one individual from the group of students.) Other participants did not have similar problems during the tasks. In this case a longer and better introduction to the tool and the possibilities of displaying the ontology might solve the

problem.

One reason for the fact that in the second task more issues were raised is that the task was more difficult than the first task. In addition to the browsing of task one also editing had to be done. Some of the participants did not completely understand the menu items in the edit menu as can be seen in Table 4.3 for the tasks 2.2 and 2.4. This led to wrong or not optimal decision in choosing the editing functionality. In one case this led to the point where the task could not be completed. It has to be verified if the menu items are appropriate and if they can be understood unambiguously.

It was also recognised that some of the participants did not always read the task descriptions very carefully. In these cases the participants did not follow the instructions of the tasks. They sometimes had to be reminded of the task. One possible explanation for this is that it comes from lack of confidence of the participant and therefore loss of concentration on the task as it is described in (Preece et al., 1994). An evaluator should be prepared for this but it is difficult to prevent it. Some more problems that can arise during evaluation with user involvement are discussed in (Wilson et al., 1997). It could also come from not well explained tasks or unintuitive.

One participant had to be reminded to how to edit the ontology. The reason for this might come from the fact that the participant works with an Apple computer with Mac OS X and did not understand the interface paradigm. For this operating system the right mouse button is simulated with the “Apple Button” and the single mouse button. In Mac OS X Editing this is not a very natural way of editing while it is a common way for user of Linux[1] and Microsoft Windows operating systems. It has to be verified if there are more suitable ways for Mac users to edit the tree.

In general I can say that most of the issues that raised in the second task came from not well understood menu items. Some of the participants could not clearly distinguish some of the menu items. Here is a need for clarification of the menu terms used or a change of the terms used.

The third task was performed by the two experts. In sub-task 3.1 one of the participants did not know if she had to close the ontology before opening another one. In my opinion this is not a major problem. It can be clarified easily. The application works

[1] with e.g. KDE as window manager

in this case like most of the ontology editors and browsers and other editors.
The fact that one of the participants in the sub-tasks 3.4 and 3.5 wanted to open the cell-ontology in the editor window needs more clarification. The user should be better informed about the fact that the cell-ontology is opened at the start of the application automatically. Both of these issues raised in the third task do not influence the usability of the application in general. A better training of the user can avoid these problems.

4.3.2 Issues Identified through the Questionnaire

In (Blackwell & Green, 2002) the authors say about the cognitive dimension *visibility and juxtaposability* that: *"Systems that bury information in encapsulations reduce visibility"*. One issue identified in this cognitive dimension was that the homology annotations are not easy to find (see Table 4.5). The annotations are stored in the cell-ontology and accessible in a tree representation like the tree visualisation for the opened ontologies. The annotations can not be seen directly in the application. They are displayed in an extra dialogue that is opened with a button in the tool-bar. This approach was chosen to avoid an overload of functionality and direct visible parts of the application. I considered it as a good trade-off.
Another issue raised was that multiple relationships are not always obvious in the tree visualisation. Since in a tree a node can have just one parent multiple parents have to be indicated with a special mechanism. For the COBrA ontology browser I decided to indicate multiple parents with additional information in the icon of each node. In comments of other participants this has been found to be a good aid. For additional parent information the node-based visualisation is available.
The comment that has been given among the answers for the cognitive dimension *viscosity* was to add undo functionality. This functionality was considered before but could not be implemented because of limited time for the project. I am aware that this is a fundamental functionality that should be provided by the application.
For the cognitive dimension *hard mental operation* the authors in (Blackwell & Green, 2002) say that: *"A notation can make things complex or difficult to work out in your head, by making inordinate demands on working memory, or requiring deeply nested goal structures"*. For this cognitive dimension three issues were raised. One partici-

