Relieving the Cognitive Load of Constructing Molecular Biological Ontology Based Queries by means of Visual Aids

by

Kieran O'Neill

BSc (University of Natal) 2003
BSc(Hons) (University of KwaZulu-Natal) 2004

A dissertation submitted in fulfilment of the
requirements for the degree of

Masters

in

Computer Science

in the

School of Computer Science
of the
UNIVERSITY OF KWAZULU-NATAL

Supervisor:
Professor Hugh Murrell

Co-Supervisors:
Mister Daniel Jacobson
Doctor Alexander Garcia-Castro

2007

Preface

Submitted in fulfilment of the academic requirements for the degree of Master of Science (M.Sc.) in the School of Computer Science, University of KwaZulu-Natal, Pietermaritzburg Campus.

This thesis represents original work by the author and has not been submitted in part or whole to this or any other university.

Most of this work was carried out at the Central Node of the National Bionformatics Network under the supervision of Alexander Garcia and Dan Jacobson.

Some of this work has been presented at bioinformatics conferences and appears in conference proceedings. This is summarised in Appendix I. Any use of the work of others has been suitably referenced.

URLs to pages demonstrating the software developed are provided in the text. The software has currently only been tested on the freely available Firefox web browser, and may not function on other web browsers.

Kieran O'Neill (Candidate)

Professor Hugh Murrell (Supervisor)

Abstract

The domain of molecular biology is complex and vast. Bio-ontologies and information visualisation have arisen in recent years as means to assist biologists in making sense of this information. Ontologies can enable the construction of conceptual queries, but existing systems to do this are too technical for most biologists. OntoDas, the software developed as part of this thesis work, demonstrates how the application of techniques from information visualisation and human computer interaction can result in software which enables biologists to construct conceptual queries.

Acknowledgments

I would like to thank Alexander Garcia Castro, Anita Schwegmann, Hugh Murrell, Dan Jacobson and Rafael Jiménez Doménech for their contributions towards finishing this thesis.

Alex, thank-you for sage guidance, advice and constructive criticism. I've learned a lot this past year.

Anita, thank-you for being an incredibly helpful and accommodating domain expert, and for providing biological context for OntoDas. I look forward to (hopefully) working with you in the future.

Rafa, thank-you for your help with integrating Dasty, friendship and free Chinese food from Panda's.

Hugh and Dan, thank-you for handling the NBN-UKZN collaboration, providing support and sorting out political hiccoughs.

This work was funded by the National Research Foundation in the form of a Prestigious Award for Masters study. Additional funding and extensive logistical support was provided by the National Bioinformatics Network. Thanks go out to both organisations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Problem Statement

1.1 Biological information integration

Biological information is highly nested, interconnected, distributed and also heterogeneous both semantically and syntactically. In recent years, formal bio-ontologies have arisen and been used to annotate biological databases, in an effort to aid information integration across them. In parallel, research from the field of information visualisation, which aims to provide cognitive support to users working with abstract information, has begun to be applied to the biological domain. Both of these efforts aim to assist biologists in finding information within their knowledge domain pertaining to their specific sphere of interest, to guide them in choosing the directions of their research.

The Gene Ontology (GO), among others, has been widely used to annotate gene products in terms of their functions. When a set of gene products shares GO annotations, this suggests that they may be functionally related. Furthermore, finding gene products having specific functions that are of interest to a biologist may help them in choosing targets for further study. Although tools exist for executing the kinds of queries against GO that could produce this information, they tend to have complicated, highly technical user interfaces, often using custom query languages, which most biologists do not have the time or inclination to learn. Masking the complexity of the process of constructing such queries, so as to make them more available to biologists, remains an open challenge.

1.2 Problem statement

"Although the standard usage of ontologies within molecular biology provides a basis for the construction of conceptual queries within the domain by the domain experts, no visual tool exists to enable them to do this."

Software currently exists to enable the construction of conceptual, ontology-based queries within the molecular biology domain. However, this software incurs high cognitive load when using it, making it inaccessible to most molecular biologists and other potential users. A key factor when solving this problem is to provide a software tool with a user interface which relieves the cognitive load incurred when constructing ontology-based molecular biological queries.

1.3 Thesis outline

This thesis describes the development of OntoDas, a visual, web-based tool which facilitates the construction of ontology-based molecular biological queries. The thesis commences in chapter 2 with a broad review of the literature pertaining to information integration, particularly in the context of the biological domain. The kinds of queries which can be facilitated by ontologies are presented with reference to the literature, as well as the desirability of relieving cognitive load. The suitability of human computer interaction and information visualisation techniques for relieving cognitive load is presented. This is followed by chapter 3, containing a survey and comparison of specific techniques and software tools related to the problem. In particular this chapter shows other methods of constructing similar queries; this survey is used later as a contrast to OntoDas. The process of designing OntoDas is described in chapter 4, which shows how specific scenarios were developed into a generic use case, and thence into paper prototypes suggesting functionality and screen layout for the final tool, all of which was carried out with the participation of a biologist. Chapter 5 discusses the technical issues involved in implementing OntoDas, and illustrates the final appearance of the system. Technical limitations and performance are also briefly presented. Chapter 6 compares and contrasts OntoDas with the systems surveyed in chapter 3 to show how it contributes new functionality. An argument from the literature is presented that the use information visualisation and participatory design in OntoDas have enabled it to solve the central problem of this thesis. The limitations of OntoDas and of

the process used to create it are discussed as a prelude to the proposal of future work. The future work itself is presented in chapter 7. Further functionality which might improve OntoDas' ability to relieve cognitive load is discussed. Options for a more comprehensive usability study and a participatory design approach are proposed. Finally, the applicability of OntoDas to new domains and ontologies is discussed. The thesis ends with a restatement of the conclusions drawn and the contributions made in chapter 8.

Chapter 2

Literature Review: Ontologies and Information Visualisation for Biological Information Integration

2.1 The problem of semantic heterogeneity of biological information

Within the biological domain, there has been a shift from hypothesis-driven research, wherein data is collected purely to answer a scientific question, to data-driven research, wherein large data sets are collected and made publicly available for analysis and interpretation [90]. This has resulted in an explosion in the amount of molecular biological data that is publicly available. This data is stored in at least 858 databases [35], using differing formats, schemata and query software [109]. To enable biologists to fully leverage this data, and the information it contains, the integration of data from disparate sources is essential.

The syntactic, or 'low level' [90] integration of data is a problem that has been addressed by systems such as Sequence Retrieval Service (SRS)[31], Entrez [88], Distributed Annotation System (DAS) [27] and others which overcome heterogeneity in the structure and representation of data [37]. In the case of sequence data, for instance, the sequence itself is unambiguous and can be analysed and searched mathematically and computationally once it has been extracted from the database or file in which it has been stored [13].

However, biological databases also store knowledge about the data they contain, in the form of inferences made by researchers about the data. This knowledge is represented in a wide variety of lexical forms [13] – researchers often have different systems of representation, which may or may not be easy for other researchers to understand – raising a need for the further integration of the knowledge and meaning assigned to biological data. This is usually called semantic [37] or 'higher level' [90] data integration.

To understand this further, it is important to define what is meant by "knowledge". Davenport and Prusak suggest the following working definition:

Definition 1. *"Knowledge is a mix of framed experience, values, contextual information, expert insight and grounded intuition that provides an environment and framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organisations, it often becomes embedded not only in documents or repositories but also in organisational routines, processes, practices and norms." [26]*

This definition arises from the field of knowledge management (KM), which is a subdivision of business management concerned with supporting knowledge transfer within business organisations [99], but can be extended to broader knowledge-sharing communities: Lave and Wenger have named these "communities of practice" [64]. A community of practice is a group which shares a sense of *joint enterprise* (a collective understanding of the community's goals), *mutuality* (social and formal interactions between community members) and a *shared repertoire* of communal resources, such as language, techniques and artifacts [106]. The "shared repertoire" referred to here closely resembles the definition provided above for knowledge: Communities of practice are held together in part by their common knowledge.

Biological researchers form a community of practice: They have a joint enterprise in their research, formal and social interactions in the form of conferences, research collaborations and informal friendships, and a shared repertoire of knowledge, in the form of research techniques, terminology, the scientific literature and more recently in the numerous and diverse publicly available databases mentioned above. Within this community, there are numerous sub-communities defined by specialised conferences, journals and fields of research, due to the high level of complexity of biological systems. Consequently, the repertoire of biological knowledge and information is enormous and highly semantically complicated. Despite the divisions and specialisations of the field, the systems studied

by biological sub-communities interact in reality. As an illustration, the DNA studied by molecular geneticists encodes and influences the concentrations of the proteins studied by biochemists, and these in turn are responsible for the metabolic processes of individual cells, which affect the overall physiology of the organism they are a part of (in the case of multi-cellular organisms). A shift towards more integrative or "systems thinking" approaches to biological research has begun a drive towards semantic information integration in the biological domain [90]. For this to occur, biological knowledge must be captured in a standardised form which can be understood by both humans and computer software. This in turn has led to a recent trend of creating and using bio-ontologies [13].

2.2 Bio-ontologies

Ontologies provide a means to facilitate semantic integration by providing a common lexicon for the representation of knowledge [13]; by acting as formalisations of knowledge, ontologies support knowledge sharing and reuse [37]. Definitions of ontology differ slightly among experts from different fields, even within bioinformatics [40]. The term originated in the field of philosophy, wherein an ontology is considered to be *"a conception of reality or simply reality itself"* [13], or in the sense of formal ontology, *"theories of the different types of entities (objects, processes, relations) existing in a given domain"* [94]. Information scientists on the other hand, tend to view ontologies as being terminologies with associated axioms and definitions, structured so as to support software applications [94], or in more detail, as *"vocabularies of representational terms, classes, relations, functions and object constants with agreed-upon definitions in the form of human-readable text and machine-enforceable, declarative constraints on their well-formed use"* [43]. Ontology development in the biomedical domain, however, has tended to focus on capturing of biological concepts as rapidly as possible, at the expense of some of the formalisms considered necessary by information scientists for them to be considered true ontologies [94] [13]. Consequently, a spectrum of knowledge formalisms exist in the domain of biological research with the purpose of helping researchers understand that domain, ranging from unstructured controlled vocabularies of terms (ie. lexica without defined relations between terms), to controlled vocabularies with inheritance relations (but no other relations), to fully formal ontologies in the information science sense [13]. The bio-ontologies relevant to the work of this thesis are primarily controlled vocabularies of biological terms describing biological con-

cepts, with inheritance relationships between pairs of terms, forming either directed acyclic graph (DAG) or tree structures. Although not strictly ontologies [13], they will be called such for the purposes of this thesis.

Definition 2. *Bio-ontologies are taxonomic representations of knowledge about biology, which organise concepts within specialisation hierarchies that provide for inheritance of attributes (after [90]).*

Gene Ontology (GO) [6] is probably the best-known, most widely used and accepted of the bio-ontologies [40]. GO captures the functions of gene products in terms of their involvement in *biological processe*s, the *cellular component* they function in, and their *molecular function*. GO has been used to annotate genes across multiple organisms, and thus can aid in cross-organism semantic integration. In addition to GO, many model organism genome projects, such as the mouse (*Mus musculus*)[11], fly (family *Drosophilidae*)[28] and worm (*Coenhorabdtis elegans*) [89], have created their own domain ontologies for capturing knowledge about their organism, such as anatomy ontologies for capturing the location of gene expression, and phenotype ontologies for capturing the effects of gene knockout [95]. Other bio-ontologies include those intended to provide uniform means to describe experimental results, such as the Microarray Gene Expression Data Ontology (MGED), which captures terms describing microarray experiments [96]. An ontology with a slightly different purpose is the TAMBIS (transparent access to multiple biological information sources) ontology [41], which serves as a global schema over multiple heterogeneous resources, and provided the semantic underpinnings of the TAMBIS query interface.

The importance to biological research and the acceptance of ontologies within the domain of knowledge is clear: A total of 64 community-developed biomedical ontologies are currently openly available in a common, openly defined syntax (the OBO file format) at the Open Biomedical Ontologies (OBO) repository [78]. Moreover, in 2006, a National Institute for Biomedical Ontology was established in the United States with the specific purpose of creating and maintaining bio-ontologies, software tools to aid ontology creation and software tools which leverage the power of ontologies [85]. Thus far in biology, ontologies have been used primarily to deliver vocabularies for describing data. In the future, however, it is likely that the increasing formality of bio-ontologies will result in greater analysis of data [13].

2.3 Knowledge bases and inference

Knowledge bases (KB) are collections of facts modelling some part of the world [71], and provide a means of capturing the information repertoire for a domain. Bio-ontologies are seen as being able to provide rich, hi-fidelity models of biology, and thereby a means of forming knowledge bases [13], such as EcoCyc [57] and RiboWeb [5]. Knowledge bases differ from conventional databases in that both abstract and concrete knowledge are stored in a common schema, whereas in databases these are kept separate [72], such as in relational databases, wherein the database schema, representing entities and their possible relationships, is kept separate from the instances of those entities, the data records [23]. This aids the purpose of building KBs: to realise some of the problem-solving, or inference capabilities possessed by domain experts (or community of practice members) [98].

To carry out this inference, knowledge bases may be created within knowledge base management systems which provide a reasoner (a set of algorithms implementing inference operations), such a RACER [44]. Reasoners require the knowledge they work with to be formally represented. Common formalisms include description logics, such as the $SHIQ$ DL used by RACER [48] and the frames paradigm used by Protégé [77]. These formalisms are class-based (concept-based), and thus provide a means of representing ontologies, with the additional capability of enabling inference operations [47]. A Description Logic knowledge base consists of a set of axioms asserting, for example, that one class is a subclass of another, or that an individual is an instance of a particular class [47]. An example of an inference operation is the deduction of these subsumption relations based on the sets of instances of classes [44]: For example, if all instances of the class "human" also belong to the class "animal", then human is a subclass of animal [47]. More complex inference can include reasoning with numbers: If a "seriously ill human" is defined to be a human with a temperature greater than 40.5°C and a "human with fever" is defined to be a human with a temperature greater than 38.5°C, then the reasoner may be able to deduce that "seriously ill human" is a subclass of "human with fever" [45]. Description logics are discussed in more detail, in the context of GO, in the next section.

2.4 Inferring functional relationships among genes

A potentially useful task in molecular biology is the discovery of functionally related genes or proteins. One means to discover such sets is by their common annotation with ontology terms describing their function.

In this section, a series of inference operations, or queries, are described which can aid in the discovery of genes or proteins sharing ontology annotations, and hence likely to be functionally related.

The inheritance relations in the DAG ontologies considered in this thesis are transitive:

Definition 3. *A relation R defined on a set S is transitive provided that:*

$$\forall x, y, z \in S, xRy \bigwedge yRz \Rightarrow xRz$$

[105]

For example, considering the GO terms "transcription factor binding", "DNA binding" and "nucleic acid binding": transcription factor binding is a type of DNA binding and DNA binding is a type of nucleic acid binding, thus transcription factor binding is a type of nucleic acid binding.

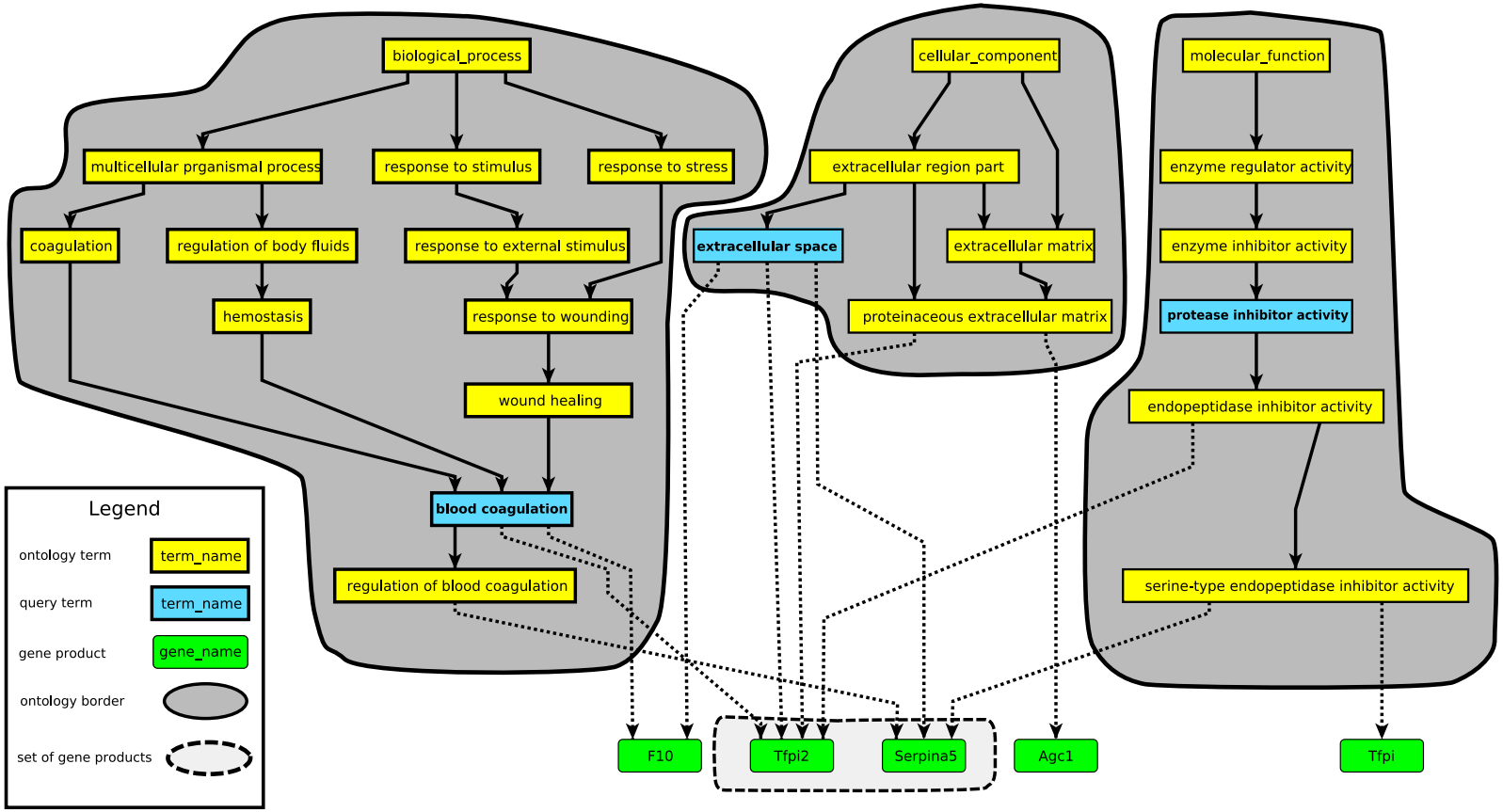

Figure 2.1: An illustration of the structure and annotations of the ontologies for an example query.

One of the more useful queries over ontologies, taking advantage of this transitivity, is the successor set query [101]. This query retrieves all terms which derive from a given term (via transitive inheritance relations such as is_a and part_of). For example, a researcher interested in proteins involved in nucleic acid binding would not just be interested in those proteins annotated by the GO term "nucleic acid binding", but also in those annotated by the immediate children of that term, their children, their children's children, and so on [101]. Using the example shown in Figure 2.1, the successor set of the term *response to wounding* would be {*wound healing, blood coagulation, regulation of blood coagulation*}. The successor set can be formally defined:

Definition 4. *Given an ontology defined as a digraph consisting of a set of terms T and a set of edges representing the transitive inheritance relation I on T, the successor set $S_t \subseteq T$ for any given $t \in T$ is defined:*

$$s_t \in S_t \Leftrightarrow tIs_t$$

However, a researcher's goal is less likely to be finding all the subfunctions of nucleic acid binding, in other words terms in GO (although these may be of interest), but rather in the proteins annotated by these. Proteins annotated with GO terms can be considered to be instances of the functionality described by the terms. Thus, for instance, in Figure 2.1, the set of instances of the term *extracellular region part* is {*F10, Tfpi2, Serpina5, Agc1*}, since the first three are annotated with *extracellular space*, *Tfpi2* and *Agc1* are annotated with *proteinaceous extracellular matrix*, and these two terms are successors of *extracellular region part*. Finding the instances of a term can be defined:

Definition 5. *Given a successor set S_t for a given term t, and given a set of instances, C of T, related by an instance relation R on $C \times T$, the instance set $C_t \subseteq C$ of t is defined:*

$$c \in C_t \Leftrightarrow \exists s_t \in S_t; cRs_t$$

Also of great interest in ontologies are the relations between and among terms that are not formally captured in the ontology, but are implicitly defined [12]. One way in which these might be implied is if the related terms share a set of common instances.

For example, the GO molecular function term "protease inhibitor activity", the GO biological process term "blood coagulation" and the GO cellular location term "extracellular space" share a set of genes which they annotate. This implies that some part of the process

of blood coagulation occurs in the extracellular space, and that this part involves protease inhibition. Furthermore, the common set of gene instances would contain the gene or genes responsible for this functional relationship. Finding commonly annotated sets of genes can be defined:

Definition 6. *Given a set of ontology terms* $\{t_1, t_2, \cdots\}$*, and their respective instance sets,* $C_{t1}, C_{t2}, \cdots$*, the set of common instances,* C_c *for those terms is:*

$$C_{t1} \cap C_{t2} \cap \cdots$$

Figure 2.1 illustrates all of these operations using a small subset of GO and a few genes from the GO annotation database. The query is to find products related to the terms *blood coagulation*, *extracellular space* and *proteinase inhibitor activity*. The gene products returned are *Tfpi2* (tissue factor protein 2) and *Serpina5* (serine protease inhibitor, clade A, member 5). *Serpina5* is annotated with the term *serine-type endopeptidase inhibitor activity*, which is a descendent term of the query term *protease inhibitor activity*, and is included in the results due to the transitivity of the inheritance relations within the ontology. If the former were substituted for the latter to make the query more specific, only *Serpina5* would be returned, as *Tfpi2* is not annotated to the more specific term. In a similar way, if the term *extracellular space* were replaced with *proteinaceous extracellular matrix*, only *Tfpi2* would be returned. Note that *F10* (blood coagulation factor X) is not included in the results, as it is only annotated with 2 out of the 3 query terms. However, if *protease inhibitor activity* were removed, *F10* would become part of the result set. Finally, the orthogonality of the ontologies is illustrated: No relations exist between terms from different ontologies. Implicit relations can, however, be inferred via the annotation of genes with terms from orthogonal ontologies.

These kinds of queries can be executed using scripts, tools such as spreadsheets, and in a few cases visual tools [92]. This can be an arduous and difficult process, however.

2.5 The need for cognitive support in biological information integration

As this chapter has so far illustrated, the integration of biological information presents a serious and difficult problem to the field of molecular biology. Much effort has

been made to tackle this problem, and systems and structures have been created to facilitate the integration of heterogeneous biological information. However, many of these systems have extremely verbose, difficult to use interfaces, often involving custom query languages or requiring programming skills, and return highly complex and verbose result sets. Mentally processing the amount of information required to construct such queries, and the amount of information returned from their execution is usually more than a human being is capable of, due to cognitive limitations such as the relatively small size of our working memory [20]. A need therefore exists for tools which provide cognitive support to users trying to make effective use of biological information integration systems.