pant had problems with *"keeping track of what is being displayed on each side"*. There are four tabs available on each side since it is possible to open two ontologies at the same time and it is possible to display either visualisation of either ontology. This can confuse the user, when she does not carefully considers the visualisations she wants to see. A better introduction in the scenarios described in section 3.2.4 is essential.
One participant said that she can easily get lost in the ontology while using the node-based view. As explained in section 2.1.3.3 the node-based visualisation is not very expressive in showing the hierarchical structure of the ontology. For this reason the tree visualisation has been introduced. A explanation of how to use both visualisations to explore the ontology can help to solve the problem.
A similar issue was raised by another participant about the tree visualisation. She said that someone can get lost even in the tree visualisation when the ontology is too big. To reduce the information the user has to cope with two mechanisms can be used. Subtrees can be collapsed or particular relationships can be hidden in the tree as described in section 3.2.1.2.
An issue that was raised in the cognitive dimension *error proneness* is the fact that there is no mechanism to prevent the user to give the same id to different GO terms. This problem should be solved since it can not be expected that the user knows what ids she can use.
Most of the problems in the cognitive dimension *role expressiveness* come from editing. This problem was also recognised while observation the participants during the evaluation. In this cognitive dimension it was also mentioned that relationships are not easily distinguished. The use of more expressive icons could help to solve the problem.

4.3.3 Testing the Hypotheses

In section 4.1.1 I proposed six hypotheses:

H1: The tree view is more often used to browse through the hierarchical structure and find terms then the node-based visualisation.

H2: The node-based visualisation is used for additional information about the GO terms and see multiple parents.

H3: The functionality of the menu-items are well understood.

H4: The functionality can be easily accessed.

H5: Drag-and-drop provides an easy way to edit the relationship links between GO terms.

H6: The application allows biologists to make homology links and explore them.

that should be verified with the evaluation.
In this section I will try to prove or disprove these hypotheses on the basis of the results of the software evaluation and the analysis of these results.
The hypothesis *H1* can be proven with the fact that all the participants used the tree visualisation in most of the tasks. With user comments like:

- “The tree graph is very useful”
- “the tree helps a lot”
- “simple tree display”
- “the tree structure helps”

and similar comments provided in the questionnaire and while performing the task this hypothesis is supported.
I recognised that the node-based visualisation was sometimes chosen in addition to the tree visualisation to navigate through the ontology. This is claimed by the hypothesis *H2*. Comments like:

- “it is necessary to have a graphical idea in your mind (and so look at the tree), while it is good to have precise data (like parents or children)”
- “I can view all structure in the tree view and I can compare more detailed relationship of an element in node-based view” or
- “the tree structure is very helpful and also the second window with the additional information”

also support the hypothesis.
The hypothesis *H3* was disproven by the fact that the editing in most of the cases has been a problem as it can be seen in section 4.3. This issue needs more attention since the editing in ontologies is a critical point.
In one case the participant had problems to remember how to access the editing functionality. As I tried to explain it in section 4.3.1 this might come from the fact that the participant is usually using the operating system Mac OS X. In general the participants did not have difficulties to access the functionality in the tool. Some comments have been made that the application guides through the tasks and that the use of similar notation helps to use the tool. This supports the hypothesis *H4*.
The hypothesis *H5* can not easily proven. Drag-and-drop was not used during the task. Although some comments have been made that the drag-and-drop functionality of the nodes is “useful”. The feature was recognised but not used. A longer period of use could prove or disprove the hypothesis.
The last hypothesis *H6* can just be proven with the results of the test with two participants. The two biologists that attended the evaluation did not have major problems in creating homology links and editing them. The problem that occurred where one of the biologist wanted to open the ontology in one side of the application can although it was available with choosing the button or menu item can be avoided by better introduction of the functionality.

Chapter 5

Conclusions and Future Work

In this chapter I describe what the results and the analysis of the results mean for the usability of the application, then I describe some points that can be improved or need more attention and after this I present suggestions about what can be done next.

5.1 Conclusions

The aim of the project was to develop and implement a tool that can can be used to explore and edit ontologies of anatomies of different species and establish homology links across these anatomies.
In this project I presented the contributions I made that led to the development and implementation of the COBrA ontology browser. The COBrA ontology browser allows the display of two ontologies simultaneously. This enables the biologist to see the two ontologies of anatomy at the same time to establish homology links. To display the ontologies and explore them the COBrA ontology browser provides both a tree visualisation and a node-based visualisation. The tree view provides a hierarchical perspective of the relationships between concepts in the ontology. The node based visualisation displays the facts about the concepts in the ontology and provides hyperlinks to explore the ontology.
However, the evaluation exposed some issues that need to be examined in more detail. The evaluation has shown that some of the editing menu-items need to be revised to