Cognitive support (sometimes called external or distributed cognition) is the augmentation of human cognition using artifacts external to the human mind. A trivial example is the use of a pencil and paper to aid a person in working out long multiplication [20]. Storing the numbers externally from the calculator's mind (on the paper) relieves the cognitive load on their short-term memory. The use of external artifacts to aid cognition is pervasive in everyday life for most people [75]. In the case of biological information integration, users are dealing primarily with abstract data (data which is non-numerical and has no obvious spatial mapping, such as that encoded by ontologies.) The field of information visualisation specifically aims to provide cognitive support when dealing with this kind of information.

2.6 Information visualisation

Information visualisation (IV) has been defined as *"the use of computer-supported, interactive, visual representations of abstract data to amplify cognition"* [20]. This is accomplished by visually organising data, using its information structure, so as to take advantage of the human brain's capacity for rapid, subconscious visual processing. Details on some ways in which information visualisation provides cognitive support are summarised in Table 2.1. The essence of all these techniques is that information visualisations aim to provide cognitive support by transferring to the computer some of the cognitive load of dealing with abstract data.

In the context of biological information integration, and specifically of finding functional relationships among genes and proteins, users need to be able to *discover* and *select* the terms used to execute the query, and to *explain* and *understand* the results. The goal of IV is to facilitate the gaining of insight, specifically in these very tasks of discovery,

decision making and explanation [20].

Method	Description
Increased Resources	
Parallel processing	Parallel processing by the visual system can increase the bandwidth of information extraction from the data.
Offload work to the perceptual system	With an appropriate visualisation, some tasks can be done using simple perceptual operations.
External memory	Visualisations are external data representations that can reduce demands on human memory.
Increased storage and accessibility	Visualisations can store large amounts of information in an easily accessible form.
Reduced Search	
Grouping	Visualisations can group related information for easy search and access.
High data density	Visualisations can represent a large quantity of data in a small space.
Structure	Imposing structure on data and tasks can reduce task complexity.
Enhanced Recognition	
Recognition instead of recall	Recognising information presented visually can be easer than recalling information.
Abstraction and aggregation	Selective omission and aggregation of data can allow higher level patterns to be recognised.
Perceptual Monitoring	Using pre-attentive visual characteristics allows monitoring of a large number of potential events.
Manipulable Medium	Visualisations can allow interactive exploration through manipulation of parameter values.
Organisation	Manipulating the structural organisation of data can allow different patterns to be recognised.

Table 2.1: Ways in which visualisation can support cognition. After [100], summarised from [20].

```
<?xml version='1.0' standalone='no' ?>
<!DOCTYPE DASGFF SYSTEM 'dasgff.dtd' >
<DASGFF>
  <GFF version="1.0" href="http://www.ebi.ac.uk/das-srv/uniprot/das/aristotle/features?segment=Q24488">
    <SEGMENT id="Q24488" version="0ed4aef73895159f394610331d72b006" start="1" stop="685">
      <FEATURE id="Q24488" label="Q24488">
        <TYPE id="description" category="">description</TYPE>
        <METHOD id="description">description</METHOD>
        <START>0</START>
        <END>0</END>
        <SCORE>-</SCORE>
        <ORIENTATION>0</ORIENTATION>
        <PHASE>-</PHASE>
        <NOTE>Tyrosine-protein kinase transmembrane receptor Ror precursor (EC 2.7.10.1) (dRor).</NOTE>
        <LINK href="http://www.ebi.uniprot.org/uniprot-srv/uniProtView.do?proteinAc=Q24488">http://www.ebi.uniprot.
        org/uniprot-srv/uniProtView.do?proteinAc=Q24488</LINK>
      </FEATURE>
      <FEATURE id="ROR1_DROME_SIGNAL_1_24" label="ROR1_DROME_SIGNAL_1_24">
        <TYPE id="SIGNAL" category="Molecule Processing">SIGNAL</TYPE>
        <METHOD id="UniProt">UniProt</METHOD>
        <START>1</START>
        <END>24</END>
        <SCORE>-</SCORE>
        <ORIENTATION>0</ORIENTATION>
        <PHASE>-</PHASE>
      </FEATURE>
      <FEATURE id="PRO_0000024462" label="ROR1_DROME_CHAIN_25_685">
        <TYPE id="CHAIN" category="Molecule Processing">CHAIN</TYPE>
        <METHOD id="UniProt">UniProt</METHOD>
        <START>25</START>
        <END>685</END>
        <SCORE>-</SCORE>
        <ORIENTATION>0</ORIENTATION>
        <PHASE>-</PHASE>
        <NOTE>Tyrosine-protein kinase transmembrane receptor Ror/FTId=PRO_0000024462</NOTE>
      </FEATURE>
      <FEATURE id="ROR1_DROME_TOPO_DOM_25_317" label="ROR1_DROME_TOPO_DOM_25_317">
        <TYPE id="TOPO_DOM" category="Regions">TOPO_DOM</TYPE>
        <METHOD id="UniProt">UniProt</METHOD>
        <START>25</START>
        <END>317</END>
        <SCORE>-</SCORE>
        <ORIENTATION>0</ORIENTATION>
        <PHASE>-</PHASE>
        <NOTE>Extracellular</NOTE>
      </FEATURE>
```

Figure 2.2: A portion of the DAS XML representation of the annotations for protein with UniProt ID Q24488, from the UniProt database. The file is a total of 621 lines long.

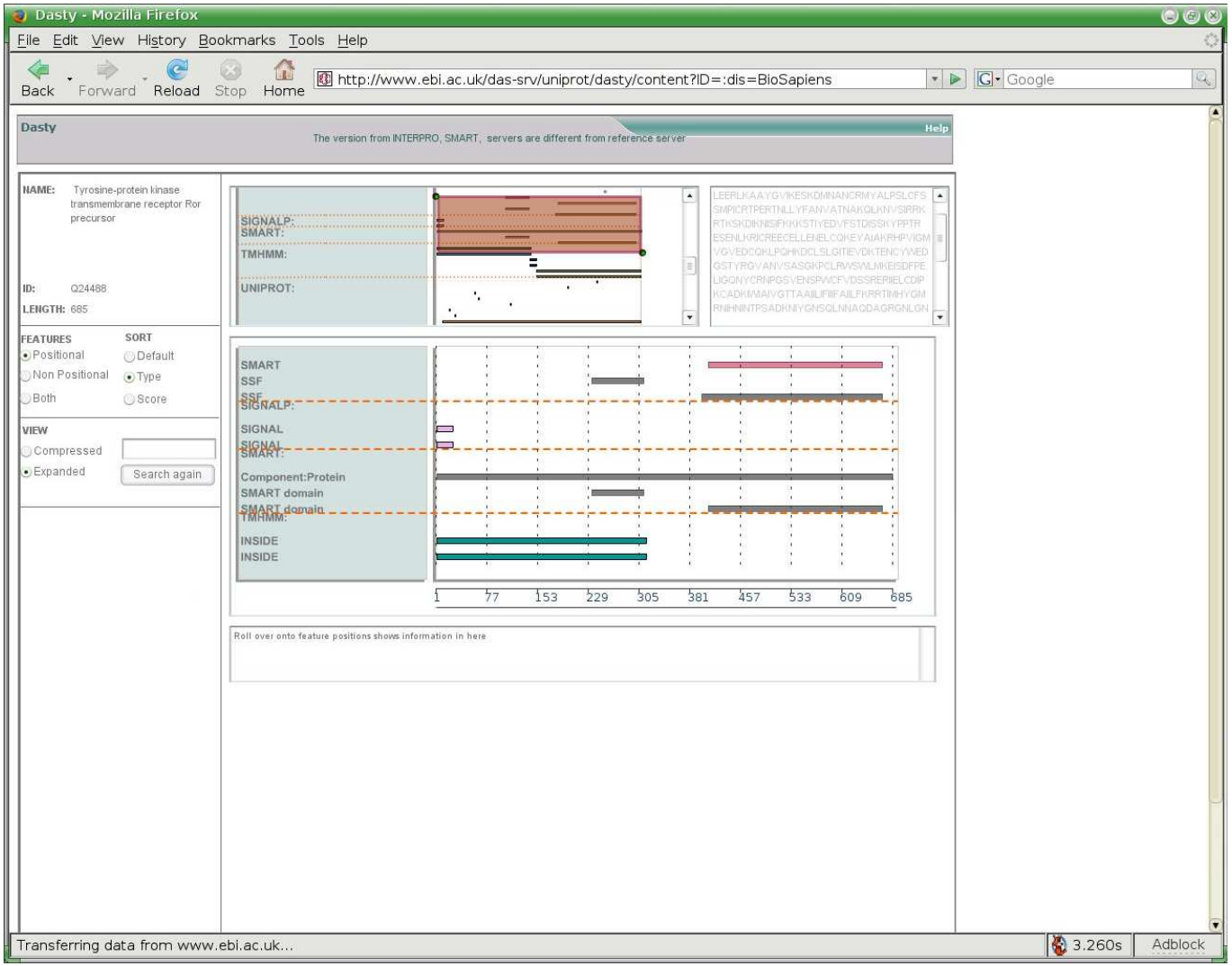

Figure 2.3: A screen shot of Dasty [55], a tool for visualising DAS output. This view is much easier for a user to handle than raw XML.

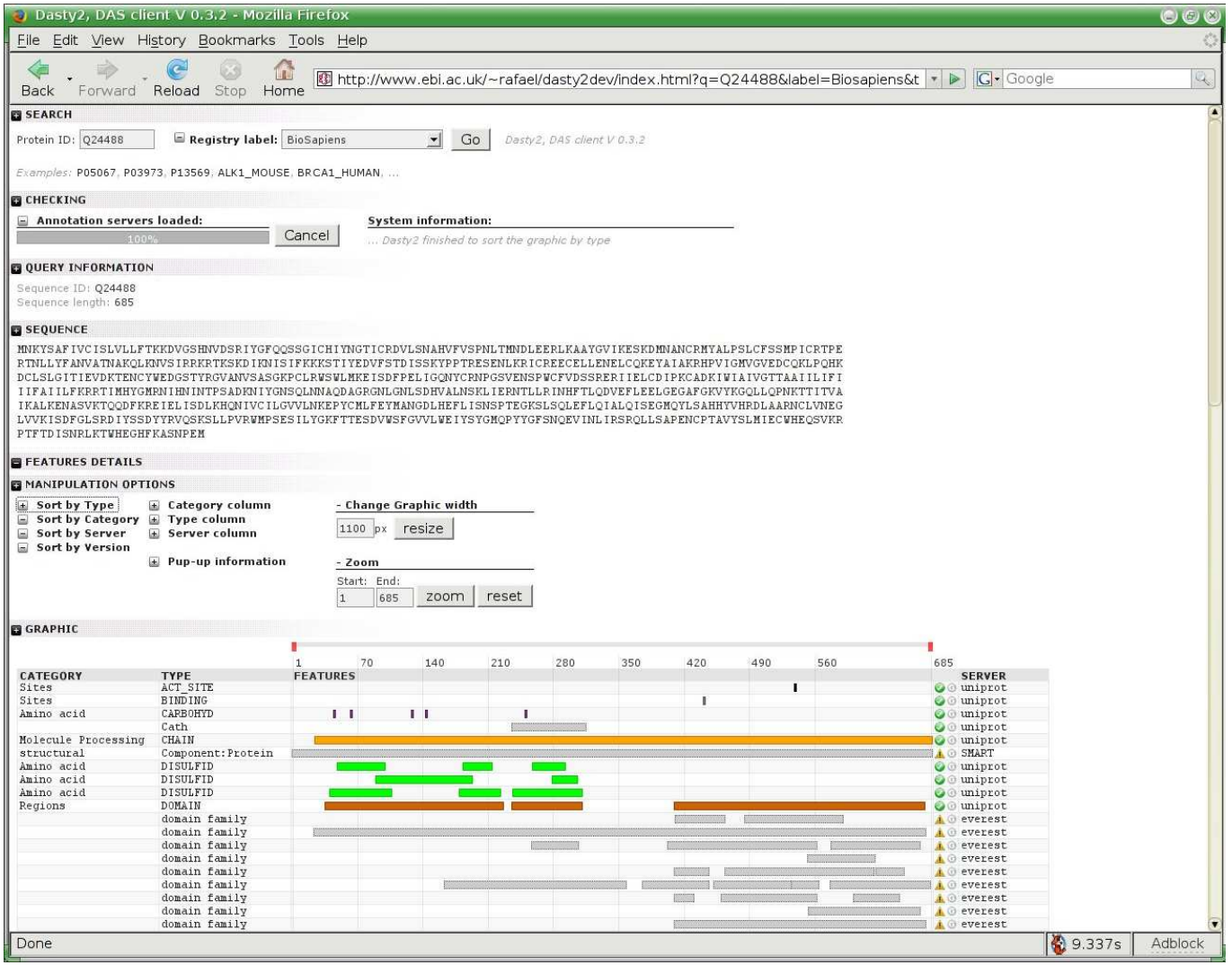

Figure 2.4: A screen shot of Dasty2, the next generation of Dasty. Tasks such as sorting all annotations (from all servers) by type are now supported, whereas in Dasty a user would have to compile this list themselves.

2.7 DAS and Dasty2

2.7.1 Distributed Annotation System

DAS (distributed annotation system) is a system to enable transparent access to multiple repositories of annotations to biological sequences [27]. DAS uses an XML (extensible markup language) representation to deliver data from heterogeneous sources in a homogeneous way. This representation, although easy to parse for computers, is highly verbose and difficult for humans to read, as illustrated in Figure 2.2. The file is much longer than what is shown, and DAS queries usually result in several such files, creating data sets far beyond human cognitive capacity to comprehend in their raw form.

2.7.2 Dasty2: An example of cognitive support for biological information integration

To help biologists make use of this information effectively, clients such as Dasty have been developed [55]. As shown in Figure 2.3, Dasty represents features as bars along a common axis representing the whole sequence, with colours to distinguish different feature types, and a separate view showing which part of the sequence a selected annotation corresponds to. Dasty thus provides support by representing a large amount of information in a small space, and by providing some grouping functionality to aid finding related features, and is clearly an improvement over the raw XML view.

However, Dasty presents some issues: For example, the annotations are always grouped by server. While it is possible to group annotations by type, this can only be done within each server grouping. If a user wants to find all features of a given type, regardless of server, they either have to remember the features of that type from each server as they view them, or use some other external aid, such as a paper and pencil or text editor to compile such a list. Dasty2 [54], the next generation of Dasty, includes this capability, thus providing improved cognitive support over Dasty, as shown in Figure 2.4.

2.8 Discussion

In the biological research domain, syntactic and semantic information integration are problems with existing solutions. Federated systems such as DAS and data warehouses

such as Ensembl and the NCBI database provide syntactically normalised access to information from diverse sources [109]. Bio-ontologies and the tools which make use of them help to overcome semantic discrepancies among diverse information sources, and to provide easy access to biological knowledge. However, the biologist who is the user of these systems has often been overlooked in their design and creation. The human-computer interface between the domain expert and the information integration system can be considered to be a third facet to information integration. The creation of tools which provide biologists with the cognitive support they need to make effective use of information integration systems is a problem which has been poorly addressed in bioinformatics. The careful application of insights and methodologies from the field of information visualisation, and its related fields, could help to provide usable, supportive tools which enable biologists to effectively integrate biological information.

Chapter 3

The State of the Art in Facilitating Ontology-Based Query Construction

3.1 Introduction

As argued in the previous chapter, the construction of ontology-based queries can be facilitated by the use of information visualisation. A range of methodologies exist for the design of human-computer interaction systems in general, with varying degrees of direct and indirect applicability to information visualisation. These are surveyed and compared in section 3.2. The following section, 3.3 briefly discusses existing methods for visualising ontologies, and explains why these provide a poor basis for solving the problem tackled by this thesis. Finally, section 3.4 provides an alternative source of insight into the problem in the form of a survey of software tools which do enable the execution of queries of the kind described in section 2.4.

3.2 Methodologies for designing IV systems

3.2.1 Introduction

To date, a comprehensive methodology specifically for designing IV systems has not been proposed, although work has begun on components of such a methodology. In the

related fields of human-computer interaction (HCI) and software engineering (SE), however, comprehensive methodologies exist:

Within HCI, the widely accepted User-Centred Design (UCD) paradigm [76] places focus on involving users in the design of the system as much as possible. Methodologies within UCD include Nielsen's usability engineering (UE) [73] and the Scandinavian-originated participatory design (PD) [68], [15]. The primary difference between the two is in the way users are involved: Within UE, users are viewed as a source for requirements gathering and as test subjects for usability testing [1]. By contrast, PD methods tend to involve users actively in the process of developing software. Neither are prescriptive, however, and there is room for crossover, such as the incorporation of PD as a first phase of UE [79].

Within SE, the iterative and incremental development model is accepted as a best practice [62]. This model emphasises a software life cycle of iterative releases, with evolutionary advancement occurring between releases [63]. Examples of methodologies using this model include the detailed but somewhat heavy-weight unified process (UP) [51] and the lighter, more "agile" [22] extreme programming (XP) [8]. These methodologies have a similar view of user involvement: Requirements are gathered from users in the form of user stories (in XP), or use cases (in UP), which constitute specific stories of using the system, and abstractions distilled from these stories, respectively [50]. The overall focus, however, is on developing the software as a whole, and the usability of the interface is often neglected [91]. This is illustrated in a recent development, the proposal of design patterns (general repeatable solutions to a commonly occurring problem in software design [36]) for information visualisation [2]: The patterns presented focus almost exclusively on designing the software implementation underlying an information visualisation rather than on the visualisation itself.

3.2.2 Participatory design for the biological domain

Given the complexity and uniqueness of the biological domain, discussed in 2.1, a methodology which focuses on ensuring that software fits biologists' way of working is likely to be an advantageous choice for developing bioinformatics software. Of the methodologies outlined above, only participatory design focuses specifically on ensuring that the software developed fits the needs and work practices of the users [15]. This makes PD a good

potential candidate methodology for developing bioinformatics software.

PD has been used both in the context of designing information visualisation systems and in the context of bioinformatics. In IV, PD has been employed to develop a risk management visualisation system at IBM [10]. Additionally, a study has been conducted in which users engaged in PD exercises to design digital library web interfaces incorporating IV techniques [112]. Within bioinformatics, PD has been used as the basis for "participatory programming" tools to support biologists in carrying out basic programming tasks [66]. It has also been successfully used to develop bio-ontologies [38].

3.3 Ontology visualization

A plethora of techniques exist for visualising ontologies, many of which have been recently reviewed in [59]. These range from techniques such as focus and context [61] and treemap [7] to exotic three dimensional "information landscapes" [32]. However, despite the great efforts undertaken to create diversity in the suite of techniques, evaluation of their suitability to specific tasks has been widely overlooked [59, 21]. Furthermore, very few of the tools created specifically to visualise ontologies provide support for reasoning over the ontologies [59], which is an essential requirement for the execution of the types of queries detailed in section 2.4. Indeed, the information visualisation reviews tend to overlook visual tools specifically aimed at assisting users when constructing ontology-based queries, such as the Drug Ontology Project for Elsevier (DOPE) [34],[17] and Flamenco [111], neither of which are mentioned in the otherwise comprehensive review by Katifori *et al* [59]. More comprehensive evaluations of these techniques are in progress [3], but final results have yet to be published. In some studies which have been undertaken, the "baseline" visualisation technique for ontologies, a collapsible tree such as found in file explorers, has been shown to outperform more sophisticated techniques for assisting users to find terms [81]. In that specific study, "information scent" (*the (imperfect) perception of the value, cost, or access path of information sources obtained from proximal cues [80]*) was shown to have a greater effect on search times than the choice of visualisation. In conclusion, a review of existing tools which enable the construction of ontology-based queries is likely to be more applicable to the central problem of this thesis.

3.4 Software for constructing ontology-based queries

3.4.1 Introduction

As in the broader domain of ontology visualisation, within the context of bio-ontologies the focus when developing tools for visualising ontologies has been on those which facilitate the browsing of ontologies, such as AmiGO [1] and those which display gene expression information, such as FatiGO [4] and GOMiner [113]. A few broader information integration tools, such as BioMart [58] and its web interface, MartView [46] provide the functionality for constructing ontology-based queries, although this is not their primary focus. One tool, GViewer [92], enables the construction of ontology-based queries over the information in the rat (*Rattus norvegicus*) genome database [103]. Outside of the biological domain, visual tools specifically aimed at constructing ontology-based queries have been created, including the Drug Ontology Project for Elsevier (DOPE) [34],[17] created by Aduna software and Flamenco [111]. No tool exists specifically for visually facilitating the construction of ontology based queries within molecular biology.

Five tools representative of the spectrum of tools facilitating the construction of ontology-based queries have been chosen for comparison in this section: AmiGO, the MartView interface to BioMart, GViewer, DOPE and Flamenco. AmiGO has been chosen as a baseline tool, as it is the standard viewer for the Gene Ontology, and likely to be familiar to most biologists interested in GO. BioMart and GViewer have been included in the comparison because they enable the execution of ontology based queries specifically within the domain of molecular biology, although they provide very little visual support for this process. DOPE and Flamenco have been included because they provide visual support for the query construction process, although the ontologies they have been used for lie outside the domain of molecular biology. The relative strengths of these tools, when discovered by comparison, can suggest features desirable in a tool for visually facilitating the construction of ontology based queries within molecular biology.

3.4.2 Criteria used for comparison

1. **Problem domains:** This specifies the domains of knowledge to which the tool has been applied. This description includes a brief description of the ontologies used to

[1] AmiGO – `http://amigo.geneontology.org/`

formalise the knowledge domain, as well as the objects they annotate.

2. **Types of queries:** This criterion specifies the types and natures of the queries the tool enables the execution of, specifically the operators allowed, the operands that can be used, and any partial restrictions on these.

3. **Support for initial term finding:** This criterion specifies whether and if so, how the tool supports the user when finding terms to begin query construction.

4. **Support for combining terms:** This criterion specifies any techniques used to facilitate the finding of valid combinations of terms.

5. **Display of results:** This criterion is used to describe how result sets are displayed, including the links that are provided to other data sources.

6. **User involvement in development:** This criterion is used to assess the extent to which domain experts (end users of the system) were involved in the design or evaluation of the system. Where users were involved, the nature and extent of the involvement has been specified.

7. **Technologies:** This criterion specifies which programming languages, software libraries and platforms the tool is based upon.

3.4.3 AmiGO

AmiGO is the official searching and browsing tool for the Gene Ontology database. As such its primary focus is on finding individual terms and gene products, and it is not intended to facilitate the construction of ontology-based queries. Regardless, it can be used as an aid to their construction, if complete gene annotation sets for each term in a query are retrieved and the query itself executed in a third-party application, such as a spreadsheet. In this analysis, AmiGO is considered in conjunction with a spreadsheet as a tool for facilitating ontology-based query construction. A screen shot of the AmiGO results view is shown in figure 3.1.

1. **Problem domains:** AmiGO is applied to the domain of gene function, using the Gene Ontology (approximately 25 000 terms). This includes all species' genes in the core GO database (approximately 2 000 000). It has also been applied to the Plant

Ontology [52], with approximately 1100 terms annotating approximately 15 000 gene products. For the purpose of this review, only the GO application will be considered.

2. **Types of queries:** AmiGO facilitates the finding of individual terms, individual gene products, and sets of gene products annotated with a given term and its descendants. Additionally, these sets of gene products can be narrowed down by annotations type and by species. Queries involving set operators are not enabled by the interface, but can be carried out, somewhat laboriously, using a spreadsheet.

3. **Support for initial term finding:** AmiGO provides a basic, form-based interface for finding individual terms, as well as a tree browser for navigating up and down the polyarchy. The results page of the search shows hits for the search keywords, and displays their definitions on demand.

4. **Support for combining terms:** AmiGO does not enable the execution of multiple-term queries.

5. **Display of results:** AmiGO provides a tabular results view, with paging (showing 50 result per page). The symbol, name and species are provided for each result, as well as the evidence code, source and reference for the annotation. Links are provided to a BLAST search with the gene product sequence for some gene products. Links are provided to view all the annotations for a single gene product, which can be used to construct new queries.

6. **User involvement in development:** There is no evidence in the literature or AmiGO documentation of user involvement of any sort in the development of AmiGO. However, there is a suggestions form on the AmiGO site, and the interface is occasionally discussed on the public Gene Ontology mailing list.

7. **Technologies:** AmiGO is web-based, written in the Perl programming language, and makes use of a MySQL database as a back end.

3.4.4 MartView and BioMart:

BioMart , is described as “a generic data warehousing solution for fast and flexible querying of large biological data sets” [58] or ”a simple, federated query system designed

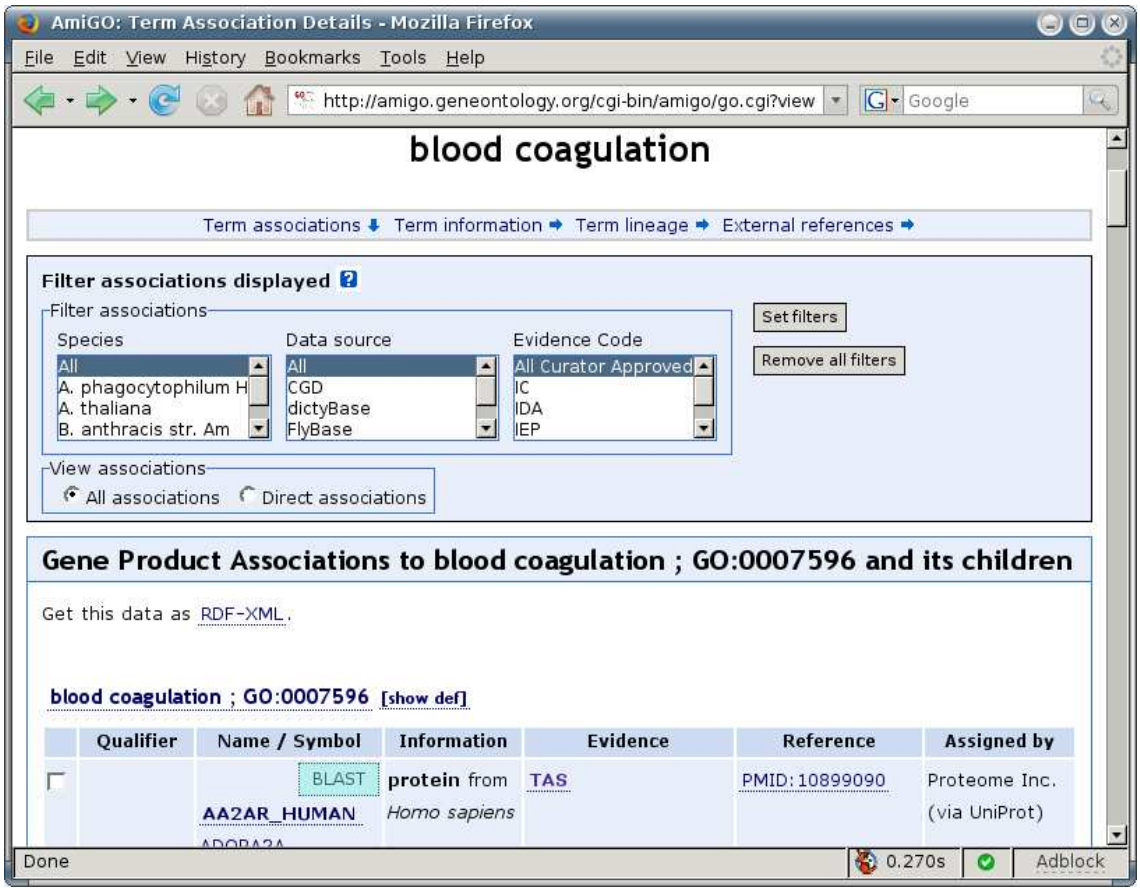

Figure 3.1: Screen shot of AmiGO, showing the results view. The query can be narrowed by species, evidence code and data source, but not by further ontology terms. Links to other databases and further detail on specific gene products returned are made available. (Image used under the terms of the Artistic License [3])

specifically for use with large datasets" [29]. It has been used by a wide range of genome and other biological databases, most notably Ensembl, an automated genome annotation database [49]. The query process involves three explicit steps: start, filter and output. During the start step, a database is chosen. During the filter step, results can be restricted to genes that have been mapped to a particular external id set. Among the external id sets which can be queried are Gene Ontology terms. Since queries can be narrowed by specific combinations of GO terms, the construction of ontology-based queries is possible using this interface, although that is not its primary purpose. For the purposes of this analysis, the "filter by GO terms" feature is focused on, in the context of the Ensembl implementation of BioMart/MartView. A screen shot of MartView, showing this feature, is in figure 3.2

1. **Problem domains:** Although BioMart has been applied to a wide range of domains for information integration, in terms of its use of ontologies it has only been applied to Gene Ontology annotations of the information it integrates. This includes all of the gene products in the Ensembl database, as well as several others.

2. **Types of queries:** BioMart enables the building of queries across multiple ontology terms. However, the MartView interface restricts these to one term per ontology (of the three ontologies within GO). The query can be narrowed by evidence code, but only one evidence code may be used per ontology term. In the Ensembl case, only one species may be chosen for querying. Interestingly, Ensembl allows users to query for genes orthologous to those annotated, in addition to the genes themselves.

3. **Support for initial term finding:** All terms in MartView are entered into a form-based interface. Only the GO accession number of the term is accepted. A button to a browsing facility is provided next to the term input box, which pops up a window containing the QuickGO [4], [19] interface, displaying the root term of the ontology. This interface allows the ontology to be browsed and keyword searched, although the keyword search interface is not immediately visible. However, once a term is found, the user must manually copy and paste the accession number back into the MartView interface.

4. **Support for combining terms:** Entering of multiple terms in MartView is carried out simultaneously in a single step, and no support is provided for finding valid

[4] QuickGO – `http://www.ebi.ac.uk/ego/`

combinations of terms.

5. **Display of results:** By default, only the gene and transcript ID numbers are displayed in the results table. More information can be displayed by manually configuring this view, however. MartView also provides functionality for exporting results to formats compatible with third-party software, such as spreadsheets.

6. **User involvement in development:** There is no evidence in the literature or documentation for MartView that users were involved in the development.

7. **Technologies:** MartView was created in the Perl programming language, is web-based and makes use of the biomart-perl library. BioMart is used as a back end system.

3.4.5 GViewer

GViewer [92], is a tool designed for and hosted by the rat genome database [103]. It was specifically designed to facilitate the construction of ontology-based queries, and displays the locations of the resulting genes on a representation of the rat chromosomes. A screen shot of the query interface for GViewer is shown in figure 3.3

1. **Problem domains:** GViewer is applied specifically to the domain of rat genomic information. The information annotated includes genes producing gene products and quantitative trait loci (QTLs), (regions of DNA that associated with a particular trait, but possibly larger than the actual genes known to be associated with the trait.) Several ontologies are queriable, including the Gene Ontology [6], Mammalian Phenotype Ontology [95], a Disease Ontology adapted from the Medical Subject Headings (MeSH) [67] and an in-house Pathway Ontology.

2. **Types of queries:** GViewer enables the execution of queries across up to three ontology terms from six different ontologies (since GO contains three), using “OR”, “AND” and “AND NOT” operators. Evidence codes cannot be used to narrow the query, and the data set queriable covers only one species.

3. **Facilitation of initial term finding:** The query construction interface is a simple form-based one, in which term names are entered. No support is provided to the user

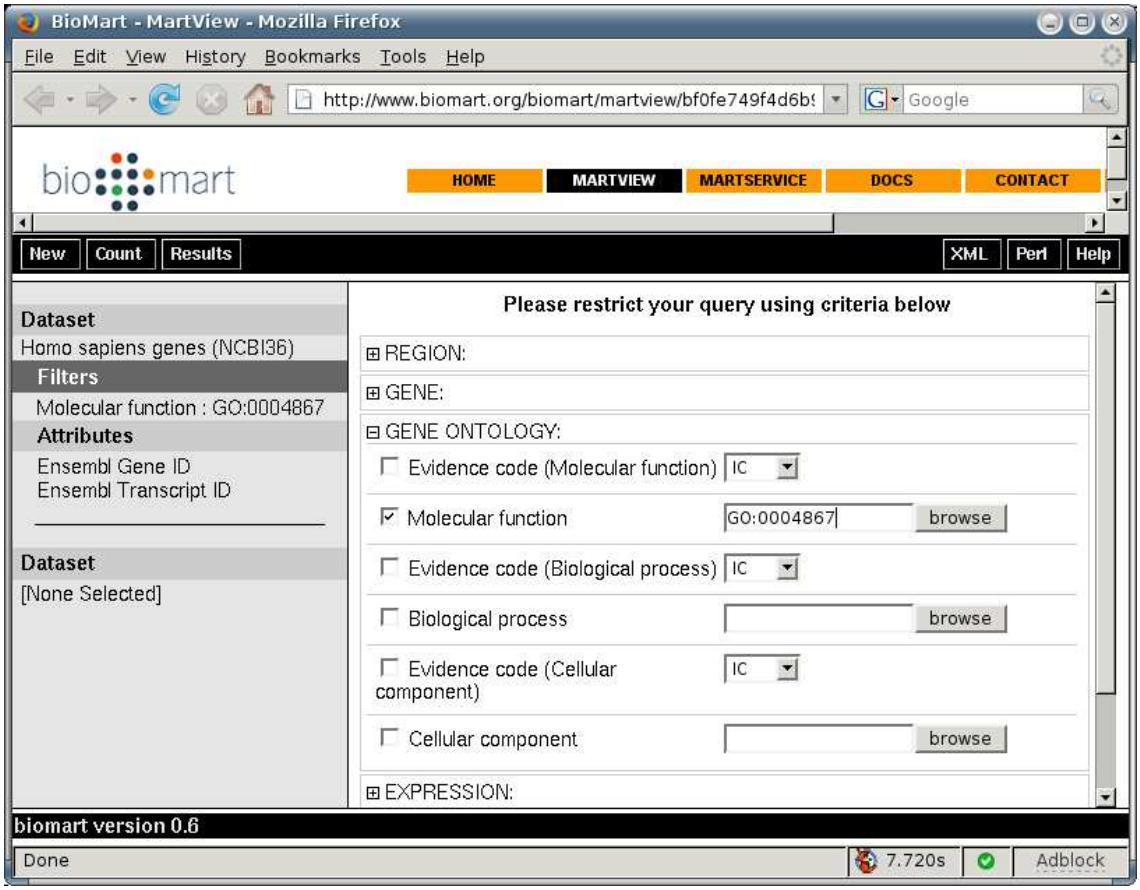

Figure 3.2: Screen shot of MartView, showing how a filter can be specified to create an ontology-based query. Only one term from each of the GO ontologies can be used, and only one evidence code per term. The "browse" button pops up the QuickGO browser, but a term, once found in that browser, has to be manually copied into the field. (Image used under the terms of the Lesser GNU Public License [6])

in finding the first term: It is up to them to guess (or use third party tools to discover) correct term names.

4. **Facilitation of term combination finding:** No support is provided for the finding of valid combinations of terms. When an invalid combination is entered and the query executed, a screen stating that no results were returned is displayed. This same screen is displayed if no results were returned due to a mis-spelled or non-existent term being entered.

5. **Display of results:** Results are displayed as a Scalable Vector Graphics (SVG) image, requiring the download of a third-party plugin for users of Microsoft Internet Explorer. For users of Mozilla Firefox and other browsers which provide built-in SVG support, the results page simply refuses to display the SVG, due to a JavaScript check which requires the presence of the plugin. In addition to the SVG graphic, a link is provided to download the results as a proprietary-format Microsoft Excel spreadsheet file. The results are not viewable as a table, and the only links to further information are more detailed views of the chromosomes. Although these chromosome displays claimed to contain links for each gene to further information within the RGD, they were not working at the time of testing.

6. **User involvement in development:** There is no evidence to suggest that users were involved, or that any form of usability engineering was carried out in the development of GViewer. A general mailing list does exist for the whole database, but the only mention of GViewer is the announcement that it was released. There is no evidence of further community interaction or further development of the tool.

7. **Technologies used:** GViewer is web-based, written in the Java programming language, and makes use of the Java Server Pages technology, as well as Scalable Vector Graphics (SVG) to display results. A custom infrastructure written in the Oracle PL/SQL language and running on an Oracle 9i database serve as the back end.

3.4.6 Drug Ontology Project for Elsevier (DOPE)

DOPE was created as a collaboration between Aduna Software, HCI and Semantic Web researchers at the Vrije Universiteit Amsterdam, a medical informaticist at Erasmus

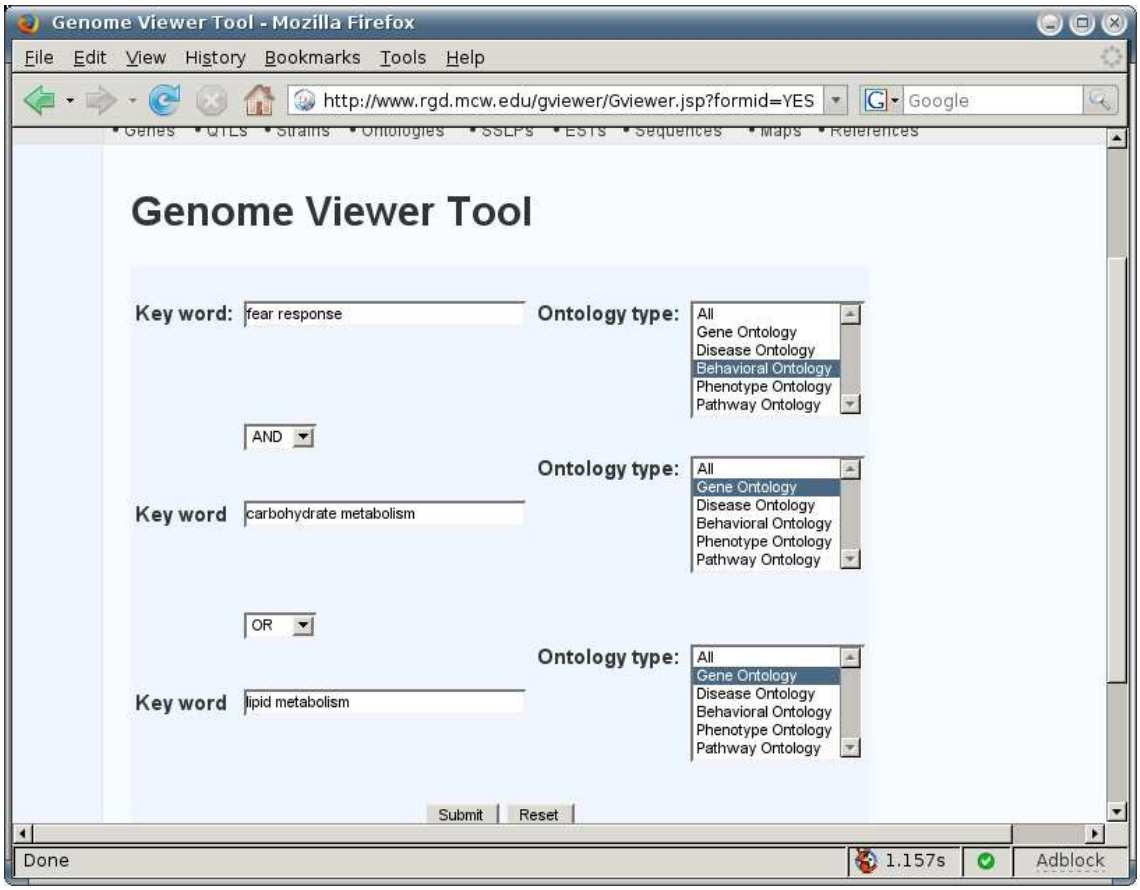

Figure 3.3: Screen shot of GViewer, showing the query construction interface. Term keywords are entered into the fields, and the ontology can be specified. No option is given to choose among terms matching the keywords, however, and no support is given for finding keywords that will actually match terms. Up to three terms can be combined using the operators "AND", "OR" and "NOT", but no support is given for finding valid combinations of terms. (Image used with permission.)

University and several members of the academic publisher, Elsevier. The purpose of the project was to develop a thesaurus based browser for accessing heterogeneous and distributed data, based on the RDF (Resource Description Framework) data model [17]. A notable feature of DOPE is the emphasis on enabling exploration of result sets by the user, to assist them in restructuring their query. This was enabled by the use of a custom visualisation known as ClusterMap. A screen shot of the DOPE integrated results/query view is shown in figure 3.4

1. **Problem domains:** DOPE was applied to a document repository consisting of the full content of Elsevier's Science Direct database, in addition to the previous ten year's worth of the Medline database (10 million entries in all). These were indexed against EMTREE, a life sciences controlled vocabulary developed by Elsevier consisting of around 50 000 terms.

2. **Types of queries:** In DOPE, a term is initially chosen to define the result set space, then further querying is done within the documents defined by that term. Within that space, AND queries consisting of an arbitrary number of terms can be constructed.

3. **Facilitation of initial term finding:** DOPE provides a form for the user to search for the first term. A list of formal terms that can be related to this string is suggested. (For example, the list of suggestions for the query "aspirin" contains the term "acetylsalicylic acid", the chemical name corresponding to the brand name.) Functionality is also provided allowing the user to navigate through the thesaurus in search of a term.

4. **Facilitation of term combination finding:** Once a term has been selected, a list of the top 500 co-occurring terms is displayed, in a tree form according to the inheritance structure of the thesaurus. Multiple terms can be selected, and all documents co-occurring among any combination of the selected terms (and, implicitly, the initial term) are shown. Although this facilitates the finding of combinations between the initial term and those displayed in the list, the only help provided for finding the third term is the display of combinations if the user serendipitously selects terms which are further combinable.

5. **Display of results:** Results are displayed using the highly visual ClusterMap tool. This shows the number of documents associated with each term selected, as well

as combinations of terms where terms are co-occurring, in a view similar to a Venn diagram. A problem identified by user testing of this interface was that the complexity increases rapidly after more than three terms are applied, making the visualisation difficult to deal with [97].

6. **User involvement in development:** The creators of the EMTREE ontology at Elsevier were involved in the development of DOPE. However, it does not appear that actual end users were involved in the development. Regardless, a usability evaluation was carried out with ten users after the system was developed, and the results included in one of the publications [97]. Users indicated that they felt that the tool helped greatly in exploring an unfamiliar information space, but that it was inadequate for concrete and specific search tasks.

7. **Technologies used:** DOPE is written in the Java programming language and is designed to run on the desktop with access to the Internet. The user interface makes use of the ClusterMap library developed by Aduna software [7] The back end uses the Sesame RDF storage and querying architecture [18], also developed by Aduna.

3.4.7 Flamenco

Flamenco (FLexible Access to MEtadata in NOvel COmbinations) is an interface designed specifically for image searching using hierarchical metadata describing the images [30]. The designers made extensive use of pre-existing studies of image retrieval tools, as well as HCI methodologies. Additionally, the HCI technique of query previewing was made use of [83]. A screen shot of Flamenco's integrated results/query view is shown in figure 3.5.

1. **Problem domains:** Flamenco has been applied to three image databases, containing architectural, fine arts and Nobel Prize winners image sets. The two larger databases contained 35000 and 36000 images respectively, and were indexed using thesauri containing around 4000 terms. The thesauri were automatically generated and were divided into around 10 orthogonal categories, which the authors chose to call "facets".

[7] Aduna ClusterMap – `http://www.aduna-software.org/projects/display/CLUSTERMAP/Home`

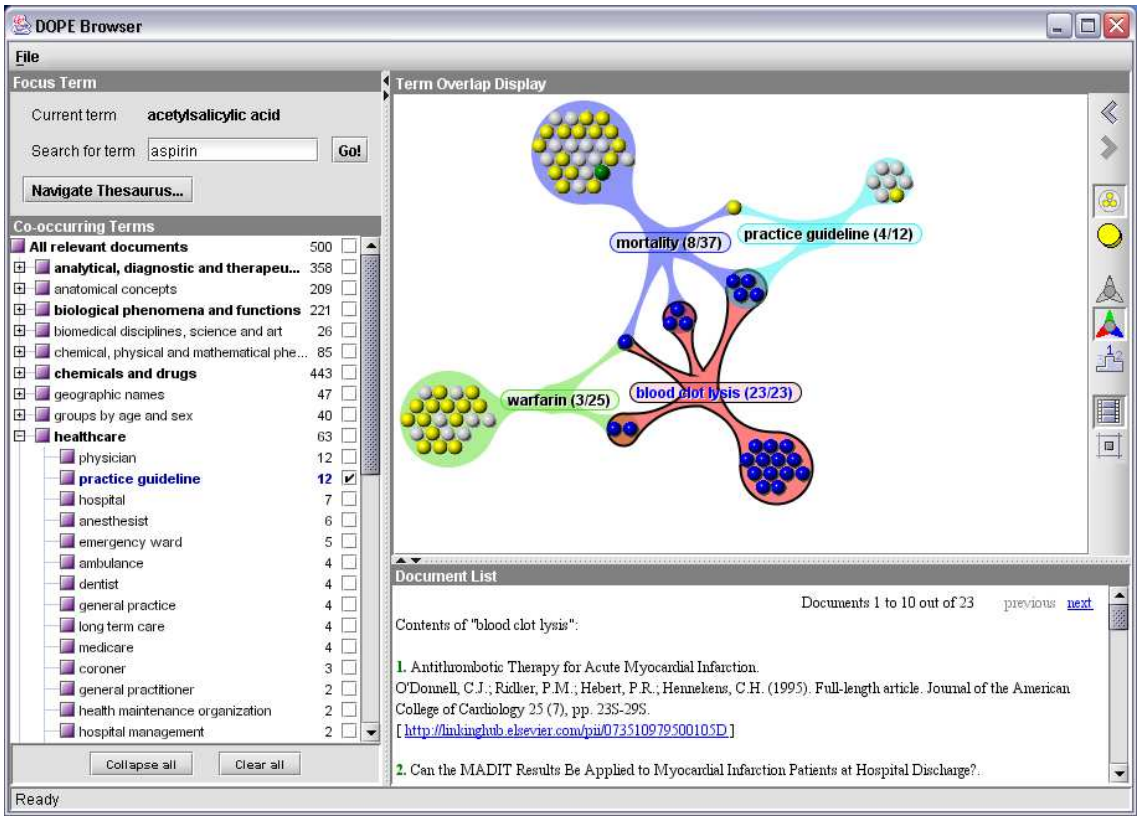

Figure 3.4: Screen shot of the DOPE browser, showing the results/query construction view. In this case, the term "acetylsalicylic acid" has been selected as the first term, or "facet". Terms related to this term by co-annotation are provided as a list on the left, and relations among the terms selected are displayed on the right. When more than three terms are selected, the visualisation can become overly complex. (Image used with permission.)