avoid ambiguity and adapt the terminology to be closer to that of the tasks the functionality is addressing. Here a closer cooperation with biologists is needed to reengineer the editing functionality and adopt it to the biologists' needs.
There are two important issues that I think should have been addressed for the project but time limitations did not allow that. The first issue involves the implementation of an undo-functionality for the editing task. Since the user is always likely to make mistakes while using editing-tools, an undo function should be implemented to help the user reverse editing mistakes. For each change made in the ontology a sequence of inverse changes has to be derived to completely undo the previous changes so that all the previous states of the ontology can be reconstructed.
The second issue involves the implementation of reading and writing the XML-version of the GO file format. In my opinion the XML version of the GO file will be more important in the future and will replace the flat-file version since XML and RDF are widely used standards to represent ontologies. The implementation involves the parsing of the XML file and mapping to the background data structure described in 3.1. The parsing could be done with RDF parsers such as that provided in the Jena[1] API.
Another issue that should be addressed again is that the evaluation was done with mainly non-biologists. Therefore it is not yet clear if the tool is actually adequate for supporting biologists in making homology links. In my opinion it is necessary to carry out additional evaluations with specialists to see if the application achieves its goal.
The high level of functionality in ontology browsers and editors like Protégé and KAON OIModeler gives the impression that they are more suitable for users like Knowledge Engineers. Domain experts like biologists might be put off by using them. To perform specialised tasks as required in the domain of this project may require additional functionality and different visualisation techniques.

5.2 Future Work

In section 2.1.3 and 3.2.1 I introduced additional visualisation techniques in addition to the tree and node-based visualisation. Some of these techniques might be more

[1] http://jena.sourceforge.net/

adequate than the tree and node-based visualisations for making homology links or providing additional aids for biologists. One reason for doing this is that the tree visualisation is not sufficient for displaying multiple parent relationships. A new evaluation with additional visualisation techniques can show if other visualisation techniques are more adequate.

The application is supposed to be an aid to the work of the biologists. Since it provides editing functionality for ontologies more effort in these components of the tool can be spent to support the domain expert in this work. In particular when adding new concepts to the ontology assistance for the user can prevent inconsistency in the ontology. The application could guide the user, with the aid of a rule-base, through the steps needed when choosing the relationship between concepts. In (Winston et al., 1987) the authors propose a way to classify the *part-of*–relationship on the basis of criteria described in the article.

Other issues like adding, removing and modifying concepts in the ontology are addressed in detail in (Maedche et al., 2003) and (Stojanovic & Motik, 2002). It could be interesting to examine in more detail if some of the issues mentioned in these papers can be formulated in rules that can be integrated in a rule base to support ontology editing.

Appendix A

Results

Table A.1 and Table A.2 show the issues raised by each participants.

Task	1	2	3	4	5	6	7	8
1.1	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	(1)	-
1.3	-	-	-	-	-	-	-	-
1.4	-	-	-	(2)	-	-	-	-
1.5	-	(3)	-	-	-	-	-	-
2.1	-	(5)	-	-	(6)	-	(4)	-
2.2	-	(9)	-	-	-	(8)	(7)	-
2.3	-	-	-	-	-	-	(10)	-
2.4	-	-	-	-	-	(12)	-	(11)
3.1	-	-	-	-	-	-	-	(13)
3.2	-	-	-	-	-	-	-	-
3.3	-	-	-	-	-	-	-	-
3.4	-	-	-	-	-	-	(14)	-
3.5	-	-	-	-	-	-	(15)	-

Table A.1: shows the issues raised for each task and participant. The numbers in the cells refer to the issues listed in Table 4.3

CD	Question	1	2	3	4	5	6	7	8
1	1	-	-	-	-	-	-	-	(1)
	2	(2)	-	-	(3)	(4)	-	(5)	-
	3	-	-	-	-	-	-	-	-
2	4	-	(6)	-	-	-	(7)	-	(8)
	5	-	-	-	-	-	-	-	-
3	6	(9)	(10)	-	-	(11)	-	-	-
	7	-	-	-	-	(12)	-	-	-
4	8	-	(13)	(14)	(15)	-	(16)	-	-
	9	-	-	-	-	-	-	-	-
5	10	(17)	-	-	-	-	-	-	-
	11	(18)	-	-	-	-	-	-	-
6	12	-	(19)	-	-	(20)	-	-	-
	13	-	-	(21)	-	-	-	-	-
	14	-	-	-	-	-	-	-	-
7	15	-	-	-	-	-	-	-	-
	16	-	-	-	(22)	-	-	-	-

Table A.2: shows the issues raised in the cognitive dimension (CD) questionnaire for participant. This are the original results before the reclassification. The issues raised are listed in Table 4.5. The cognitive dimensions are 1. visibility and juxtaposability, 2. viscosity, 3. hard mental operation, 4. error proneness, 5. closeness of mapping, 6. role expressiveness and 7. consistency.

Appendix B

Usability Questionnaire

This questionnaire collects your views about how easy the Concept Ontology Browser for Anatomy (COBrA) is to use. The questions refer to actions and operations such as *brows*, *edit* and *make annotation* which apply to the various components of the system. The relevant components are sometimes identified in the question in order to assist with understanding the question. However, if you feel the question relates to other objects please mention that in your answer. Some questions refer to *notation* - in this context notation is the form of information used to communicate with the system. Notation can include text, special symbols, pictures, and diagrams. The letters on a computer keyboard and the text on the screen are two notations which are very similar. The numbers on a telephone key-pad and the clicks and tones heard while dialling are less easy to relate. Please consider *presentation* and *notation* in the widest sense.

A. How long have you been using the COBrA tool ?

B. Do you consider yourself proficient in its use ?

C. Have you used other similar systems ? If so please name them.

1. How easy is it to see or find various elements of the ontology while it is being created or changed? Why?

2. What kinds of things are more difficult to see or find?

3. If you need to compare or combine different parts, can you see them at the same time? If not, why not?

4. When you need to make changes to previous work, how easy is it to make the change? Why?

5. Are there particular changes that are more difficult or especially difficult to make? Which ones?

6. What kind of things require the most mental effort with the display?

7. Do some things seem especially complex or difficult to work out in your head (e.g. when combining several things)? What are they?

8. Do some kinds of mistake seem particularly common or easy to make? Which ones?

9. Do you often find yourself making small slips that irritate you or make you feel stupid? What are some examples?

10. How closely related are the tree and node views to the structure that you are describing? Why?

11. Which parts seem to be a particularly strange way of doing or describing something?

12. When viewing the representations, is it easy to tell what each part is for in the overall scheme? Why?

13. Are there some parts that are particularly difficult to interpret? Which ones?

14. Are there parts that you really don t know what they mean, but you put them in just because it s always been that way? What are they?

15. Where there are different parts of the notation that mean similar things, is the similarity clear from the way they appear? Please give examples.

16. Are there places where some things ought to be similar, but the notation makes them different? What are they?

Appendix C

GO Flat-File

This is an example of a GO flat-file. Lines that start with an exclamation mark (“!”) are comments. The line that starts with the dolar (“$”) indicates the domain of the ontology. Each line after the domain represents a GO term. Indentation is used to indicate parent-child relationships. A *is-a*-relationship is indicated by the “ˊ, *part-of* relationship is indicated by “%” and lineage is indicated by “ ”. Properties of a GO term are separated by a semicolon.
Each line also contains the GO id of the GO term. This comes directly after the term name and is separated by a semicolon. A line can also contain additional information like synonyms of the GO term. Multiple parents are listed after the term properties. They are recognised by the relationship symbol. A line for a go term follows the schema below:

```
< | % term [; db cross ref]* [; synonym:text]*  [ < | % term]*
```