2. **Types of queries:** Flamenco facilitates the construction of AND queries across any number of terms which produce a result. However, the controlled vocabularies used do not allow multiple inheritance, so only one term may be used per facet. No additional factors are available to further constrain the query, and only the AND operator is allowed. Raw keyword searching through the text annotating the image can be combined with a term-based query.

3. **Facilitation of initial term finding:** Flamenco shows the terms at the top level of each facet, as well providing a keyword-based search form. As additional aids, the number of images associated with each term is displayed next to the term, and the next level of terms down from the term can be previewed as a tool tip by hovering the mouse pointer over them.

4. **Facilitation of term combination finding:** Once the first term has been selected, results are displayed, and a revised view of the facets is generated, displaying only those terms which are combinable with the existing query, in compliance with the concept of query previews [83]. This eliminates zero-result queries, and provides substantial support to the user when trying to find combinable terms.

5. **Display of results:** Results are displayed as page-able thumbnails, grouped by a chosen facet. When grouped, the top level of terms within the chosen facet are used to visually cluster the results. Each thumbnail provides a link to the complete record for the image, including all of the terms used to annotate it and brief description text. This interface can also be used to construct a new query, using a combination of the terms used for the image being viewed.

6. **User involvement in development:** The initial step in designing Flamenco was an ethnographic study of how architects search for and use images. This was followed by a cycle of low-fidelity prototyping, informal usability testing and redesign. Finally, two cycles of development and usability testing were carried out. Finally, a usability study was conducted to compare Flamenco with a "baseline" image search interface, which incorporated features from popular image search tools such as Google Image Search. Flamenco came out favourably in this test [111].

7. **Technologies used:** Flamenco is web based and written primarily in the Python programming language. The WebWare library is used within Python, and MySQL is

used as the back end. Full-text search is optional, requiring the Java-based Lucene library.

3.4.8 Trends in the comparison

Table 3.1 summarises the individual evaluations. Certain trends emerged:

Most tools were web-based

With the exception of DOPE, all the tools were web-based. All the bioinformatics specific tools were web-based. A possible reason for this is that most bioinformatics information is available via the web [37] , so that a web browser enables users to rapidly access information related to the results they obtain from the tool.

The bioinformatics tools reviewed showed little to no evidence of the use of HCI methodologies

A notable difference between the bioinformatics-specific tools and DOPE and Flamenco lies in the use of HCI methodologies in their development. In the case of Flamenco, a complete usability engineering process was worked through, right from ethnographic studies through to iterative design based on usability testing [111]. By contrast, there is little mention of users being taken into consideration in the engineering of the bioinformatics tools reviewed here, or, indeed, in the broader bioinformatics literature. Furthermore, there is little to no mention of the use of pre-existing concepts from the literature in the design of the bioinformatics interfaces, whereas both Flamenco and DOPE make extensive use of pre-existing HCI, information visualisation and information retrieval theory as a conceptual framework for their initial designs. Since HCI methods have been shown to be extremely important to the usability of an interface [73], it is likely that their greater application within bioinformatics would result in more usable software. The application of existing theory from related fields would most likely be beneficial, too.

The bioinformatics tools tended not to support complex query construction or results exploration

Where the bioinformatics tools provided the ability to construct complex queries, they did not provide support to the user for finding valid combinations of terms. This

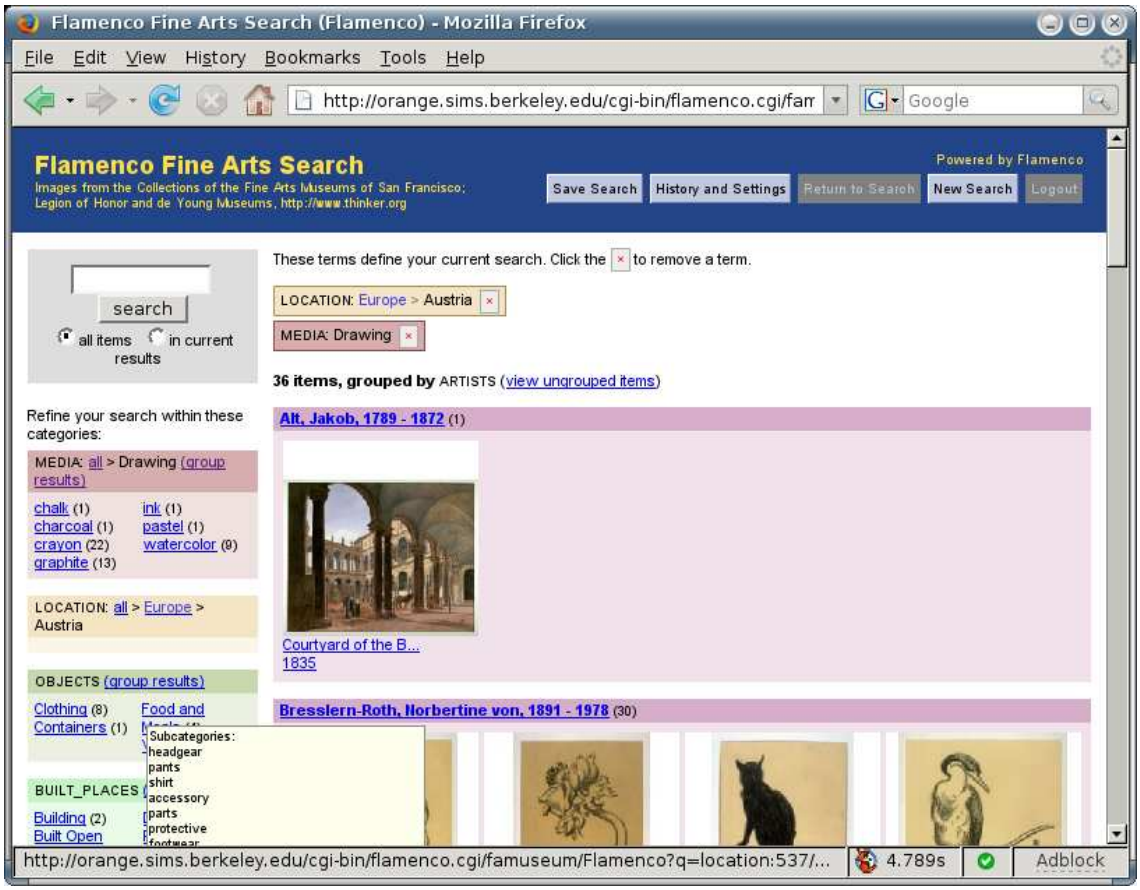

Figure 3.5: Screen shot of Flamenco, showing the Fine Arts application. In this view, drawings in the collection which were produced in Austria have been selected, and grouped by artist. The mouse cursor hovers over the term "Clothing" under the "objects" facet, showing the subcategories of that term. Each image can be clicked on to bring up details and all its annotations, which can be used to construct a new query. (Image used under the terms of the BSD license [9].)

can lead to the finding of many zero-hit queries or queries producing large numbers of results for users to have to browse [83]. Furthermore, in both MartView and GViewer, no distinction is made between queries that return zero results due to an invalid term name and those which return zero results due to a combination of terms which are simply unrelated. By contrast, both Flamenco and DOPE provide support for finding combinations of terms, although the visual techniques used in DOPE have been shown not to scale well. A problem with the Flamenco interface, however, was that it used a relatively small dataset, and used controlled vocabularies that only allowed single inheritance. The complete GO core dataset indexed by AmiGO contains two orders of magnitude more entries and an ontology five times larger, which includes multiple inheritance. It is uncertain whether the Flamenco approach would work on such a data set. Regardless, both Flamenco and DOPE provide substantially better cognitive support to users when constructing complex ontology-based queries; it is likely that this resulted from the application of pre-existing theory and the use of HCI methodologies involving users.

Form-based interfaces did not use real-time query expansion

Real-time query expansion (RTQE) is an interface mechanism whereby candidate expansion terms are presented to the user as they enter a search query [107]. This has been shown to improve the quality of initial queries, and does not obstruct the normal use of a form based query interface [107]. Within the context of bio-ontologies, such an interface for finding ontology terms has been implemented, in the form of the Ontology Lookup Service (OLS) [24]. Regardless, none of the interfaces reviewed, within or without the biological domain, implemented this functionality. It is likely that this would assist greatly in the finding of initial terms in an ontology-based query interface.

3.4.9 Conclusions

Although the back end technology for executing complex ontology-based queries within the biological domain does exist, there is yet to be an interface which provides substantial cognitive support to users when constructing such queries. Such interfaces have been created outside of the biological domain, but these have had issues of scalability, and their direct application to the domain may not be feasible. A new interface for constructing ontology-based queries, with a design based on existing theory and refined by user involve-

ment, applied to the biological domain, would be a useful and novel addition to the suite of bioinformatics software available to biologists.

Criterion	AmiGO	MartView	GViewer	DOPE	Flamenco
1.Problem domains:	Gene Ontology (25K terms) to multi-species gene DB (2M entries)	Gene Ontology (25K terms), individual species gene DBs (30K entries each)	Gene Ontology (25K terms), several others (5K each); rat gene and QTL data (9K entries)	EMTREE ontology (50K terms), custom literature DB (10M entries)	image metadata thesauri (4K terms each); image databases (35K entries each)
2.Types of queries:	single term only; narrowing by evidence code, species.	multiple terms, only one term, evidence code per ontology. Only one species per query	AND, OR, NOT over several ontologies; no species / evidence code narrowing	AND queries of any number of terms; no narrowing criteria	AND queries across any number of terms; only one term per orthogonal ontology; supplementation with keyword search
3.Initial term finding:	forms, tree navigation	QuickGO browse, search, but no tie-in with MartView interface	no support	Form with intelligent term suggestion; tree navigation	keyword search, dynamic tree navigation
4.Combination finding:	no support	no support	no support	valid combinations with first term shown, but limited support for 3 or more	all valid combinations with current query displayed, also size of result set (query previewing)
5.Display of results:	paged table, links to detail on each query, links to external information.	simplistic but configurable to be richer; spreadsheet export	highly visual SVG; no table but proprietary spreadsheet export	interactive, visual cluster map; problems with scalability	page-able table, links to detail on each entry, ability to construct new queries from annotations of entry
6.User involvement:	minimal, though possibly via mailing list	no evidence of any	no evidence of any	usability evaluation post-development	multiple cycles of testing, re-development, evaluated against a baseline
7.Technologies:	web-based: Perl, MySQL	web-based: Perl, BioMart	web-based: Java JSP and Oracle 9i PL/SQL	Desktop-based: Java / Swing, ClusterMap, Sesame RDF store	Web-based: Python WebWare, MySQL, Java / Lucene optional

Table 3.1: A comparison of five tools for facilitating the construction of ontology based queries, according to the criteria specified earlier in the chapter.

Chapter 4

Design

A series of interviews were conducted with a domain expert in the field of molecular immunology (henceforth the DE). The domain expert was chosen because she was a molecular biologist with extensive wet lab research experience, but also prior experience in using boolean ontology queries to aid her research. Specifically, she used the Mouse Genome Informatics web site to obtain sets of genes for individual terms, then used a spreadsheet, Microsoft Excel, to perform set operations on these results. This prior experience made her ideally suited to suggest biological cases in which ontology based queries would prove useful, and to suggest ways in which a user interface could support the query construction process.

The interviews had three major goals: firstly, to obtain biological scenarios to drive design and development, secondly, to cooperatively design the user interface of OntoDas, and thirdly, to cooperatively refine OntoDas as it was developed.

In the context of software engineering, a scenario, also known as a use case instance, is defined to be *a description of a specific sequence of actions, in which specific persons replace actors, and only one path is taken through the use case's possible basic and alternate flows.* [108]. Less formally, it is a *particular story of using a system* [62]. A collection of scenarios is used to define a use case, which is a *complete task of a system that provides a measurable result of value for an actor* [108], [62]. The first phase of interviews focused on eliciting biological scenarios to compose into the overall use case for OntoDas. In order to do this, and to gain insight into the work process of biologists, the process of constructing queries was simulated using existing online tools and a script library using the GO MySQL database. This helped in gathering the scenarios, in agreeing (with the DE) on the structure

of the general use case for OntoDas and in setting the scene for cooperative design of the user interface of OntoDas.

The design of system screens was carried out cooperatively with the domain expert, in a system similar to cooperative prototyping [16]. This was carried out in the form of pen mock-ups of screens, shown later in this chapter. These mock-ups, their accompanying notes taken by the system designer, and the use case developed from them were used to develop the user interface for OntoDas. The requirements and design of OntoDas were further refined based on the outcomes of evaluations by the domain expert during the development cycle.

This chapter begins by describing the scenarios that were elicited. It then describes how these scenarios were used to derive an overarching use case, and the steps of which it is composed. The chapter then shows the design of OntoDas, with reference to the paper mock-ups developed in cooperation with the DE. Technical aspects of the implementation are left for the next chapter.

4.1 Biological scenarios

As a result of the work with the DE, three scenarios were created to be used as use case instances. One of these, discussed in 4.1.3, was explored in greater detail, using a working copy of the system itself. It was felt, by agreement between the DE and the designer, that these were representative of what users would want when using the system. This section provides some of the biological background to the scenarios and mentions both the queries produced and the steps taken to get to them.

4.1.1 Genes with potentially fatal knockout effects in knockout mice

Mice are often used to model both hereditary and infectious diseases. In the context of infectious diseases, genes, especially those known or suspected to be involved in the immune system, are sometimes silenced to discover their effect on the pathology of a particular disease. (Mice that have had a gene silenced in this way are referred to as "knockout mice".) However, as many genes are multifunctional, the gene silenced may prove to be essential during embryonic development, and its silencing may result in mice that die *in utero* or are born with anatomical abnormalities. Predicting genes with this

potential, so that other genes may be chosen as the focus of research, would be extremely helpful to immunologists.

The query thus constructed is:

Find genes that **play a role in** *embryonic development* and that **play a role in** *immune response*.

4.1.2 Finding blood coagulation related protease inhibitors

Background

Blood coagulation is a vital part of haemostasis (the arrest of bleeding from an injured blood vessel) [9]. It is a complex process, involving a series of enzymes which each activate the next enzyme in a cascade, terminating with the activation and cross-linking of the protein fibrin. Disorders in blood coagulation can result in the pathological formation of thrombi (blood clots) in veins, arteries or the chambers of the heart, with potentially fatal effects [9]. A researcher interested in thrombotic disorders may attain insight into the disorders by finding proteins known to be involved in the process, and by investigating the other roles these proteins play.

Scenario

1. The DE views the ontology terms used to annotate a known coagulation factor, Coagulation factor VII (*FA7*). Three terms are shown: *coagulation factor VIIa activity*, *blood coagulation* and *extracellular region*. The DE chooses all three terms to begin a query with.

2. The DE views the results of this query. Only one gene product (*FA7*) is found. The DE realises that the term *coagulation factor VIIa* activity is too specific, and substitutes it with its parent term, *serine-type endopeptidase activity*.

3. The DE views the results of this query, and is interested in seeing what other terms can be combined with the query. The term *apoptosis* is chosen to add to the query, as the DE had been unaware of a connection between this process and blood coagulation.

4. The DE views the results for this query and sees that only around three gene products of the nine results returned are non-redundant. (The remainder are primarily

orthologues of protein C, one of the results, in different organisms.) At this stage, the DE wants to obtain more information on the results from external data sources.

4.1.3 A relationship between the integrin-mediated signalling pathway and phagocyte maturation

Background

The disease listeriosis is caused by the ubiquitous food-borne bacterium *Listeria monocytogenes* [87], and is of particular risk to immunocompromised individuals, such as those with HIV/AIDS or undergoing chemotherapy [56]. The bacterium invades host cells via phagocytosis (being enclosed in a membrane-bound vesicle called a phagosome). In phagocytes, cells of the immune system, the phagosome usually fuses with a lysosome (another vesicle containing digestive enzymes) after phagocytosis. This causes the pathogen within the phagosome to be broken down by the enzymes. Listeria, however, prevents the maturation of phagosomes to a state where they can fuse with lysosomes, thereby preventing its being killed by the host phagocyte [25].

This has been shown to occur by the prevention of the activation of a protein encoded by *Rab5A* [84]. This protein becomes active by binding to guanidine triphosphate (GTP). In carrying out its activity, it removes a phosphate from the GTP, converting it to guanidine diphosphate (GDP). Guanidine exchange factors then replace the GDP with a GTP to reactivate the protein. In Listeria, it was found that the *Rab5A* protein remained in its GDP-bound form, and was thus not reactivated [84]. The mechanism by which this might happen is of some interest to the DE.

Scenario

1. The DE found the terms *phagocytic vesicle* and *GTPase activator activity.* She started a query with *phagocytic vesicle.* Searching through the combinable terms (of over 1000), however, she could not find the other term.

2. The DE decided to rather begin the query with *GTPase activator activity.* Searching through the combinable terms for "phagocytosis", she found the terms *phagocytosis* and *phagocytosis, engulfment.*

3. The DE decided to add *phagocytosis to the query.* This query produced 4 genes,

and had a much smaller set of combinable terms. From these terms, *oxidoreductase activity* caught her eye, as it is involved in the production of toxic molecules.

4. The DE added *oxidoreductase activity* to the query, producing only 1 gene. This gene was *DOCK1_human*, human dedicator of cytokinesis protein 1.

5. The DE viewed the annotations to *DOCK1_human*. In this list was the term *integrin-mediated signalling pathway*. This interested the DE, as she had not known of a link between this pathway and phagocytosis. She chose to research this gene further in the literature.

4.2 The use case: construct query

From the scenarios gathered from the DE, the general flow of query construction was extracted. This is illustrated in figure 4.1. This flow was then formalised into a use case, which was given the name "construct query". The steps are as follows: There are two alternate first steps which can be taken in the use case. The first is to construct the query *de novo*, and build it up term by term. The second is to examine a known gene product of interest, and create a query from some or all of the terms used to annotate the gene. Once the query has been created (ie. once it contains at least one term), the user would want to choose from removing a term, substituting a term with a related term, adding a new term, and viewing the final result set. These steps are outlined in more detail, illustrated and summarised in this section.

4.2.1 Construct query from gene product annotations

Gene products in the GO database tend to be annotated with several ontology terms each. Combinations of these terms can be used as the foundation of a query. An example of this is shown in section 4.1.2. To support this functionality, OntoDas must enable users to select some or all of the terms used to annotate a gene product, and execute a query using the terms chosen.

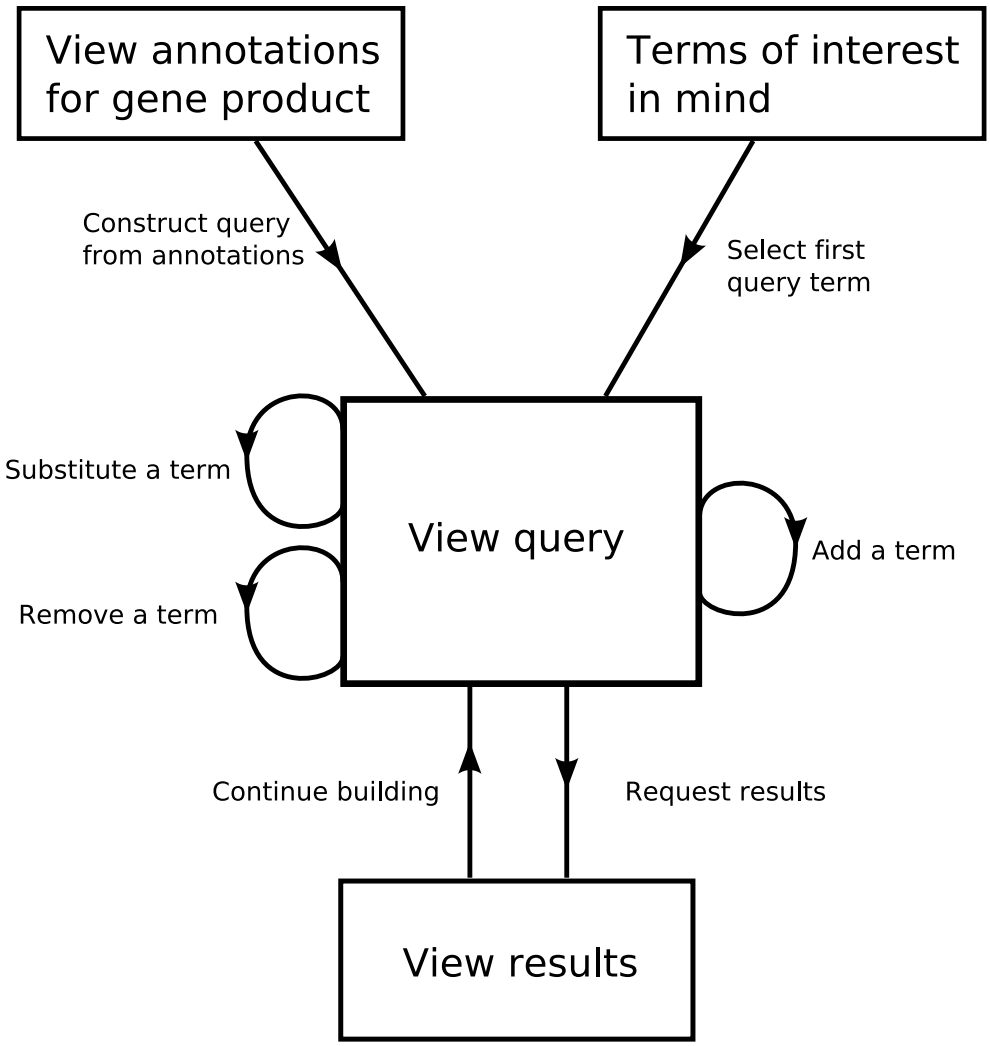

Figure 4.1: Query construction from the user's point of view. This is represented as a state diagram (boxes represent states and arrows represent transitions from one state to another). The design that was suggested by this system is elaborated on in table 4.1 and in figure 4.2.