The fragment of the GO flat-file of the drosophila-ontology is shown below. I had to leave out some information for some of the lines to keep the idea of the GO flat-file. These lines can be recognised by triple dots:

```
!autogenerated-by:     DAG-Edit version 1.320
!saved-by:             gwg
!date:                 Tue Sep 09 14:42:51 BST 2003
!version: $Revision: 2.835 $
!type: % ISA Is a
!type: < PARTOF Part of
$Gene_Ontology ; GO:0003673
 <molecular_function ; GO:0003674
  %anticoagulant activity ; GO:0008435
  %antifreeze activity ; GO:0016172
   %ice nucleation inhibitor activity ; GO:0016173
  %antioxidant activity ; GO:0016209
   %glutathione dehydrogenase (ascorbate) activity ; GO:0045174 ...
   %glutathione-disulfide reductase activity ; GO:0004362 ...
   %peroxidase activity ; GO:0004601, GO:0016685 ...
   %thioredoxin-disulfide reductase activity ; GO:0004791 ...
  %apoptosis regulator activity ; GO:0016329
   %apoptosis activator activity ; GO:0016506
    %apoptotic protease activator activity ; GO:0016505 ...
   %apoptosis inhibitor activity ; GO:0008189
  %binding ; GO:0005488 ; synonym:ligand
   %acyl binding ; GO:0000035
   %amino acid binding ; GO:0016597
    %glutamate binding ; GO:0016595 ; synonym:glutamic acid binding
    %glycine binding ; GO:0016594 ; synonym:Gly binding ...
   %antigen binding ; GO:0003823 ; synonym:antibody ...
    %peptide antigen binding ; GO:0042605 % peptide binding ; GO:0042277
     %endogenous peptide antigen binding ; GO:0042606
     %exogenous peptide antigen binding ; GO:0042607
    %toxin binding ; GO:0015643 ; synonym:antitoxin activity
   %bacterial binding ; GO:0008367
```

```
 %Gram-negative bacterial binding ; GO:0008368
 %peptidoglycan binding ; GO:0042834 ...
%boron binding ; GO:0046714
%calcium oxalate binding ; GO:0046904
%carbohydrate binding ; GO:0030246
 %peptidoglycan binding ; GO:0042834 ...
 %polysaccharide binding ; GO:0030247
  %cellulose binding ; GO:0030248
  %chitin binding ; GO:0008061
  %galacturonan binding ; GO:0048028 ...
 %sugar binding ; GO:0005529
  %disaccharide binding ; GO:0048030
   %lactose binding ; GO:0030395
```

Appendix D

GO XML-File

This is an example snapshot of an XML-version of the GO flat file. The basic unit of the GO XML-version is the *go:term* which has an *rdf:about* attribute. This is used to be refered from other parts of the XML-file. Relationships are represented with the tags *go:isa* and *go:part-of*. These tags contain the attribute *rdf:resource* which points to the parent term for the relationship.
The example was taken from the GO-homepage[1]:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE go:go>

<go:go xmlns:go="http://www.geneontology.org/xml-dtd/go.dtd#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

 <go:version timestamp="Wed May  9 23:55:02 2001" />
 <rdf:RDF>
  <go:term rdf:about=
   "http://www.geneontology.org/go#GO:0003673">
   <go:accession>GO:0003673</go:accession>
   <go:name>Gene_Ontology</go:name>
   <go:definition></go:definition>
```

[1] http://www.geneontology.org/doc/GO.format.html

```
</go:term>
<go:term rdf:about=
 "http://www.geneontology.org/go#GO:0003674">
 <go:accession>GO:0003674</go:accession>
 <go:name>molecular_function</go:name>
 <go:definition>The action characteristic of a gene product.
 </go:definition>
 <go:part-of rdf:resource=
  "http://www.geneontology.org/go#GO:0003673" />
 <go:dbxref>
  <go:database_symbol>go</go:database_symbol>
  <go:reference>curators</go:reference>
 </go:dbxref>
</go:term>
<go:term rdf:about=
 "http://www.geneontology.org/go#GO:0016209">
 <go:accession>GO:0016209</go:accession>
 <go:name>antioxidant</go:name>
 <go:definition></go:definition>
 <go:isa rdf:resource=
  "http://www.geneontology.org/go#GO:0003674" />
 <go:association>
  <go:evidence evidence_code="ISS">
   <go:dbxref>
    <go:database_symbol>fb</go:database_symbol>
    <go:reference>fbrf0105495</go:reference>
   </go:dbxref>
  </go:evidence>
  <go:gene_product>
   <go:name>CG7217</go:name>
   <go:dbxref>
```