Step	Description
4.2.1 Construct query from annotations	View terms used to annotate a gene product. Select one or more to form into a query.
4.2.2 Construct query *de novo*	Select one term to be the first in query construction.
4.2.3 Add term	View terms which are combinable with the current query. Select a term to add.
4.2.4 Substitute/remove term	View details of a term in the current query. Substitute the term with another, related term. Remove the term from the query.
4.2.5 View results	View the results of the query. View and be able to follow links to entries on the results in other databases. Export results to a spreadsheet.

Table 4.1: A summary of the steps making up the use case, showing brief details on each step. Complete details are given in the text. The numbers refer to the section in which the step is described.

4.2.2 Construct query *de novo*

To enable a user to build a query step by step, it is necessary to provide them with the ability to find the first term to build their query from.

4.2.3 Add term

Once a query has begun to be constructed, it is necessary to refine the query by adding terms as constraints. To allow this, users must be provided with the ability to find terms to add to the query, and add them.

4.2.4 Substitute or remove term

During the process of refining a query, users often wish to remove a term, or replace it with another term. Users need to be able to select terms within the query they have built, and find replacement terms, or remove terms.

4.2.5 View results

When the user is finally satisfied with the query, and in order to determine whether they are satisfied, the user must be able to view the results of the query. Users also indicated that they would like to be able to export these results to a spreadsheet, and be provided with links to other databases containing more information on specific gene products.

4.3 Natural language query representation

It was decided to provide users with a natural language representation of the queries they were constructing, in the hope that this would easier to understand than a more mathematical representation. In creating the rules for this representation, phrases were used based upon the formal relations laid out by Smith et al [93]. The phrase *"participate in"* is used to refer to terms from the biological process ontology, to express the relation *"has_participant"*. The phrase *"are located in"* refers to terms from the cellular component ontology, in order to express the relation *"has_location"*. For terms from the molecular function ontology, since no relation has been agreed on, the phrase *"have"* is used, expressing a simple, non-specific property relation. An example of a query represented in this way is:

Find genes that **participate in** *blood coagulation* and that **are located in** the *extracellular region* and **have** *serine-type endopeptidase activity*.

4.4 Screens and views

The use case described above was used, in collaboration with the DE, to design the user interface for OntoDas. The main views designed and the panels making them up designed, as well as the transitions between them, are illustrated in figure 4.2. This section outlines the initial design, showing the paper prototypes used, and detailing additional information that was elicited during the process about the design.

4.4.1 Dasty2 ontology terms view extension

It was decided that Dasty2, described in section 2.7.2, would be used to view details of individual gene products. This involved adding a panel to Dasty2 which would display the ontology annotations of a gene product and allow queries to be constructed, to fulfill the "construct query from annotations" step in the use case.

The paper prototype is illustrated and explained in figure 4.3. The ontology terms are listed, with check boxes next to them. The number of gene products annotated by each ontology term is displayed next to the term. As the query is built by selecting terms, a natural language representation is updated below. It was desired that the ability to filter the query by species and evidence code (as is possible in AmiGO – see section 3.4.3) be included.

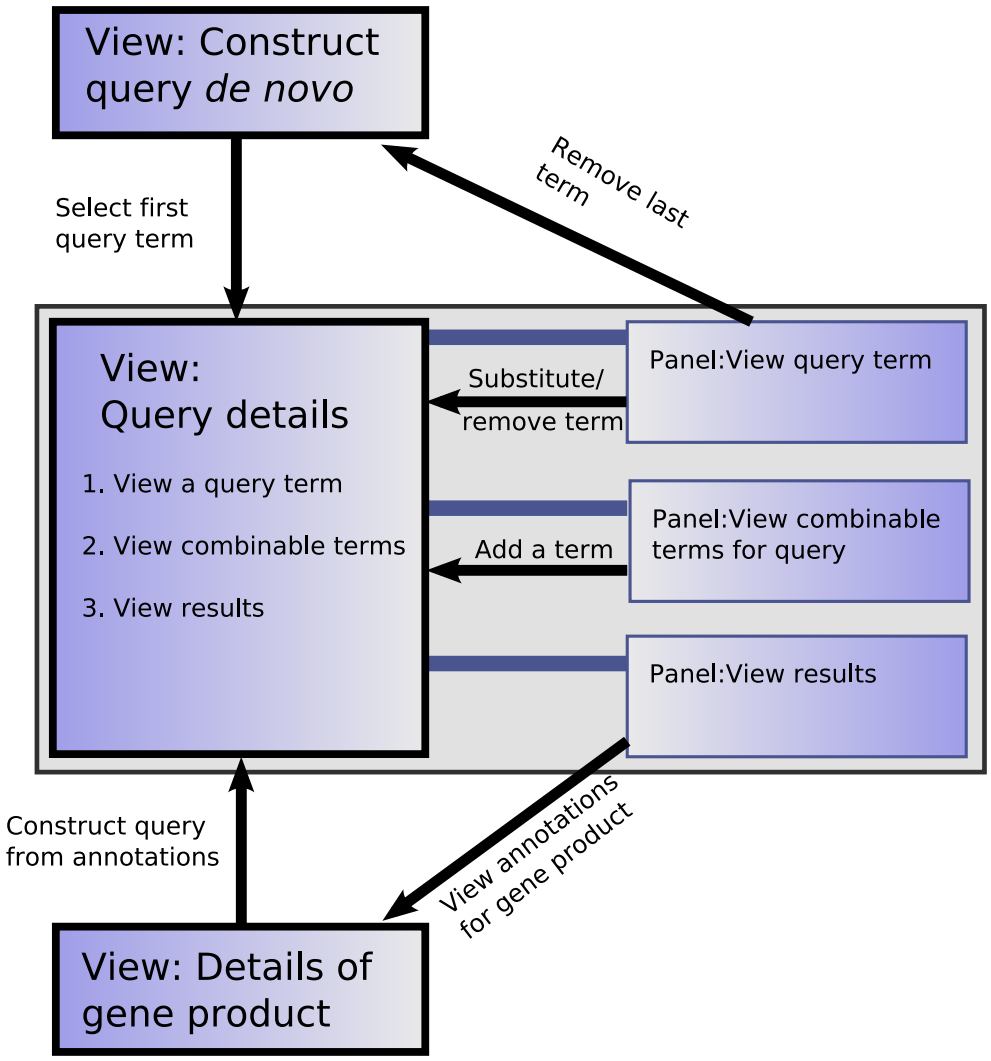

Figure 4.2: Views designed for query construction. Arrows indicate changes in the state of the information being displayed. The "query details" (aka "main query") view is shown with the panels that make it up. Actions from the "query term details" and "combinable terms" panels produce a change in the state of the query (by adding, removing or substituting a term), returning the user to the query details view, with the new query. The two starting points for constructing queries are shown.

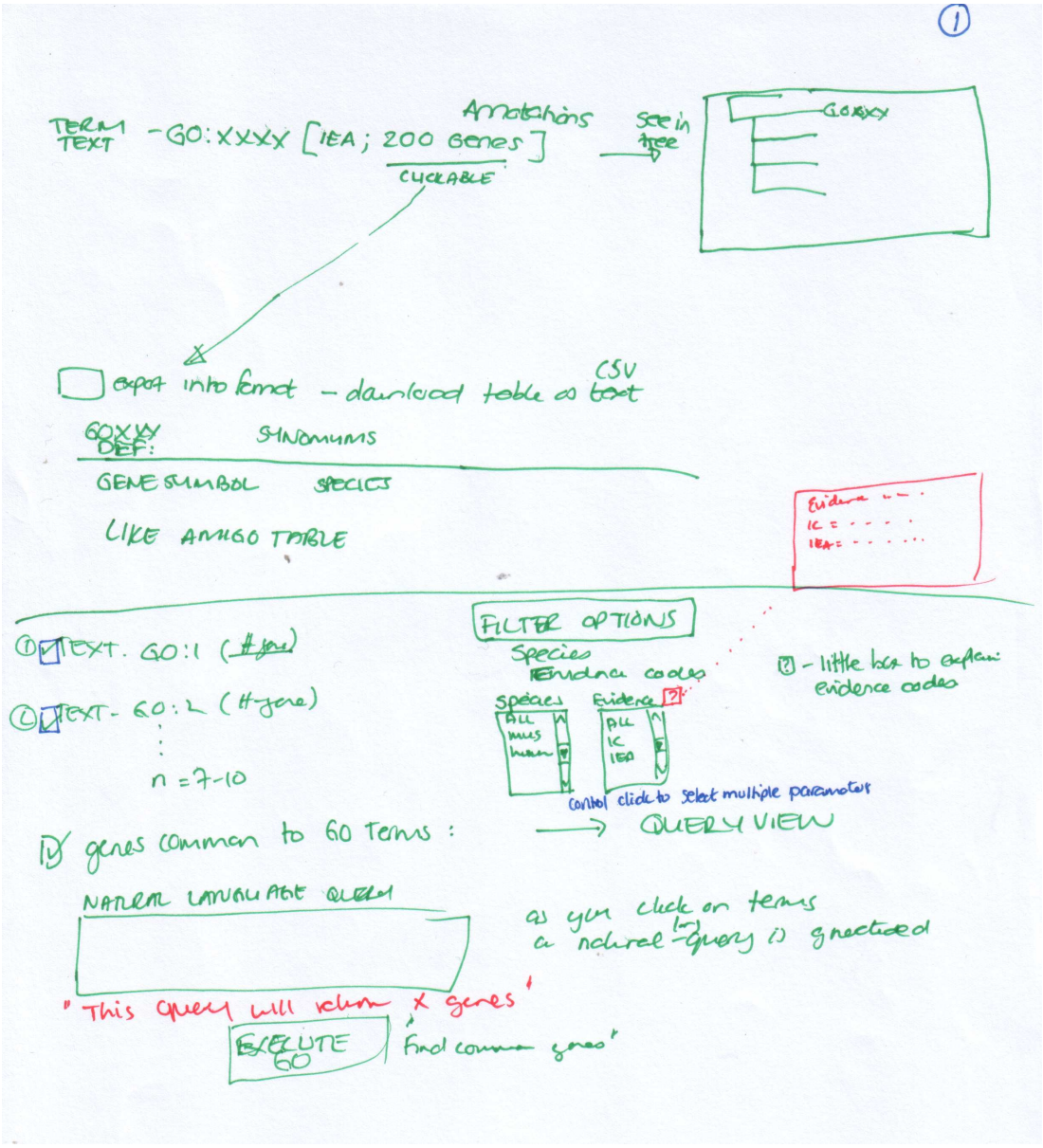

Figure 4.3: Paper prototype of the Dasty2 extension.

4.4.2 The main OntoDas view

The main OntoDas view, shown in figure 4.2, was designed to provide the *"view query"* central state in query construction shown in figure 4.1. An overview of the query is provided, consisting of a natural language representation of the query, and a summary of the results. Three additional panels, each providing specific functionality, and being collapsible, are provided as part of this view. These enable the user to view, replace or remove a term already in the query, add another term to the query, and view the results respectively. The relationships among these panels and the main view are shown in figure 4.2.

The design sketch for the main view, with all panels collapsed, is shown in figure 4.4. The query summary, in natural language, is shown at the top. Below this are panels for each of the query terms, providing the *"substitute or remove term"* functionality, followed by a panel showing combinable terms, providing the *"add term"* functionality. Finally, the results are shown, providing the *"view results"* functionality. More details are provided in figures 4.5, 4.6 and 4.7.

4.4.3 Substitute/remove term panel

This panel implements the *"substitute or remove term"* functionality. One panel is displayed for each term used in the query. The requirements for this panel were that it displayed details for the term in question, provided a link to remove the term and provided query previews for all the terms related to the term in question which could be substituted for it in the query to produce a new, valid, non-zero result set query. Specifically, it was required that parent terms (terms which the one in question is related to by the *is_a* or *part_of* relation), neighbouring terms (terms that share a common parent with the term in question), child terms (terms which the term in question is a parent of) and lexically similar terms (terms whose names are textually similar to that of the term in question) be displayed, with links to the new queries to be produced. Additionally, the size of the new queries would need to be displayed next to each term.

Details of the design are illustrated in figure 4.5. The panel was designed to contain three sub-panels: The first would display details on the term, including its definition and synonyms, and a link to view genes annotated to this term only. The second would display related terms which could be substituted for the term in question to produce valid queries, specifically parents, neighbours (terms with a common parent), children, and lexically sim-

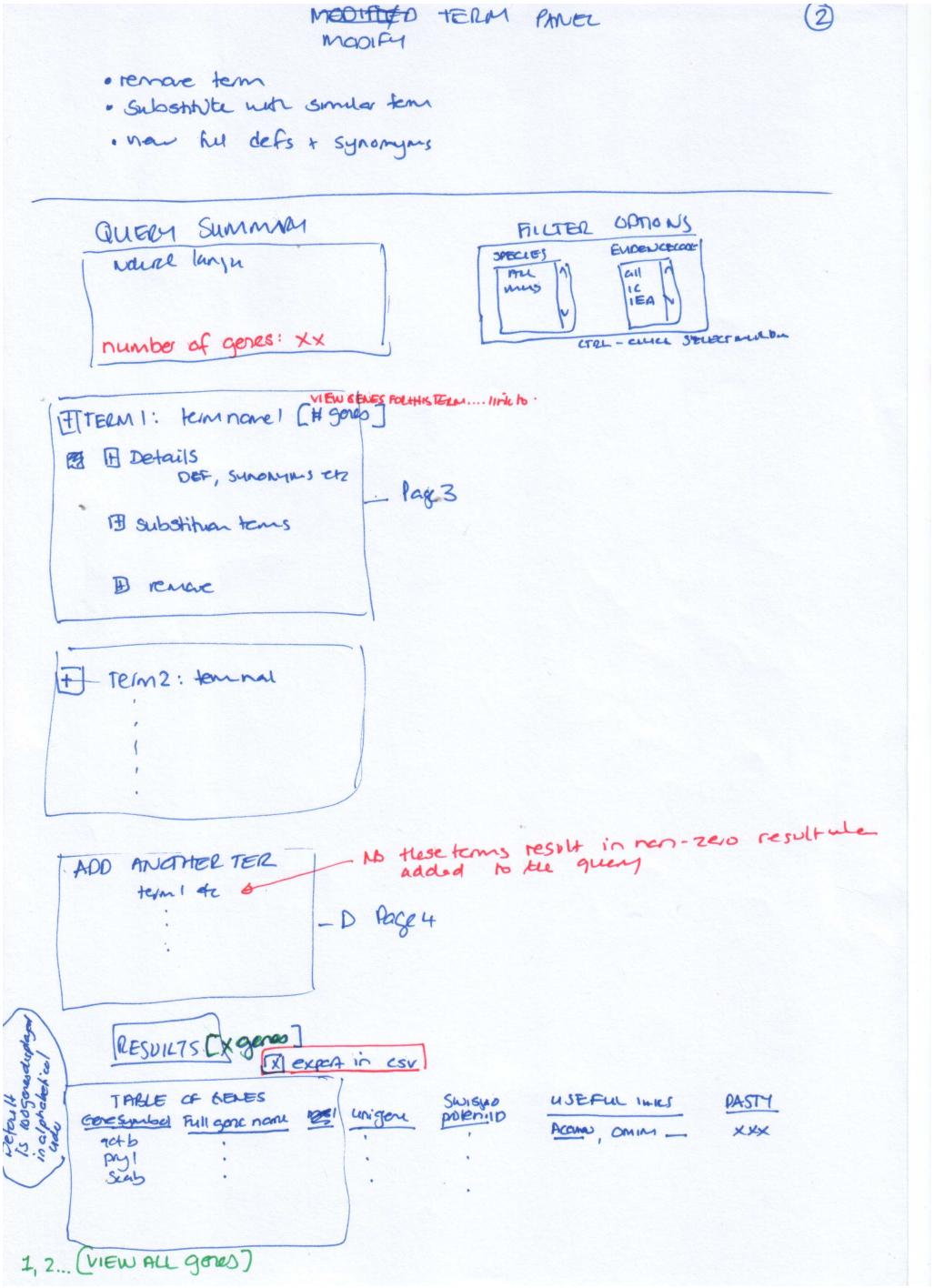

Figure 4.4: Paper prototype of the main OntoDas view. This began as a design for the substitute/remove term panel, but became a design for the view as a whole. (The detailed design for the substitute/remove panel is shown in figure 4.5.)

ilar terms. Finally, a link to remove the term was desired. An additional requirement was that definitions of the candidate terms to be substituted be provided via pop-ups. The DE suggested the use of a "book" icon to indicate this functionality.

4.4.4 Add term panel

This panel was designed to provide the *"add term"* functionality. The panel needs to display all the terms that can be combined with a given query. Since this is likely to be a long list, tools such as keyword search, sorting and grouping of the list were suggested.

Details of the design are explained in figure 4.6. It was desired that the results be displayed as a list, with options to group by ontology and by first letter, and to sort alphabetically and by number of results for the new queries. Additionally, a keyword search interface was suggested. All of these controls would be found above the list. Within the list itself, the terms were designed to be displayed similarly to those in the *"substitute/remove term"* panel, with the term name, result set size and a link to a pop-up of the term definition being provided. A final proposal for this panel was to display the terms on a tree representation of the ontology.

4.4.5 Results panel

This panel was designed to provide the *"view results"* functionality. This involves displaying the gene products resulting from a query. The suggestion was to display these as a list, with paging for result sets larger than 100, and the ability to export into the CSV (comma-separated value) format, understandable by spreadsheets. Links out to other sources of information were desired, as well as a link back to the Dasty view for each gene product.

The design of the results panel is shown in figure 4.7. The design suggests displaying the results as a paged list, with 100 results per page, ordered alphabetically. Switches for other orderings were also suggested. Important information on each gene product is provided, including its name, symbol and species. Links out to other data sources were suggested to be provided to the right of the key information. An extensive list of data sources was suggested, including cross-references to genomic databases, sequence information (nucleotide, peptide, cDNA and primer), lists of orthologues, functional domains and expression information. It was agreed, however, that most of these functionalities could

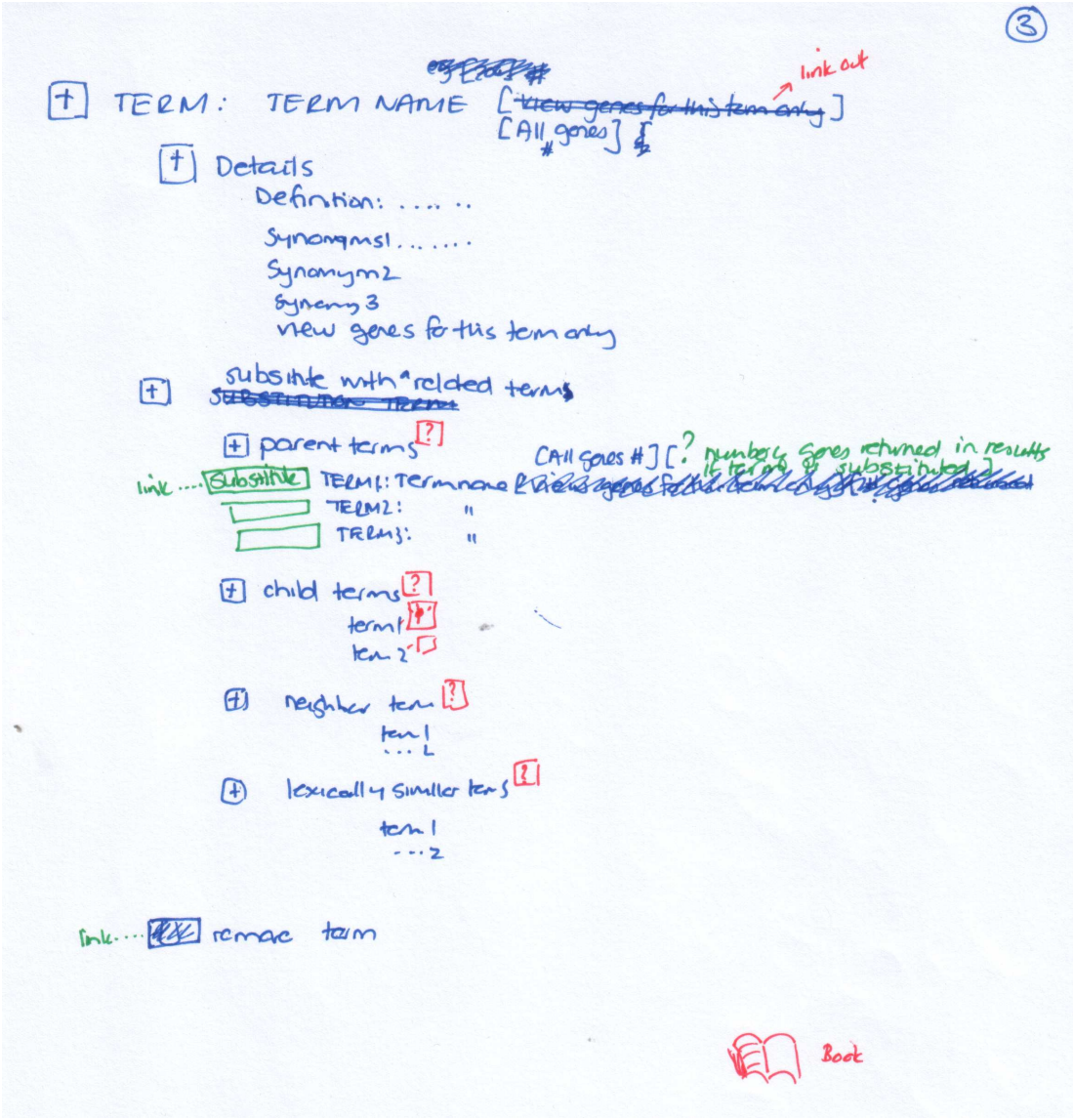

Figure 4.5: Paper prototype of the "substitute/remove term" panel.

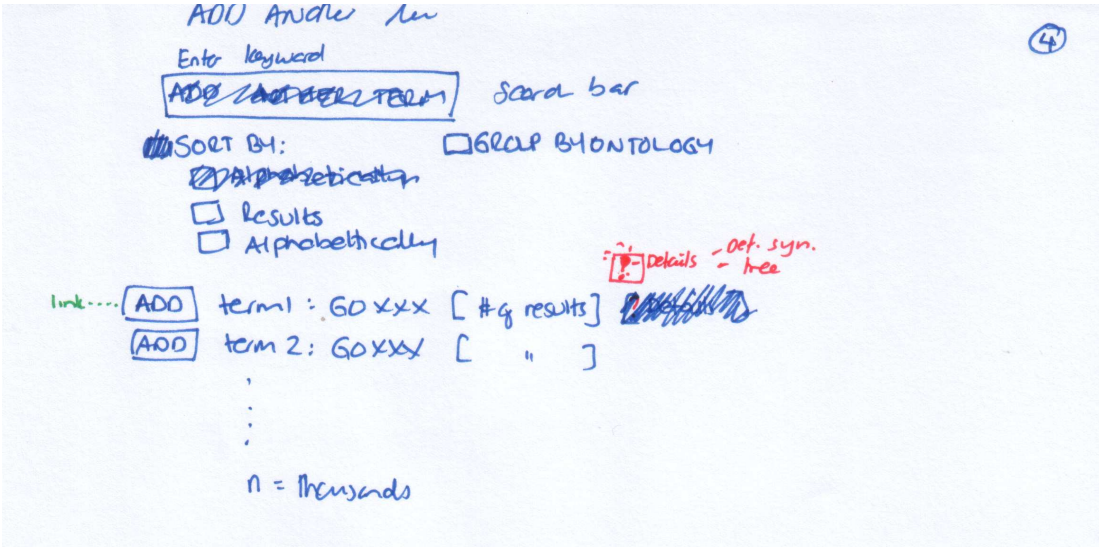

Figure 4.6: Paper prototype of the "add term" panel.

be provided using a relatively small number of links to other databases that specialised in providing some or all of this information.