```
      <go:database_symbol>fb</go:database_symbol>
      <go:reference>FBgn0038570</go:reference>
     </go:dbxref>
    </go:gene_product>
   </go:association>
   <go:association>
    <go:evidence evidence_code="ISS">
     <go:dbxref>
      <go:database_symbol>fb</go:database_symbol>
      <go:reference>fbrf0105495</go:reference>
     </go:dbxref>
    </go:evidence>
    <go:gene_product>
     <go:name>Jafrac1</go:name>
      <go:dbxref>
       <go:database_symbol>fb</go:database_symbol>
       <go:reference>FBgn0040309</go:reference>
      </go:dbxref>
    </go:gene_product>
   </go:association>
  </go:term>
 </rdf:RDF>
</go:go>
```

Bibliography

(1997). *A Guided Tour to Developing Ontologies Using Ontolingua*. Stanford KSL Network Services. http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/guided-tour/index.html.

(1998). *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*. International Organization for Standardisation.

(2002). *OIModeler Tutorial*. Forschungszentrum Informatik at the University of Karlsruhe. http://kaon.semanticweb.org/docus/.

A. Blackwell & T. Green (2000). 'A Cognitive Dimensions questionnaire optimised for users'. In A. Blackwell & E. Bilotta (eds.), *Proceedings of the Twelth Annual Meeting of the Psychology of Programming Interest Group*, vol. 12, pp. 137–152. http://www.ppig.org/papers/12th-blackwell.pdf.

A. Blackwell & T. Green (2002). 'Notational Systems - the Cognitive Dimensions of Notations framework'. http://www.cl.cam.ac.uk/users/afb21/publications/CarrollChapter.pdf.

M. T. Boren & J. Ramey (2000). 'Thinking Aloud: Reconciling Theory and Practice'. *IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION* **43**(3).

P. Borst, et al. (1997). 'Engineering ontologies'. *International Journal of Human-Computer Studies* **46**(2-3):365–406.

D. Brickley (2000). 'Resource Description Framework (RDF) Schema Specification 1.0'. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

J. Broekstra, et al. (2000). ‘Adding formal semantics to the web: building on top of rdf schema’. http://www.ontoknowledge.org/oil/extending-rdfs.pdf.

G. Canfora & L. Cerulo (2002). ‘A visual approach to define XML to FO transformations’. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pp. 563–570. ACM Press.

F. v. H. Christiaan Fluit, Marta Sabou (2002). ‘Ontology-based Information Visualisation’. In V. Geroimenko (ed.), *Visualising the Semantic Web*. Springer Verlag.

G. O. Consortium (2001). ‘Creating the gene ontology resource: design and implementation’. http://www.genome.org/cgi/content/full/11/8/1425.

J. Converse & S. Presser (1986). *Survey questions. Handcrafting the standardized questionnaire*. Sage Publications.

A. J. Dix, et al. (1998). *Human-Computer Interaction*. Prentice Hall, 2nd edn.

Frank van Harmelen, et al. (2001). ‘Ontology-based Information Visualisation’. In *Proceedings of the workshop on Visualisation of the Semantic Web (VSW’01)*, London. in conjunction with the 5th International Conference on Information Visualisation.

T. Green & A. Blackwell (1998). ‘Cognitive Dimensions of Information Artefacts: a tutorial’. http://www.ndirect.co.uk/ thomas.green/workStuff/Papers/.

T. R. Gruber (1993). ‘Towards Principles for the Design of Ontologies Used for Knowledge Sharing’. In N. Guarino & R. Poli (eds.), *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers.

N. Guarino (1997). ‘Understanding, building, and using ontologies: A commentary to ”Using Explicit Ontologies in KBS Development”’. citeseer.nj.nec.com/guarino97understanding.html.

N. Guarino & P. Giaretta (1995). ‘Ontologies and Knowledge Bases: Towards a Terminological Clarification’. In N. Mars (ed.), *Towards Very Large Knowledge Bases*

- *Knowledge Building and Knowledge Sharing 1995*, pp. 25–32, Amsterdam. IOS Press.

D. M. Hilbert & D. F. Redmiles (2000). 'Extracting usability information from user interface events'. *ACM Computing Surveys* **32**(4):384–421. http://ftp.ics.uci.edu/pub/eden/papers/journals/1999/acmcs/acmcs99.pdf.