4.5 Conclusion

By working collaboratively with a domain expert, a descriptive design for the user interface was developed, which greatly assisted the process of implementing OntoDas. During implementation, contact was maintained with the DE, in order to guide the process. The use of participatory design techniques proved to be extremely useful in creating a design that fit with the DE's way of thinking, and in gaining a better idea of how the DE works. Furthermore, it became clear that the use of ontologies both as an aid to construct queries and as a means of finding new information was feasible and genuinely useful to biologists.

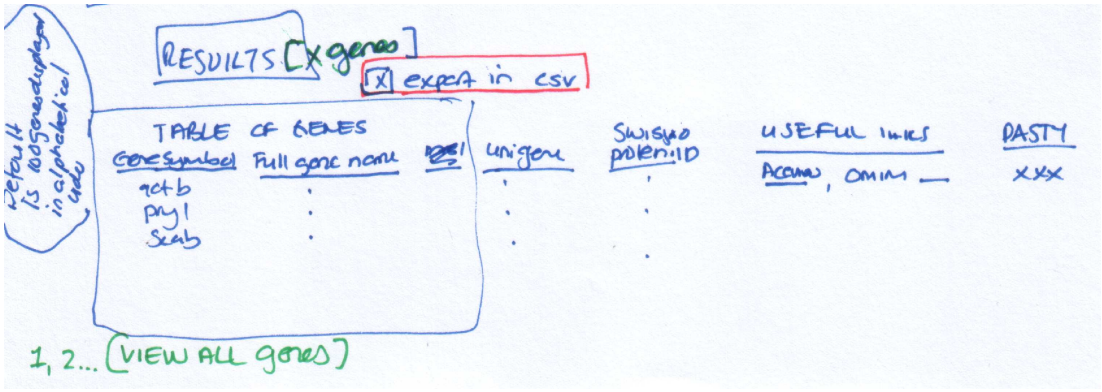

Figure 4.7: Paper prototype of the results panel.

Chapter 5

Implementation

Once the initial design had been created, development began. Implementation was constrained in part by technological factors. This chapter describes the technical consideration taken during the implementation of OntoDas. First, the overall architecture of OntoDas is introduced. Following this, the technologies used in creating OntoDas, and how they were used, are discussed. (This section includes a discussion of web services which are used by OntoDas.) The chapter concludes with a presentation of the final appearance of the user interface to OntoDas, by means of a scenario played out through screen shots.

5.1 System architecture

It was decided to make OntoDas web-based. In addition, the Ajax (Asynchronous JavaScript and XML) approach to web application architecture [39] was chosen. One reason for this choice was that it would ease integration with Dasty2, which was also Ajax based [54]. The Gene Ontology database was chosen as a model, and the MySQL database dump of that database [1] was chosen, prescribing MySQL [2] as the choice of back end database engine. As middleware between the JavaScript and MySQL, the Python web framework TurboGears [3] was chosen. The system architecture is illustrated in figure 5.1, and explained in more detail below.

[1] GO archive – `http://archive.geneontology.org/`
[2] MySQL – `http://www.mysql.com/`
[3] TurboGears – `http://turbogears.org/`

5.1.1 Ajax front end

The Ajax approach is to run an Ajax engine, written in JavaScript, within the client's browser. The engine handles locally the responses to user actions that don't require server round trips, and asynchronously handles those that do, so that the user's interaction with the application isn't stalled. Since system responsiveness is important for usability [73], this improves the usability of the system. Furthermore, network traffic is reduced, as the page can be updated without requiring a new copy of the entire page to be delivered to the user's browser by the web server.

OntoDas uses an Ajax engine to allow the user to obtain many different views of the available query terms and results, without delays due to server round trips. The OntoDasController JavaScript component (the Ajax engine) running in the DE's browser makes calls to functions provided by the TurboGears server. The functions are called using web services following the representational state transfer (REST) architecture [33]: Function names are specified using URLs, and arguments are passed as the arguments in an http GET request. The response from the server is delivered as a plain text web page, encoded in JavaScript Object Notation (JSON) [4]. The results of function calls are transformed into JavaScript objects, which are processed by the working code of the JavaScript component, and used to update the view by manipulating the browser's document object model (DOM) [110].

Full page refreshes occur whenever a new query is executed. This ensures that every query has a unique URL, which can be stored (for instance as a bookmark) and used to access the query again later. Ajax calls are used to populate the main query view with information as it returns from the server. This information included details on the query terms, options for substitution of the query terms, the set of terms which can be combined with the query, and the set of results. Since some of these sets can take several seconds for the server to compute, by displaying those that arrive, the user is not left waiting with a blank screen, but is provided with some information to interact with. Ajax is also used in the Dasty query construction view, to dynamically retrieve the size of the result set for the query being built as new terms are added.

For DOM manipulations, Ajax calls and JSON decoding, functions from the MochiKit JavaScript library [5] were used. (Numerous free Ajax libraries exist for JavaScript,

[4] JSON – `http://www.json.org/`
[5] MochiKit – http://mochikit.com/

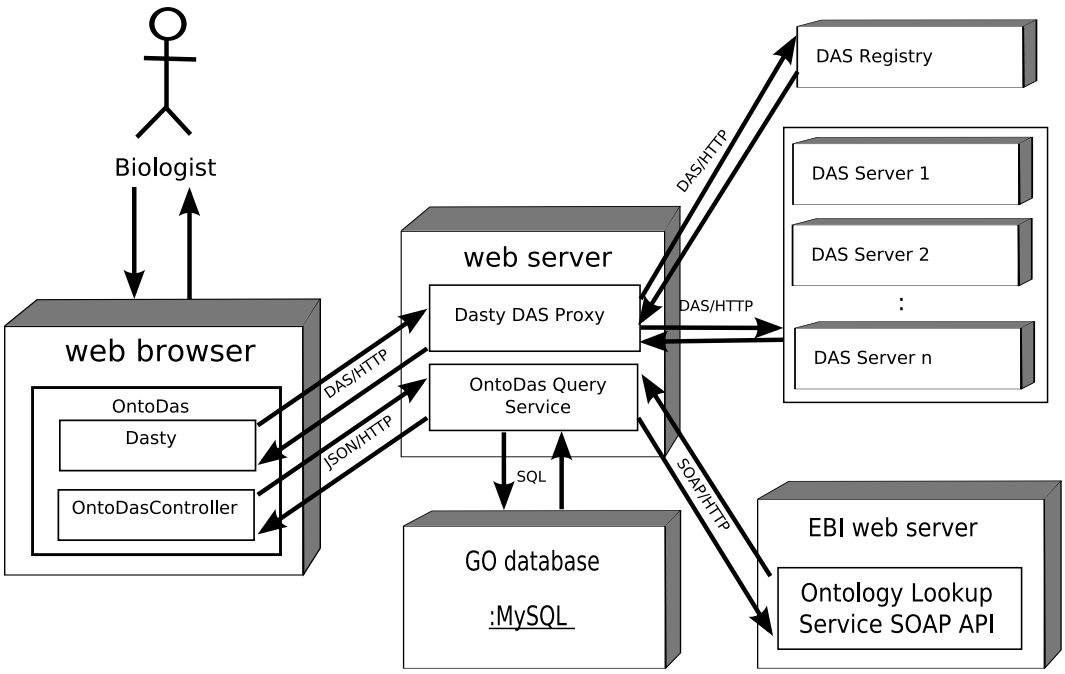

Figure 5.1: The system architecture of OntoDas. Platforms and the components running on them are shown, as well as the protocols used for encoding data.

with largely similar functionality and performance. MochiKit was chosen due to the author's preference, and because it is the standard library distributed with TurboGears). For rendering the popup windows, a library by Brian Gosselin was used [6]. Panels were rendered using code in Dasty2 [7]. All other JavaScript functionality was created by the author.

5.1.2 TurboGears as middleware

TurboGears is a lightweight web framework that follows and reinforces the Model-View-Controller software design pattern [60]. In this paradigm, the application is split into three components: The model component is the domain-specific implementation of the application's central structure. Views deal with everything graphical, requesting data from the model and displaying it. Controllers mediate between input from outside of the application, the views and the model. In TurboGears, the view component consists either of dynamic web pages whose content can be altered using a templating language, or an Ajax

[6] Popup Windows – `http://scriptasylum.com/misc/popup/popup.html`
[7] Dasty2 – `http://www.ebi.ac.uk/dasty/`

interface. Controllers consist of Python functions that pass results from the model on to the templates, or, with a small change in code, to JavaScript code as JSON web services. The model consists of an object relational framework, SQLObject [8], which provides access to a database, which can be MySQL. Any custom Python code can also be used as the model. TurboGears provides the advantage that creating JSON web services and connecting them to a database back-end can be done relatively quickly and easily.

A TurboGears controller was created to provide the JSON web service to the Ajax component of OntoDas. For simpler queries to the GO MySQL database, SQLObject was used. For more complex queries involving multiple joins, queries in raw SQL, with some Python, were used. The code for these queries is discussed in more detail in the next section.

5.1.3 Query execution – Python and MySQL

The gene ontology was chosen as a model against which OntoDas was developed primarily because, as mentioned in chapter 2, it is the best known and most popular bio-ontology. It was also chosen because the DE was interested in information represented by GO (although she was also interested in several other ontologies, particularly those used to represent mouse genome information). Finally, GO is freely and easily available, in the form of a MySQL database dump containing all GO terms, the gene products they annotate, and extensive metadata. The availability of a dump which could be easily loaded into MySQL made the GO MySQL database an ideal candidate for creating OntoDas against.

The GO MySQL database was constructed with the transitive closure of the graph precomputed and stored in a table, so as to speed up the task of finding all the descendants of a term. Initially, this was used to execute queries, but it proved to be quite slow. (Queries took longer than one minute to execute in some cases.) Although the transitive closure had been computed, this only extended as far as the ontology terms themselves, but did not include associations with gene products. As an additional optimisation, two tables were created for storing the complete transitive closure, by directly storing the relationships between each term in the ontologies and every gene product that term and its descendants was associated to. A script was created to pre-compute these values to store them in the database, and OntoDas was altered to make use of them. This enabled an approximate tenfold increase in speed of queries, and hence an increase in the responsiveness of the

[8] SQLObject – `http://www.sqlobject.org/`

interface.

In the query engine, queries are executed using a combination of MySQL and Python. Multiple-term queries are executed by retrieving the set of all the gene products associated with each term, then using Python's built-in set library to determine the intersection of these sets. Substitutable terms are found using simple MySQL queries, then filtered by executing each possible new query, and removing those that produce no results. Combinable terms are found by finding every term associated with every gene product produced as a result of the given query. Each potential new query is then executed to determine the size of the result set.

To find terms lexically similar to query terms, Ontology lookup service (OLS) [24] was used. OLS is an ontology text search engine which regularly downloads all of the OBO ontologies, and indexes them using the powerful Lucene text search engine [9]. Although OLS runs stand-alone, it makes use of (and provides to third party users) a set of simple object access protocol (SOAP) web services to it functionality. This has the benefit that third party applications can make use of OLS in a language-agnostic manner. Thus, although OLS is written in Java, and OntoDas is written in Python, OntoDas can and does make use of OLS.

The OLS SOAP services are accessed by the OntoDas query execution engine, using the ZSI (Zolera SOAP Infrastructure) library [10]. There were two reasons for choosing to call OLS from within the web server, rather than directly from JavaScript: Firstly, JavaScript code is usually run in "sandbox" mode, and can only access web services from within the same domain as the one the script was served from [42], so would be unable to access OLS directly. Secondly, it was desired that the OLS results be filtered to only include those producing non-empty queries, which required the query execution engine. By providing a proxy to the OLS web service, OntoDas implements the REST-Based Model View Controller Pattern [42]; Dasty2 uses a similar architecture [54].

5.2 Final appearance of the user interface

The final screens implemented the core functionality laid out in the design described in the previous chapter. Not all of the initially designed features were implemented,

[9] Apache Lucene – `http://lucene.apache.org/`
[10] Zolera – `http://pywebsvcs.sourceforge.net/`

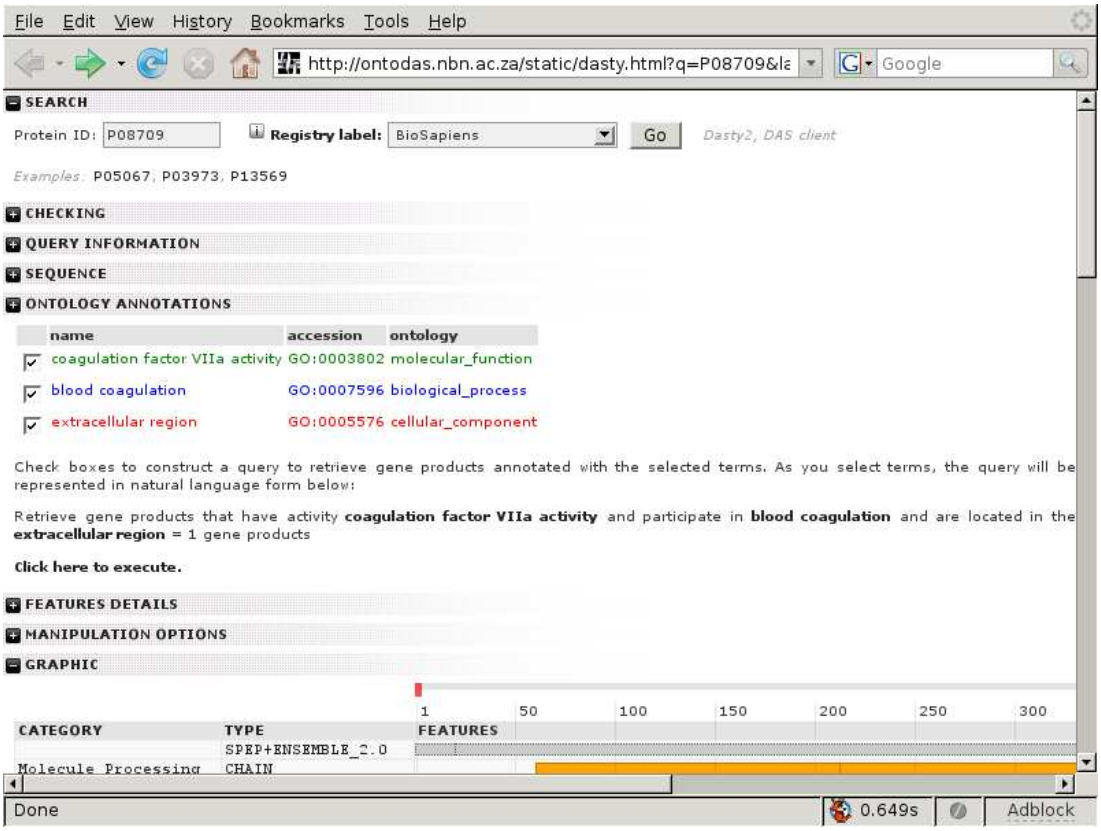

Figure 5.2: A screen shot of Dasty2, showing the ontology term viewer extension, based on the design shown in figure 4.3, and showing the first step in the scenario presented in section 4.1.2. This screen can be viewed at `http://ontodas.nbn.ac.za/static/dasty.html?q=P08709`

due to time constrains, but features agreed to be critical to the success of the system were implemented. To illustrate these features, the scenario laid out in section 4.1.2 is played out in the following sections using screen shots from OntoDas.

5.2.1 Viewing details of a gene product

In the first step, the DE views the sequence and ontology annotations for Coagulation Factor VII (UniProt accession P08709) in Dasty2. The Dasty view is shown in figure 5.2, with panels collapsed so that the ontology annotations panel is highlighted. The query constructed by the DE is shown, with all three ontology terms selected using the check boxes, and the natural language query displayed below. A detailed instruction message is

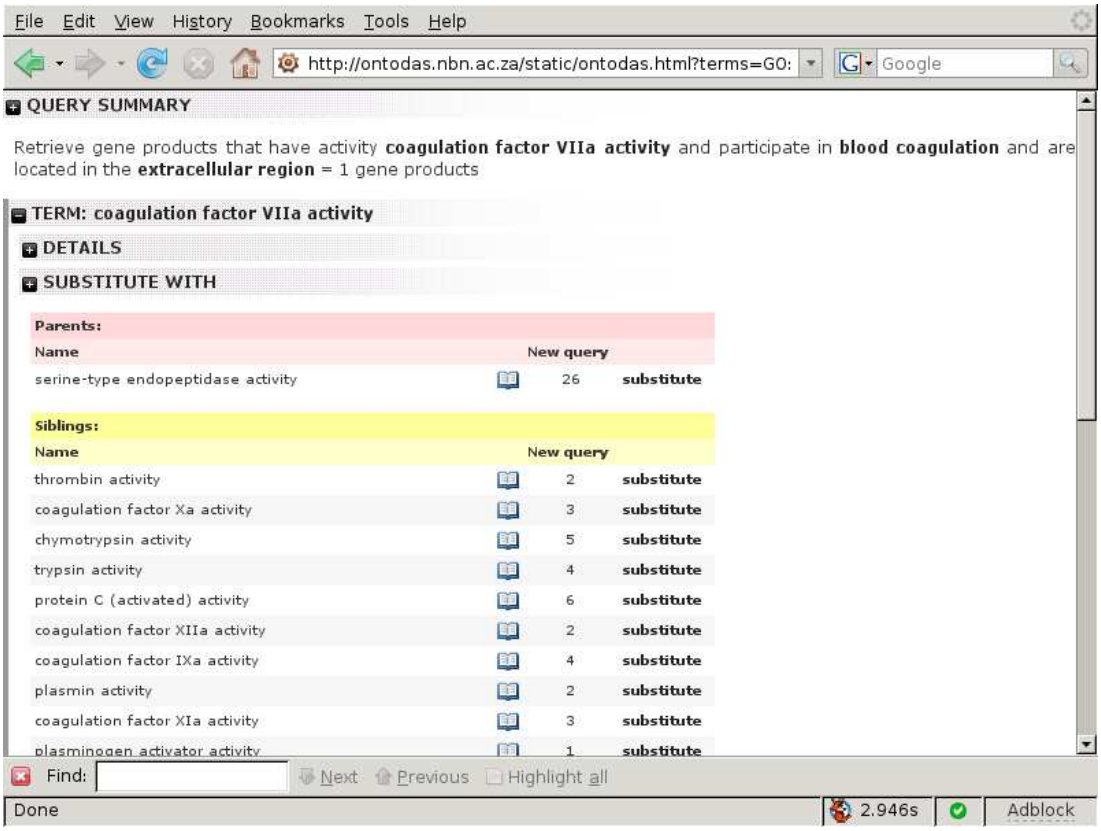

Figure 5.3: A screen shot of the "modify term" panel. As described in section 4.4.3, the panel contains three sub-panels. The first, providing a description of the term, is shown here collapsed. The third, providing the "remove term" functionality, is further down on the screen. The focus here is on the "substitute term" sub-panel. This screen can be viewed at `http://ontodas.nbn.ac.za/static/ontodas.html?terms=GO:0003802,GO:0007596,GO:0005576`

provided, as suggested by the design with the DE. The query can be executed by clicking on the prominent *"Click here to execute."* link. Clicking this link takes the DE to the main OntoDas view for the query.

5.2.2 Substituting a term

In the second step, the DE begins by viewing the results of the query. The query summary indicates that only one gene product was returned. They decide to examine the term *coagulation factor VIIa activity* by opening the corresponding panel, as shown in figure 5.3. Under "parents", they see the term *serine-type endopeptidase activity*, which,

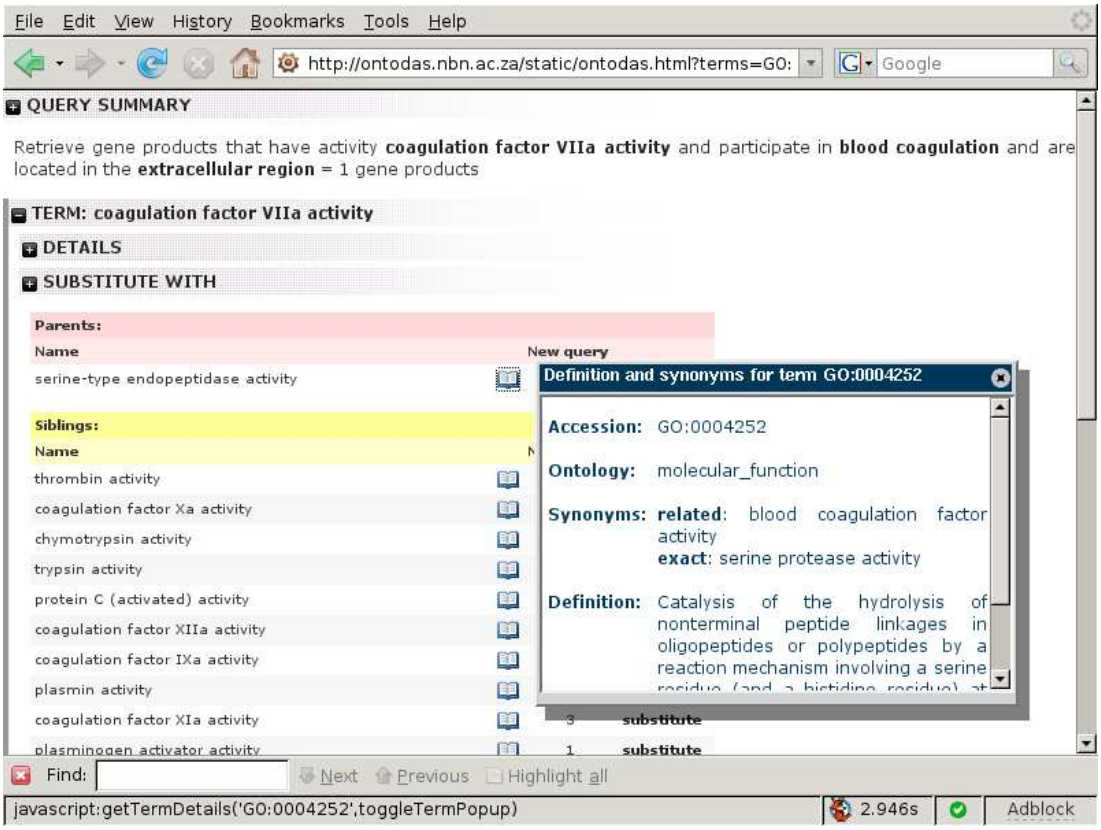

Figure 5.4: A screen shot of the "modify term" panel, showing the popup window displaying term definitions and other details.

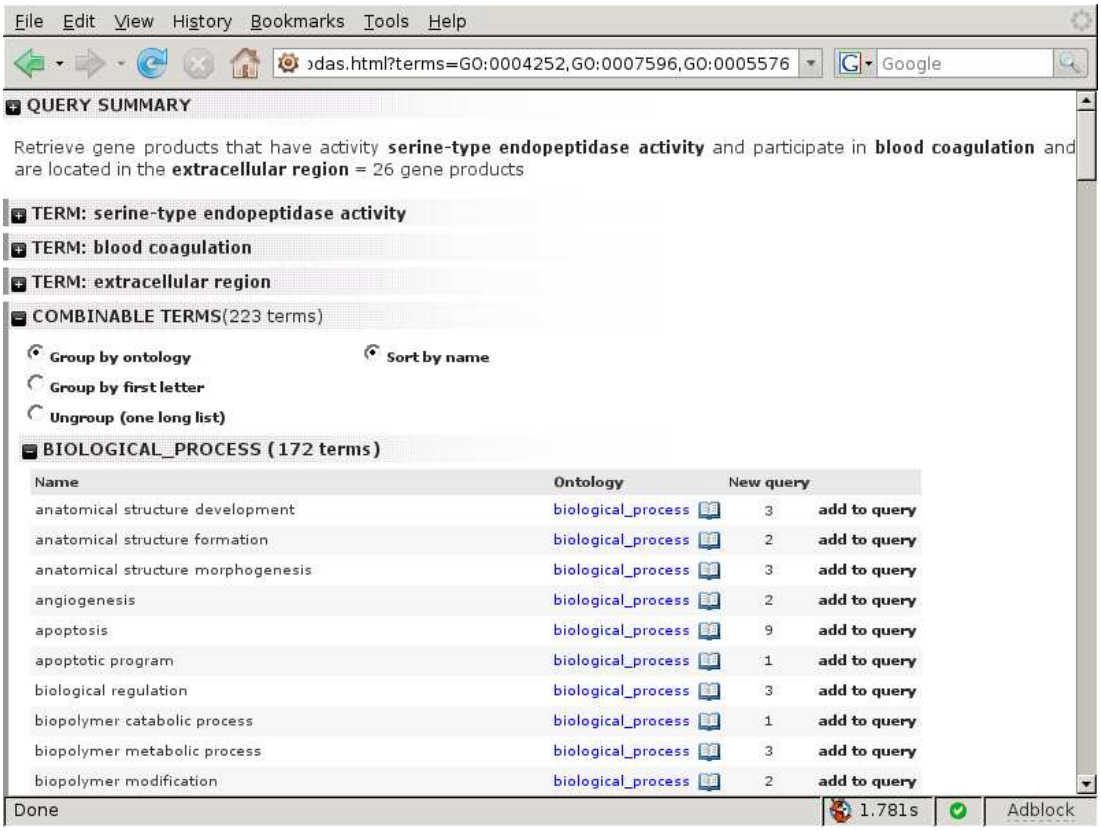

Figure 5.5: A screen shot of the "add term" panel. Various options are provided for grouping the list of terms. Sorting is automatically done by name, but other sorting options were not implemented. The proposed tree view was also not implemented. This screen can be viewed at `http://ontodas.nbn.ac.za/static/ontodas.html?terms=GO:0004252,GO:0007596,GO:0005576`

if substituted, would produce a query containing 26 gene products. They decide to view details of this term by clicking on the "book" icon, to confirm that it is what they are looking for. This pops up the window shown in figure 5.4, showing the accession, ontology, synonyms and definition of the term. Satisfied that the term is the one they want, the DE closes the popup window and clicks on the "substitute" link to execute the new query.

5.2.3 Adding a term

In the third step, the DE chooses to examine the terms that are combinable with the query they have just constructed. As shown in figure 5.5, they view the terms grouped by ontology. The term *apoptosis* catches their eye; this term would, if added, produce a

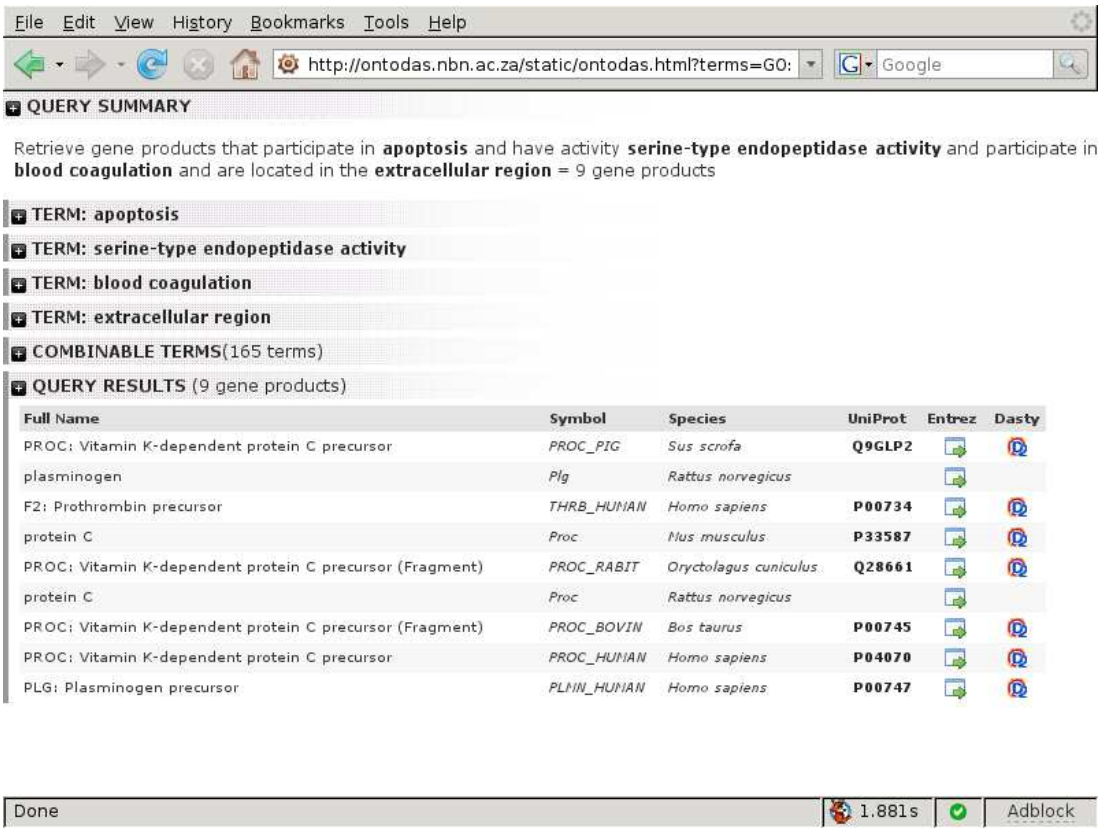

Figure 5.6: A screen shot of the "results" panel. Links out to other databases, as well as back to Dasty2, are provided. This screen can be viewed at `http://ontodas.nbn.ac.za/static/ontodas.html?terms=GO:0006915,GO:0004252,GO:0007596,GO:0005576`

query returning 9 gene products. The DE clicks on the "add to query" link to execute the new query.

5.2.4 The results view

In the final step, the DE views the results, and is particularly interested in protein C. They open several new browser windows using the links provided, to view details of the gene product in other databases. They also open some Dasty2 windows, to retrieve sequence annotations for the gene product.

5.3 Performance and size

5.3.1 Size of system

In terms of the size of OntoDas two aspects are worth noting: the size of the codebase (both the server-side Python and the client-side JavaScript) and the size of the database, both in terms of number of records and disk space usage.

In terms of the JavaScript code, which actually gets delivered to the client's web browser, the core OntoDas code is only approximately 29Kb, with the MochiKit and popup libraries it depends upon taking up 111Kb and 13Kb respectively. Dasty2, which also gets delivered when it is used, contains 273Kb of JavaScript, along with 411Kb of supporting libraries.

In terms of database size, the gene ontology itself contains approximately 25 000 terms. However, the greater part of the database used was taken up by the two million gene products contained therein, many of which also have sequence information stored. As a consequence, the dump itself is a 1.0Gb gzipped file, which decompresses to approximately 2.5Gb of text. Once loaded into MySQL, with all the indices and metadata generated, this uses up to 5Gb of actual disk space.

5.3.2 Performance

Two key aspects affect the performance (and hence usability) of OntoDas: performance of the client's web browser when executing the JavaScript, and performance of the Python server when executing web service requests.

Most operations within the browser executed in less than one second. Notable exceptions were the population of the combinable terms and results list when these lists contained more than a few hundred entries. When the combinable terms lists contained a thousand or more terms, the code to group these could take around ten seconds to execute, locking up the browser in the process. (more cases?)

Like the browser code, most of the web services responded in less than one second. The slowest, however, were the functions to find combinable terms, and to find the size of the result sets of these. In particular, the latter had to compute the entire result sets for every new combination of terms. Consequently, queries with one thousand combinable terms or so could take between 10 and 30 seconds to execute. To improve user experience, OntoDas

was designed to first load the combinable terms themselves (a relatively fast operation), then make a second request to find the sizes of the new queries. (more cases? 2000+?)

5.4 Conclusions

OntoDas makes extensive use of free open source software libraries and applications to deliver the functionality specified in the previous chapter. By following software engineering best practices for designing software architecture, OntoDas was built to be responsive and supportive of the target users, domain experts in molecular biology.

Chapter 6

Discussion

This thesis set out with the purpose of solving the problem of providing conceptual querying functionality to biologists. The approach taken was to develop a software tool that employed standard HCI and IV techniques and was created following the participatory design methodology, with the intended purpose of providing this functionality. In demonstrating that the development of OntoDas has solved the problem, OntoDas is first compared with the existing systems that were reviewed in chapter 3. In this comparison, it is shown how OntoDas addresses the shortcomings of these tools and combines their strengths into a solution to the specific problem at hand. In the following section, an argument is posed from the literature that the use of HCI, IV and PD techniques has resulted in a tool which provides cognitive support and hence relieves cognitive load when constructing conceptual molecular biological queries. The chapter concludes with a discussion of the shortcomings of both OntoDas itself and of the methodology used to create it, which forms a basis for the subsequent chapter proposing future work.

6.1 Comparison with existing tools

In determining whether OntoDas overcomes the shortcomings of similar existing software, the criteria used to assess these tools have been used below to assess OntoDas. This is followed by a new version of table 3.1, expanded to include the assessment of OntoDas.

1. **Problem domains:** OntoDas was applied to the same domain as AmiGO: That of gene function, using the 25 000 term gene ontology, and a database of approximately 2 000 000 gene products from a wide variety of different species of organisms.

2. **Types of queries:** OntoDas allows the finding of gene products commonly annotated with any number of given query terms, with the only restriction being that the terms must actually have gene products in common. Of the other tools, only Flamenco provides this functionality, and on a significantly smaller data set.

3. **Support for initial term finding:** OntoDas does not specifically support initial term finding although the Dasty2 component can assist this. A real-time query expansion [107] interface for finding initial ontology terms, based on ontology lookup service would be a candidate solution to this.

4. **Support for combining terms:** OntoDas provides extensive support for finding combinations of terms: Firstly, the Dasty2 interface enables the execution of queries using the annotations from a gene product. Secondly, OntoDas incorporates a specific "combinable terms" panel, showing for the given query all the terms which can be combined with it to produce valid new queries. Finally, OntoDas enables fine tuning of the query by providing valid choices for substituting terms in the query with other, related terms. Like Flamenco, OntoDas provides query previews by displaying the size of the result sets of potential new queries, and by restricting the set of options presented to the DE to those that produce non-empty result sets.

5. **Display of results:** OntoDas displays results as a list, without options for paging. Links are provided to external databases, and back to Dasty2 for each gene product. Although requested in design, CSV output of results has not been implemented. Paging, sorting and CSV output would be good directions for future work on OntoDas.

6. **User involvement in development:** Participatory design involving one wet lab biologist was used extensively in the design and development of OntoDas.

7. **Technologies:** OntoDas is primarily Ajax based, with MySQL and web services forming the back end, and the TurboGears web framework forming middleware between these.

Criterion	AmiGO	MartView	GViewer	DOPE	Flamenco	OntoDas
1.Problem domains:	Gene Ontology (25K terms) to multi-species gene DB (2M entries)	Gene Ontology (25K terms), individual species gene DBs (30K entries each)	Gene Ontology (25K terms), several others (5K each); rat gene and QTL data (9K entries)	EMTREE ontology (50K terms), custom literature DB (10M entries)	image metadata thesauri (4K terms each); image databases (35K entries each)	Gene Ontology (25K terms) to multi-species gene DB (2M entries)
2.Types of queries:	single term only; narrowing by evidence code, species.	multiple terms, only one term, evidence code per ontology. Only one species per query	AND, OR, NOT over several ontologies; no species / evidence code narrowing	AND queries of any number of terms; no narrowing criteria	AND queries across any number of terms; only one term per orthogonal ontology; supplementation with keyword search	AND queries across any number of terms
3.Initial term finding:	forms, tree navigation	QuickGO browse, search, but no tie-in with MartView interface	no support	Form with intelligent term suggestion; tree navigation	keyword search, dynamic tree navigation	not yet implemented
4.Combination finding:	no support	no support	no support	valid combinations with first term shown, but limited support for 3 or more	all valid combinations with current query displayed, also size of result set (query previewing)	extensive; all valid combinations displayed, as well as size of result set (query previewing)
5.Display of results:	paged table, links to detail on each query, links to external information.	simplistic but configurable to be richer; spreadsheet export	highly visual SVG; no table but proprietary spreadsheet export	interactive, visual cluster map; problems with scalability	page-able table, links to detail on each entry, ability to construct new queries from annotations of entry	table with links out; paging and CSV download not yet implemented
6.User involvement:	minimal, though possibly via mailing list	no evidence of any	no evidence of any	usability evaluation post-development	multiple cycles of testing, re-development, evaluated against a baseline	extensive participatory design throughout the life cycle
7.Technologies:	web-based: Perl, MySQL	web-based: Perl, BioMart	web-based: Java JSP and Oracle 9i PL/SQL	Desktop-based: Java / Swing, ClusterMap, Sesame RDF store	Web-based: Python WebWare, MySQL, Java / Lucene optional	Web-based: Ajax (MochiKit and others), Python TurboGears, MySQL, web services

Table 6.1: A comparison of five tools for facilitating the construction of ontology based queries with OntoDas, according to the criteria specified in chapter 3.

6.2 Application of IV and HCI techniques

As detailed in chapter 2, the application of HCI and IV techniques can provide cognitive support to users. OntoDas differs from similar tools targeted at the biological domain in that it takes into account techniques from information visualisation and HCI. Firstly, as in Flamenco, query previews [83] are provided, ensuring that DEs will never encounter a zero-result query. This provides cognitive support by reducing the query space which DEs must search to find queries of interest, thereby reducing the load on their short-term memory [20]. Secondly, "information scent" (*the (imperfect) perception of the value, cost, or access path of information sources obtained from proximal cues [80]*) is provided to guide users to queries of interest, in the form of query previews and the provision of term details via popup windows. Thirdly, a natural language representation of the query can be far easier for users to understand than a more technical representation [73] – by using best practices established by convention within the bio-ontologies community [93], it was possible to provide this in OntoDas. Lastly, additional grouping and sorting features assist users in finding terms of interest in the sets of combinable terms offered to them. In conclusion, by employing techniques these techniques, the development of OntoDas has delivered a tool that relieves cognitive load when constructing ontology based queries within molecular biology.

6.3 Participatory design

As is described in chapter 2, the biological domain is a highly complex one. Understanding the domain and the challenges facing researchers within it can be difficult for designers of bioinformatics software. Even with the powerful techniques available, designing software that matches the way biologists work presents an additional obstacle [65]. When designing OntoDas, participatory design provided a "third space" between the domains of the software designer (the author) and the domain expert, as is described in the literature [69, 14]. As a result, ideas were produced and a system designed which neither the designer nor the DE could have achieved alone, and which the DE expressed satisfaction with.

However, a flaw in the PD methodology used is that the number of domain experts involved in the design was only one. Although no explicit study of the importance of group size has been carried out within the PD literature, implicitly most papers recommend sizes

of at least two to ten; examples include [68, 102, 16]. A difficulty with small group size is that it may make the final product too specific for more general use [1, 70]. For the technique used, cooperative prototyping, more DEs, at least five in total [16] should ideally have been involved. Better still would have been to involve DEs from a variety of different laboratories who were working on different areas within molecular biology. This could form the basis for further design work, provided DEs could be found to assist. The creation of new scenarios besides the ones presented in chapter 3 could also be of benefit, as the scenarios themselves could be too specific due to the small number of DEs they were gathered from. Regardless, it is almost certain that the final tool produced is better suited to biologists in general than if the developer had undertaken the design single-handedly.

6.4 Limitations of OntoDas

6.4.1 Performance and scalability

As mentioned in the previous chapter, queries which have large sets of combinable terms (1000 or more) result in poor performance. On the browser side, grouping and sorting can take up to ten seconds to complete, locking the browser in the process. From a usability perspective, this is highly undesirable [73]. On the server side, the computation of both the set itself, then of the size of the prospective new result sets, can take up to 30 seconds or more. A danger exists that such a delay could disrupt a user's flow of query construction. Both of these issues could be addressed to make OntoDas a better system.

It is possible to ameliorate these issues without needing to seriously redesign OntoDas. To improve browser responsiveness, sorting and grouping could be shifted to the server side, with web services providing result sets already grouped and sorted. To accelerate the finding of combinable terms, measures have already been taken by pre-caching the result sets for individual terms. Further optimisations, such as executing the complete query natively within MySQL using stored procedures, could make improvements. However, an obvious solution would be to implement lazy loading in the user interface – only retrieving particular subsets of combinable terms when a user opens the relevant panel. Further in this vein, rather than displaying long lists of terms (which are difficult for a user to search through and make sense of [20], apart from the performance overheads), results could be displayed as lazily-loaded, paged lists. To supplement this functionality, text search could

be implemented within the results – this functionality was in fact suggested by the DE in the design, but not implemented. Another fairly simple measure would be to use the new curator-approved only download of the database, which was made available after OntoDas was created. This version contains one tenth the number of gene products (approximately two hundred thousand compared to two million). Since only these links are currently used in OntoDas anyway, this could substantially reduce the number of options MySQL would have to search to find results. Together, all of these features could improve the responsiveness, and hence usability, of OntoDas.

6.4.2 Unsupported cases

Although OntoDas supports most combinations of terms, two general cases exist that are not supported. The first is that where a single term is given which is extremely general. Examples include the terms *binding*, *cytoplasm* and *metabolic process*, which return 43 293, 48 828 and 56 334 gene products respectively. Although the exact number of terms combinable with each of these is not known due to the extremely high cost of computing that, it is likely that the set of terms included most of the gene ontology. Apart from being incapable of computing such a large set of combinable terms on the server side before the browser times out the connection, if the result set were sent to the browser, it is likely that it would crash. Thus, OntoDas does not support extremely general single term queries.

However, such queries are unlikely to be of interest, as they are extremely general and do not assist biologists in isolating specific gene products of interest. They are also quite simple, and thus not the types of queries OntoDas aims to support. Nevertheless, it is possible that users may wish to use such terms as a starting points for a query. This is a possibility which could be investigated in future PD work. It is also possible that the performance improvements suggested in the previous section will ameliorate this problem. If not, then it would provide for interesting future work to investigate how best to represent queries in these cases.

The second case is where a term is so specific that it has not yet been used to annotate any gene products, and hence will never be involved in an OntoDas query. Examples include *larval burrowing behavior* and *3-galactosyl-N-acetylglucosaminide 4-alpha-L-fucosyltransferase activity*. However, this is unlikely to be immediately problematic to OntoDas' overall usefulness, since the purpose of using query previews within OntoDas is

specifically to avoid encountering such queries. Nevertheless, a possibility does exist that a user may wish to know when a term has no gene products associated, and may wish to use that term as a starting point for a query (for instance, when its parents or siblings do have gene products associated). Again, this is a case which could be brought up in future PD work, and act as a driver for future development.

6.4.3 Unimplemented features

Several features were suggested by work with the DE in chapter 3 which were not implemented in OntoDas. As the previous paragraph mentions, both text search and paging through long list of items are two. Also requested was the ability to begin constructing a query with a single ontology term. Such functionality could be readily adapted from OLS, which is open source [24]. A small, but easy to implement feature is the export of result sets in a spreadsheet-compatible format, such as comma separated value (CSV). Finally, the implementation of pop-up help windows next to difficult terms to understand (such as "siblings") could make OntoDas easier to use without training. All of these are suggested in the following chapter as future work.

6.5 Conclusions

By employing techniques from HCI and IV, OntoDas provides biologists with cognitive support when constructing conceptual queries in ways that previous tools have not. By using PD, it was possible to adapt these techniques to the way in which a biologist works, thereby ensuring that they were not misapplied. Shortcomings remain, both in terms of unfulfilled features requested by the biologist, and in terms of there being only one domain expert involved in the PD process. Nevertheless, OntoDas is a tool which relieves cognitive load when constructing ontology-based molecular biological queries, and hence provides a solution to the problem posed by this thesis.

Chapter 7

Future Work

OntoDas, in its current state, represents a working software system that solves the problem proposed by this thesis. The development carried out thus far can be regarded as one or two iterations in an iterative software development cycle: The core functionality has been implemented, but, as mentioned in section 6.4.3 there are lower priority requirements that were suggested during design that have yet to be fulfilled. Furthermore, there are performance issues, as detailed in section 6.4.1, which suggest solutions through further development. Finally, it is clear that the participatory design work carried out could have substantially benefited from the involvement of more domain experts, while a longitudinal usability study could ensure that OntoDas truly does match the needs of biologists.

This chapter proposes plans for the future development and wider application of OntoDas. The chapter commences with a series of proposals for addressing the issues already raised in the discussion chapter through the development of new features. This is followed by a proposal for a usability study for evaluating OntoDas in order to provide insights for further development. The chapter concludes with suggestions for applying OntoDas to new contexts.

7.1 Improvements suggested by existing analysis

Section 6.4 details numerous shortcomings in the existing implementation of OntoDas. These limitations have give rise to several possible directions for future development, which are outlined below:

- Implement an interface, based on OLS, for finding a single term to begin constructing

a query with.