P. Holleran (1991). 'A methodological note on pitfalls in usability testing'. *Behaviour and Information Technology* **10**(5):345–357.

I. Human Factors Research Group, Cork (2000). 'Questionnaires in Usability Engineering - A List of Frequently Asked Questions (3rd Ed.)'. http://www.ucc.ie/hfrg/resources/qfaq1.html.

T. Kato (1986). 'What "Question-Asking Protocols" Can Say about the User Interface'. *International Journal of Man-Machine Studies* **Volume 25**(6):659–673.

H. Knublauch (2003). 'An AI tool for the real world: Knowledge modeling with Protégé'. http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html.

O. Lassila & R. Swick (1998). 'Resource Description Framework (RDF) model and syntax specification'. http://www.w3.org/TR/WD-rdf-syntax.

C. Lewis (1982). 'Using the 'thinking-aloud' method in cognitive interface design'. Tech. Rep. RC9265, IBM T.J. Watson Research Center.

A. Maedche, et al. (2003). 'An infrastructure for searching, reusing and evolving distributed ontologies'. In *Proceedings of the twelfth international conference on World Wide Web*, pp. 439–448. ACM Press. http://doi.acm.org/10.1145/775152.775215.

J. Nielsen (1994). 'Estimating the Number of Subjects Needed for a Thinking Aloud Test'. *International Journal of Human-Computer Studies* **41**(3):385–397.

J. Nielsen, et al. (2002). 'Getting access to what goes on in people's heads?: reflections on the think-aloud technique'. In *Proceedings of the second Nordic conference on Human-computer interaction*, pp. 101–110. ACM Press.

J. Nielson (1993). ‘Usability Engineering’. Academic Press.

J. Nielson & R. L. Mack (eds.) (1994). *Usability Inspection Methods*. John Wiley and Sons.

M. A. M. Nieto (2003). ‘An Overview of Ontologies’. Tech. rep., Center for Research in Information and Automation Technologies Interactive and Cooperative Technologies Lab Universidad De Las Americas Puebla.

N. F. Noy & D. L. McGuinness (2001). ‘Ontology Development 101: A Guide to Creating your First Ontology’. Tech. rep., Stanford Knowledge Systems Laboratory.

A. Oppenheim (1992). *Questionnaire design, interviewing and attitude measurement*. Pinter Publishers.

M. Petre, et al. (1997). ‘Cognitive questions in software visualisation’. citeseer.nj.nec.com/petre96cognitive.html.

J. Preece, et al. (1994). *Human-Computer Interaction*. Addison Wesley Publishing Company, 1st edn.

J. Rice, et al. (1996). ‘Using the Web Instead of a Window System’. In *CHI*, pp. 103–110. http://www-ksl-svc.stanford.edu:5915/doc/papers/ksl-95-69/index.html.

S. Riihiaho (2000). *Experiences with Usability Evaluation Methods*. Ph.D. thesis, Laboratory of Information Processing Science, Helsinki University of Technology, http://citeseer.nj.nec.com/riihiaho00experiences.html.

L. Stojanovic & B. Motik (2002). ‘Ontology Evolution within Ontology Editors’. In J. Angele & Y. Sure (eds.), *Evaluation of ontology-based tools*, vol. 62, pp. 53–62.

M.-A. Storey, et al. (2001). ‘Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé’. http://citeseer.nj.nec.com/storey01jambalaya.html.

S. Sudman & N. M. Bradburn (1982). *Asking Questions: A Practical Guide to Questionnaire Design*. Jossey-Bass.

N. Walsh (1998). ‘What is XML?’. http://www.xml.com/pub/a/98/10/guide1.html.

S. Wilson, et al. (1997). ‘Helping and hindering user involvement - a tale of everyday design’. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 178–185. ACM Press.

M. Winston, et al. (1987). ‘A Taxonomy of Part-Whole Relations’. In *Data and Knowledge Engineering*, vol. 20, pp. 259–286. Hamilton College.

D. Wixon & C. Willson (1997). ‘The usability engineering framework for product design and evaluation’. In M. G. Helander, T. Landauer, & P. Prabhu (eds.), *Handbook of Human-Computer Interaction*, pp. 653–688. Elsevier Science North-Holland.