- Enable downloading of results in CSV format.
- Include pop-up contextual help windows for portions of the interface.
- Shift the grouping and sorting of combinable terms to the server.
- Implement lazy loading in the user interface. (Only load the contents of a panel when it is opened, rather than loading the contents of all panels on page load.)
- Implement paging of results and combinable terms.
- Implement keyword search through results and combinable terms.
- Use the curator-approved only version of the GO database.
- Optimise the execution of queries, possible using MySQL stored procedures.

However, it is important to note that biologists may desire features not listed here, and that any further extension process should ideally occur in tandem with a participatory design process to ensure that development effort is not wasted on undesired features [62].

7.2 Evaluation of usability

Evaluation of an information visualisation system [82] and even of HCI systems in general [73], is hard. The technique of heuristic evaluation, wherein usability experts evaluate an interface according to established best practices [74], can be used. However, this technique tends to catch a relatively low proportion of usability problems in a system [53], so is best suited to very rudimentary initial evaluation to inform early design work [73]. Usability testing, wherein users carry out routine tasks using an interface and the results observed [73], could provide an alternative. Yet, even task-based analysis has been criticised in the context of IV systems [82, 21], since the systems are often designed to allow unfettered exploration, making specific use case instances difficult to predict. A potential solution to this problem has been proposed in the form of "insight-based evaluation" [86]; this methodology recognises that the goal of an IV system is to promote the reaching of insights, and uses a task-free system for evaluating the discovery of insights. Some disadvantages of this system are in that it requires greater effort to capture and code

results – in particular it is recommended that highly skilled domain experts are recruited to assist in this process. In summary, a series of options exist for evaluating the usability of OntoDas, with increasing efficacy but also increasing difficulty in performing. This suggests a potential methodology:

1. Evaluate OntoDas using several usability experts, according to established usability guidelines, to identify fundamental problems with usability. (To some extent this has occurred already through the analysis carried out in chapter 6.)

2. Gather new scenarios of use from a broader group of users, in a participatory manner.

3. Test OntoDas with these and other users, using the new (and existing) scenarios.

4. Ideally, recruit domain experts to assist in coding the results from an insight-based evaluation, then carry out this evaluation, allowing users to freely explore OntoDas and coding the results of their exploration.

All of these steps could be extremely useful in informing future development on OntoDas.

7.3 Application to broader contexts

OntoDas at present only works with the gene ontology. It could be beneficial to apply it to a broader set of ontologies used to annotate gene products, as was GViewer [92]. Some examples of interesting ontologies which could be loaded include the mammalian phenotype and anatomy ontologies used to annotate the mouse [11] and rat [104] genome databases, as well as the Medical Subject Headings (MeSH) [67] used to annotate PubMed. A possible means to do integrate this data into OntoDas would be to use the loading scripts provided by the GO consortium, which enable any ontology in the OBO file format to be loaded into the GO MySQL schema. Additional scripts would be needed to load the relations between the ontology terms and the gene products. In this way, OntoDas could be applied to a new and broader set of data, to potentially provide insight into it.

Chapter 8

Conclusions

The goal of this thesis was to solve the problem that *"'Although the standard usage of ontologies within molecular biology provides a basis for the construction of conceptual queries within the domain by the domain experts, no visual tool exists to enable them to do this."* In doing so, techniques from human computer interaction and information visualisation techniques were applied to create such an interface, and participatory design was applied to ensure that the interface was applicable to the domain of molecular biology:

1. By using techniques including query previews, provision of information scent and natural language query representation, OntoDas provides cognitive support to biologists, to make the functionality it provides accessible to them.

2. By involving a biologist through interviews and cooperative prototyping exercises, as well as via correspondence, OntoDas was developed so as to fit into the domain of molecular biology.

3. By supporting biologists in a way that is compatible with their way of working, OntoDas enables them to construct conceptual queries using ontologies within the domain of molecular biology.

Appendix A

Publications resulting from this work

A.1 Conference papers

1. O'Neill, K.M.; Garcia-Castro, A. and Jacobson, D. (2007) *Knowledge Driven User Interfaces for Complex Biological Queries*, Proceedings of the First Southern African Bioinformatics Workshop

2. O'Neill, K.; Schwegmann, A.; Jimenez, R.; Jacobson, D. and Garcia, A.. (2007) *OntoDas – integrating DAS with ontology-based queries*, Joint AFP - BioSapiens Special Interest Group, 15th Annual International Conference on Intelligent Systems in Molecular Biology

3. O'Neill, K.; Schwegmann, A.; Jimenez, R.; Jacobson, D. and Garcia, A.. (2007) *OntoDas – integrating DAS with ontology-based queries*, Bio-ontologies Special Interest Group, 15th Annual International Conference on Intelligent Systems in Molecular Biology

A.2 Conference posters

1. O'Neill, K.M.; Garcia-Castro, A. and Jacobson, D. (2007) *Visually enabling the construction of ontology-based cross-database queries*, 15th Annual International Conference on Intelligent Systems in Molecular Biology

Bibliography

[1] C. Abras, D. Maloney-Krichmar, and J. Preece. *Berkshire Encyclopedia of Human-Computer Interaction*, chapter 3: User-Centered Design. Berkshire Publishing Group, 2004.

[2] M. Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, 2006. Student Member-Jeffrey Heer.

[3] K. Akrivi, T. Elena, H. Constantin, L. Georgios, and V. Costas. A comparative study of four ontology visualization techniques in protege: Experiment setup and preliminary results. In *IV '06: Proceedings of the conference on Information Visualization*, pages 417–423, Washington, DC, USA, 2006. IEEE Computer Society.

[4] F. Al-Shahrour et al. FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes. *Bioinformatics*, 20(4):578–580, 2004.

[5] R.B. Altman, M. Buda, X.J. Chai, M.W. Carillo, R.O. Chen, and N.F. Abernethy. RiboWeb: An ontology-based system for collaborative molecularbiology. *Intelligent Systems and Their Applications, IEEE*, 14(5):68–76, 1999.

[6] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[7] E. Baehrecke, N. Dang, K. Babaria, and B. Shneiderman. Visualization and analysis

of microarray and gene ontology data with treemaps. *BMC Bioinformatics*, 5(1):84, 2004.

[8] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000.

[9] M.H. Beers, R. Berkow, et al. *The Merck manual of diagnosis and therapy*. Merck Research Laboratories, 1999.

[10] R.K.E. Bellamy, T. Erickson, B. Fuller, WA Kellogg, R. Rosenbaum, J.C. Thomas, and T.V. Wolf. Seeing is believing: Designing visualizations for managing risk and compliance. *IBM Systems Journal*, 46(2), 2007.

[11] J.A. Blake, J.T. Eppig, C.J. Bult, J.A. Kadin, J.E. Richardson, et al. The Mouse Genome Database (MGD): updates and enhancements. *Nucleic Acids Research*, 34(Database Issue), 2006.

[12] C. Blaschke, L. Hirschman, and A. Valencia. Information extraction in molecular biology. *Briefings in Bioinformatics*, 3(2):154, 2002.

[13] O. Bodenreider and R. Stevens. Bio-ontologies: current trends and future directions. *Briefings in Bioinformatics*, 7(3):256–274, 2006.

[14] S. Bødker. Scenarios in user-centred design–setting the stage for reflection and action. *Interacting with Computers*, 13(1):61–75, 2000.

[15] S. Bødker, P. Ehn, D. Sjögren, and Y. Sundblad. Co-operative Design - perspectives on 20 years with the Scandinavian IT Design Model. In *Proceedings of NordiCHI 2000*, pages 22–24, 2000.

[16] S. Bødker and K. Grønbæk. Cooperative prototyping. *International Journal of Man-Machine Studies*, 34(3), 1991.

[17] J. Broekstra, C. Fluit, A. Kampman, F. van Harmelen, H. Stuckenschmidt, R.Bhogal, A.Scerri, A. de Waard, and E. van Mulligen. The Drug Ontology Project for Elsevier. In *Proceedings of the WWW'04 workshop on Application Design, Development and Implementation Issues in the Semantic Web*, pages xxx–xxx, New York, May 2004 2004.

[18] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *First International Semantic Web Conference*, 2342:54–68, 2002.

[19] E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, J. Maslen, D. Binns, N. Harte, R. Lopez, R. Apweiler, et al. The Gene Ontology Annotation(GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Research*, 32(90001):262–266, 2004.

[20] S.K. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[21] C. Chen. *Information Visualization: Beyond the Horizon*. Springer, 2004.

[22] A. Cockburn. *Agile Software Development*. Addison-Wesley, 2002.

[23] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[24] R.G. Côté, P. Jones, R. Apweiler, and H. Hermjakob. The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC Bioinformatics*, 2006(7):97, 2006.

[25] W. J. Dai. Impaired macrophage listericidal and cytokine activities are responsible for the rapid death of Listeria monocytogenes-infected IFN-gamma receptor-deficient mice. *The Journal of Immunology*, 158(11):5297–5304, 1997.

[26] T. H. Davenport and L. Prusak. *Working Knowledge: How organisations manage what they know.* Havard Business School Press, Boston, Massachusetts, 1998.

[27] R.D. Dowell, R. M. Jokerst, A. Day, S. R. Eddy, and L. Stein. The Distributed Annotation System. *BMC Bioinformatics*, 2:7, 2001.

[28] R.A. Drysdale, M.A. Crosby, W. Gelbart, K. Campbell, D. Emmert, et al. Flybase: genes and gene models. *Nucleic Acids Research*, 33:390–395, 2005.

[29] S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber. BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21(16):3439–3440, 2005.

[30] A. Elliott. Flamenco image browser: using metadata to improve image search during architectural design. In *CHI '01 extended abstracts on Human factors in computing systems*, pages 69–70, New York, NY, USA, 2001. ACM Press.

[31] T. Etzold and P. Argos. SRS–an indexing and retrieval tool for flat file data libraries. *Bioinformatics*, 9:49–57, 1992.

[32] M. Eyl. The Harmony Information Landscape Interactive, Three-Dimensional Navigation Through an Information Space. Master's thesis, Graz University of Technology, Austria, 1995.

[33] R.T. Fielding and R.N. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[34] C. Fluit, M. Sabou, and F. van Harmelen. *Visualising the Semantic Web*, chapter Chapter 3. Ontology-based Information Visualisation. Springer Verlag, 2002.

[35] M.Y. Galperin. The molecular biology database collection: 2006 update. *Nucleic Acids Research*, 34:D3–D5, 2006.

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

[37] A. Garcia, Y.P.P. Chen, and M.A. Ragan. Information integration in molecular bioscience. *Applied Bioinformatics*, 4(3):157–173, 2005.

[38] Alexander Garcia-Castro, Angela Norena, Andres Betancourt, and Mark A. Ragan. Cognitive support for an argumentative structure during the ontology development process. In *9th International Protégé Conference*, 2006.

[39] J.J. Garrett. Ajax: A new approach to web applications. http://www.adaptivepath.com/publications/essays/archives/000385.php, 2005.

[40] C. Goble and C. Wroe. The montagues and the capulets. *Comparative and Functional Genomics*, 5(8):623–632, 2004.

[41] C.A. Goble, R. Stevens, G. Ng, S. Bechhofer, N.W. Paton, P.G. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(2):532–551, 2001.

[42] C. Gross. *Ajax Patterns and Best Practices*. Apress, 2006.

[43] T.R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Cambridge, MA*, pages 601–602, 1991.

[44] V. Haarslev and R. Moller. RACER system description. *Proceedings of the International Joint Conference on Automated Reasoning*, 2001.

[45] V. Haarslev, R. Möller, and M. Wessel. *RACER User's Guide and Reference Manual Version 1.7.19*, 2004.

[46] S. Haider, R. Holland, D. Smedley, and A. Kaspryzk. Martview, a web interface to biomart, 2007.

[47] I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[48] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic shiq. *Arxiv preprint cs.LO/0005017*, 2000.

[49] T.J.P. Hubbard, B.L. Aken, K.. Beal, B. Ballester, M. Caccamo, Y. Chen, L. Clarke, G. Coates, F. Cunningham, T. Cutts, et al. Ensembl 2007. *Nucleic Acids Research*, 35(Database issue):D610, 2007.

[50] M. Imaz and D. Benyon. How stories capture interactions. In *Proceedings of Human-Computer Interaction-INTERACT*, volume 99, 1999.

[51] I. Jacobson, G. Booch, and J. Rumbaugh. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.

[52] P. Jaiswal, S. Avraham, K. Ilic, E.A. Kellogg, S. McCouch, A. Pujar, L. Reiser, S.Y. Rhee, M.M. Sachs, M. Schaeffer, et al. Plant Ontology (PO): a controlled vocabulary of plant structures and growth stages. *Comparative and Functional Genomics*, 6(7-8):388–397, 2005.

[53] R. Jeffries and H. Desurvire. Usability testing vs. heuristic evaluation: was there a contest? *SIGCHI Bull.*, 24(4):39–41, 1992.

[54] R. C. Jimenez, A. F. Quinn, A. Labarga, K. O'Neill, A. Garcia, and H. Hermjakob. Dasty2, a web client for visualizing protein sequence features. Poster, Joint AFP Biosapiens meeting, ISMB 2007, July 2007.

[55] P. Jones, N. Vinod, T. Down, A. Hackmann, A. Kahari, E. Kretschmann, A. Quinn, D. Wieser, H. Hermjakob, and R. Apweiler. Dasty and UniProt DAS: a perfect pair for protein feature visualization. *Bioinformatics*, 21(14):3198–3199, 2005.

[56] R.L. Jurado, M.M. Farley, E. Pereira, R.C. Harvey, A. Schuchat, J.D. Wenger, and D.S. Stephens. Increased risk of meningitis and bacteremia due to listeria monocytogenes in patients with human immunodeficiency virus infection. *Clinical Infectious Diseases*, 17(2):224–7, 1993.

[57] P.D. Karp, M. Riley, M. Saier, I.T. Paulsen, J. Collado-Vides, S.M. Paley, A. Pellegrini-Toole, C. Bonavides, and S. Gama-Castro. The EcoCyc Database. *Nucleic Acids Research*, 30(1):56–58, 2002.

[58] A. Kasprzyk, D. Keefe, D. Smedley, D. London, W. Spooner, C. Melsopp, M. Hammond, P. Rocca-Serra, T. Cox, and E. Birney. EnsMart: a generic system for fast and flexible access to biological data. *Genome Research*, 14:160–169, 2004.

[59] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methodsa survey. *ACM Comput. Surv.*, 39(4):10, 2007.

[60] G.E. Krasner and S.T. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. Technical report, ParcPlace Systems, Inc, 1988.

[61] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[62] C. Larman. *Applying UML and Patterns*. China Machine Press,, 2004.

[63] C. Larman and VR Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, 2003.

[64] J. Lave and E. Wenger. *Communities of Practice*. Cambridge, 1991.

[65] C. Letondal. *Interaction et programmation - Conception d'applications programmables avec des non-informaticiens*. PhD thesis, Universit de Paris-Sud, 2001.

[66] C. Letondal. A web interface generator for molecular biology programs in unix. *Bioinformatics*, 17(1):73–82, 2001.

[67] C.E. Lipscomb. Medical subject headings (mesh). *Bull Med Libr Assoc*, 88(3):265–266, 2000.

[68] M.J. Muller. PICTIVE: an exploration in participatory design. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 225–231, New York, NY, USA, 1991. ACM Press.

[69] M.J. Muller. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, chapter Participatory Design: The Third Space in HCI, pages 1051–1068. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2002.

[70] M.J. Muller and S. Kuhn. Participatory design. *Communications of the ACM*, 36(6):24–28, 1993.

[71] J. Mylopoulos. An overview of knowledge representation. In *Proceedings of the 1980 workshop on Data abstraction, databases and conceptual modeling*, pages 5–12, New York, NY, USA, 1980. ACM Press.

[72] J. Mylopoulos. Tutorial on artificial intelligence research. In M.L. Brodie and S.N. Zilles, editors, *Proceedings of the Workshop on Data Abstraction, Databases and Conceptual Modelling*, volume 11, pages 13–18. ACM Press, 1980.

[73] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[74] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, New York, NY, USA, 1990. ACM Press.

[75] D.A. Norman. *The Psychology of Everyday Things*. Basic Books New York, 1988.

[76] D.A. Norman and S.W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 1986.

[77] N.F. Noy, R.W. Fergerson, and M.A. Musen. The knowledge model of protege-2000: Combining interoperability and flexibility. *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, pages 17–32, 2000.

[78] Open biomedical ontologies. http://obo.sourceforge.net, 2007.

[79] G.M. Olson and J.S. Olson. Human-computer interaction: Psychological aspects of the human use of computing. *Annual Review of Psychology*, pages 491–517, 2003.

[80] P. Pirolli and S.K. Card. Information foraging. *Psychological Review*, 106(4):643–675, 1999.

[81] Peter Pirolli, Stuart K. Card, and Mija M. Van Der Wege. The effects of information scent on visual search in the hyperbolic tree browser. *ACM Trans. Comput.-Hum. Interact.*, 10(1):20–53, 2003.

[82] C. Plaisant. The challenge of information visualization evaluation. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 109–116, New York, NY, USA, 2004. ACM Press.

[83] C. Plaisant, B. Shneiderman, K. Doan, and T. Bruns. Interface and data architecture for query preview in networked information systems. *ACM Transactions on Information Systems*, 17(3):320–341, 1999.

[84] A. Prada-Delgado, E. Carrasco-Marin, C. Pena-Macarro, E. del Cerro-Vadillo, M. Fresno-Escudero, F. Leyva-Cobian, and C. Alvarez-Dominguez. Inhibition of Rab 5 a exchange activity is a key step for Listeria monocytogenes survival. *Traffic*, 6(3):252–265, 2005.

[85] D.L. Rubin, S.E. Lewis, C.J. Mungall, S. Misra, M. Westerfield, M. Ashburner, I. Sim, C.G. Chute, H. Solbrig, M.A. Storey, et al. The national center for biomedical ontology: Advancing biomedicine through structured organization of scientific knowledge. *OMICS A Journal of Integrative Biology*, 10(2):185–198, 2006.

[86] P. Saraiya, C. North, and K. Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11:443–456, 2005.

[87] W.F. Schlech III. Foodborne listeriosis. *Clinical Infectious Diseases*, 31(3):770–775, 2000.

[88] G.D. Schuler, J.A. Epstein, H. Ohkawa, and J.A. Kans. Entrez: molecular biology database and retrieval system. *Methods in Enzymology*, 266:141–62, 1996.

[89] E.M. Schwarz, I. Antoshechkin, C. Bastiani, T. Bieri, D. Blasiar, P. Canaran, J. Chan, N. Chen, W.J. Chen, P. Davis, et al. Wormbase: better software, richer content. *Nucleic Acids Research*, 2006.

[90] D.B. Searls. Data integration: challenges for drug discovery. *Nature Reviews in Drug Discovery*, 4(1):45–58, 2005.

[91] A. Seffah and E. Metzker. The obstacles and myths of usability and software engineering. *Communications of the ACM*, 47(12):71–76, 2004.

[92] M. Shimoyama, V. Petri, D. Pasko, S. Bromberg, W. Wu, J. Chen, N. Nenasheva, A. Kwitek, S. Twigger, and H. Jacob. Using multiple ontologies to integrate complex biological data. *Comparative and Functional Genomics*, 6(7-8):373–378, 2005.

[93] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A.L. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome Biology*, 6:R46, 2005.

[94] B. Smith, J. Williams, and S. Schulze-Kremer. The ontology of the gene ontology. *Proceedings of the American Medical Informatics Association Annual Symposium*, 2003:609–613, 2003.

[95] C.L. Smith, C.A.W. Goldsmith, and J.T. Eppig. The mammalian phenotype ontology as a tool for annotating, analyzing and comparing phenotypic information. *Genome Biology*, 6(1):R7, 2005.

[96] C.J. Stoeckert and H. Parkinson. The mged ontology: a framework for describing functional genomics experiments. *Comparative and Functional Genomics*, 4(1):127–132, 2003.

[97] H. Stuckenschmidt, A. de Waard, R. Bhogal, C. Fluit, A. Kampman, J. van Buel, E. van Mulligen, J. Broekstra, I. Crowlesmith, F. van Harmelen, et al. A topic-based browser for large online resources. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW04), Lecture Notes in Artificial Intelligence*. Springer, 2004.

[98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1-2):161–197, 1998.

[99] Y. Sure. *Methodology, Tools and Case Studies for Ontology Based Knowledge Management*. PhD thesis, Universitt Karlsruhe, 2003.

[100] M. Tory and T. Moller. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):72–84, 2004.

[101] S. Trissl and U. Leser. Querying ontologies in relational database systems. In *2nd Conference on Data Integration in the Life Sciences (DILS05)*. Springer, 2005.

[102] L. G. Tudor, M. J. Muller, and T. Dayton. A C.A.R.D. game for participatory task analysis and redesign: macroscopic complement to PICTIVE. In *CHI '93: INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, pages 51–52, New York, NY, USA, 1993. ACM Press.

[103] S. Twigger, J. Lu, M. Shimoyama, D. Chen, D. Pasko, H. Long, J. Ginster, C.F. Chen, R. Nigam, A. Kwitek, et al. Rat genome database (rgd): mapping disease onto the genome. *Nucleic Acids Research*, 30(1):125–128, 2002.

[104] S.N. Twigger, M. Shimoyama, S. Bromberg, A.E. Kwitek, and H.J. Jacob. The Rat Genome Database, update 2007–Easing the path from disease to data and back again. *Nucleic Acids Research*, 35(Database issue):D658, 2007.

[105] E. W. Weisstein. Transitive. From MathWorld - A Wolfram Web Resource. http://mathworld.wolfram.com/Transitive.html.

[106] E. Wenger. Communities of practice and social learning systems. *Organization*, 7(2):225, 2000.

[107] R. W. White and G. Marchionini. A study of real-time query expansion effectiveness. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 715–716, New York, NY, USA, 2006. ACM Press.

[108] G. Winters. Use case terminology. *Software, IEEE*, 22(2):67–67, 2005.

[109] L. Wong. Technologies for integrating biological data. *Briefings in Bioinformatics*, 3(4):389–404, 2002.

[110] L. Wood, V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A.L. Hors, G. Nicol, J. Robie, P. Sharpe, B. Smith, J. Sorensen, R. Whitmer, R. Sutor, C. Wilson, et al. Document Object Model (DOM) Level 1 Specification. World Wide Web Consortium Recommendation http://www.w3.org/TR/REC-DOM-Level-1/, 1998.

[111] K.P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *Proceedings of the conference on Human factors in computing systems*, pages 401–408, 2003.

[112] P. Zaphiris, K. Gill, T. Hoi-Yan Ma, S. Wilson, and H. Petrie. Participatory design of information visualisation interfaces for digital libraries. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA)*, 2004.

[113] B.R. Zeeberg, H. Qin, S. Narasimhan, M. Sunshine, H. Cao, D.W. Kane, M. Reimers, R.M. Stephens, D. Bryant, S.K. Burt, et al. High-Throughput GoMiner, an 'industrial-strength' integrative gene ontology tool for interpretation of multiple-microarray experiments, with application to studies of Common Variable Immune Deficiency (CVID). *BMC Bioinformatics*, 6(1):168, 2005.