Development and application of ontologies for biological applications

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades

"Doctor rerum naturalium"

der Georg-August-Universität Göttingen

im Promotionsprogramm Umweltinformatik (PEI)

der Georg-August University School of Science (GAUSS)

vorgelegt von

Jürgen Dönitz

aus Düsseldorf

Göttingen, 2015

Betreuungsausschuss:

Prof. Dr. Edgar Wingender
Institut für Bioinformatik, UMG

Prof. Dr. Stephan Waack
Theoretische Informatik und Algorithmische Methoden, Institut für Informatik

Prüfungskommission:

Referent: Prof. Dr. Gregor Bucher
Abteilung für evolutionäre Entwicklungsgenetik, Johann-Friedrich-Blumenbach-Institut für Zoologie und Anthropologie

Korreferent: Prof. Dr. Edgar Wingender
Institut für Bioinformatik, UMG

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Ernst Wimmer
Abteilung für Entwicklungsbiologie, Johann-Friedrich-Blumenbach-Institut für Zoologie und Anthropologie

Prof. Dr. Burkhardt Morgenstern
Abteilung für Bioinformatik, Institut für Mikrobiologie und Genetik

Prof. Dr. Stephan Waack
Theoretische Informatik und Algorithmische Methoden, Institut für Informatik

Prof. Dr. Ulrich Sax
Institut für Medizinische Informatik, UMG

Tag der mündlichen Prüfung: 27. Januar 2016

Table of content

1. Abstract ... 3
2. Introduction ... 5
   2.1 Background and history of ontologies ... 5
   2.2 Ontology in computer sciences ... 5
   2.3 Relation types in ontologies ... 7
   2.4 Graph representation of ontologies ... 8
   2.5 Tools for ontologies ... 9
   2.6 Ontologies in the field of biology ... 10
   2.7 The red flour beetle *Tribolium castaneum* as model system ... 11
   2.8 The iBeetle project ... 12
   2.9 Aims ... 12
3. Results ... 14
   3.1 The Cytomer ontology ... 16
   3.2 *Tribolium* Ontology TrOn ... 20
   3.3 The ontology-based answers (OBA) service ... 29
   3.4 iBeetle-Base ... 41
   3.5 Other ontology enhanced applications ... 48
      3.5.1 EndoNet ... 48
      3.5.2 TFClass ... 49
      3.5.3 OntoScope ... 51
4. Discussion ... 54
   4.1 Ontologies, the data source ... 54
      4.1.1 The Tribolium Ontology TrOn ... 54
      4.1.2 The Cytomer ontology ... 56
   4.2 OBA service, processing of ontologies ... 57
   4.3 Biological applications enhanced by ontologies ... 59
5. References ... 61
6. Acknowledgements ... 66
7. Curriculum vitae ... 67

1. Abstract

In the century of information technology automatic processing of knowledge is evolving to be a crucial challenge. However, it is usually not obvious to extract information from the syntax of texts. Hence, ontologies are an important pillar by defining entities of a knowledge domain and setting them in relation to each other. The multitude of facts within a field is represented as a collection of single statements following a strict syntax. This structured representation of knowledge allows for automatic processing of the information.

Within the iBeetle project, more than 5,000 genes were knocked down by RNAi in *Tribolium castaneum* in a large-scale screen, and the resulting phenotypes were annotated and compiled in the database of the iBeetle-Base website. I developed the *Tribolium* Ontology (TrOn), a compilation of the morphological structures found in *Tribolium castaneum,* in order to enable consistent annotation and machine-readable description of the observed phenotypes as well as a user-friendly search function. TrOn supports to search for defects in general anatomical structures like “leg”, although the annotation may have pointed to a substructure, e.g. the coxa, a part of the leg. With around 1,000 terms the ontology covers the morphological structures visible from the outside and some selected internal ones. All structures are modeled for the main developmental stages larva, pupa and adult, regardless of their occurrence in the iBeetle phenotype screen. There are three sets of entities within TrOn defining its architecture: concrete, abstract and mixed classes. Concrete ontology classes represent dissectible anatomical structures of a specific developmental stage. Abstract ontology classes stand for umbrella terms, e.g. “wing”, that rather stresses the concept of a biological function and are independent of the developmental stage. Anatomical structures, which are only present in a single developmental stage, and thus represent a dissectible structure as well as a biological concept, are listed in the set of mixed classes.

Initially, ontologies are data stored in a file. Tools are required to process these data, to edit and to visualize them, and also to enable their use as input for statistical calculations. I developed the service Ontology Based Answers (OBA) to offer ontologies and their semantic information to other applications. The OBA service provides access to the ontologies over a network interface. The client of the OBA service uses this

interface to load the connected entities in the background only when required while traversing the ontology. The service is extended by plugins implementing ontology-specific knowledge such as annotation guidelines of the ontology. The OBA service enables an application developer, who is unfamiliar with the semantic of ontologies, to quickly enhance applications similar to iBeetle-Base with the information provided by the ontologies.

The ontology TrOn, the ontology service OBA, and the search function of iBeetle-Base, which I developed in this work, demonstrate the benefit of ontologies for biological applications.

2. Introduction

2.1 Background and history of ontologies

Ontology is a part of philosophy and deals with the nature of being and existence. The term “onto” is derived from the greek ὄν (on) and can be translated with “being”, while “logy” is the common postfix to name a specific science. The term ontology was first used by Rudolf Goclenius (1547-1628) in the *Lexicon Philosophicum* (Goclenius, 1613) to distinguish the philosophy about being from the other areas of metaphysics. Before that, metaphysics comprised all studies beyond the material physics, including theology and philosophical discussion about being and existence. Thoughts about the nature of being were already handed down from Parmenides (520/515-460/455 BCE) (Palmer, 2012) who discussed whether the seed is the same as the tree, where are the differences, and what is the essence of a seed or a tree?

Immanuel Kant (1724-1804) extended the question from what defines the existence of things to how others perceive them. This resulted in observational statements e.g. “the leaf is green”, “the leaf is part of the tree”. With this descriptive component, ontologies evolved into collections of statements the content of a knowledge field, also known as knowledge domain. Simple and concise statements allow an unambiguous description of entities within a certain domain including their relationships. Precise and correct statements reduce the risk for misinterpretation given by personal interpretation and subjective perception e.g. is the leaf part of the tree or is it attached to it?

2.2 Ontology in computer sciences

Evolving from a long tradition in philosophy, ontologies came into the focus of computer sciences where they are used to establish clear, unequivocal representations of things and concepts, as well as their relationship to each other in standardized formats. In a trend-setting article, Tim Berners-Lee introduced a vision about intelligent agents, which should support the human by retrieving and combining relevant information from the WWW (Berners-Lee et al., 2001). In the example of Berners-Lee the intelligent

agents of two siblings arrange a therapy plan for their mother in the best-suited clinic. Therefore, the agents autonomously retrieve information about the therapy, local hospitals and their working hours and combine this information with the calendars of the brother and sister. Such intelligent agents will remain visionary until the information in the web becomes accessible on a machine processable layer and intelligent tools evolve. Ontologies are often used in the background to process information, e.g. by the major search engines to map the user query to the indexed data and to label unambiguous information.

Several systems have been developed to structure ontologies. Common to all systems is that objects of the real world or theoretical concepts are represented by an element of the ontology, here called class. Each ontology class has a unique name or identifier, which is used as representation of the class. The class can be further defined by attributes like label, description, synonyms or specifications of the measurements of the represented object in the real world. Links to other ontology classes can be used to indicate the correlations of the entities in a knowledge domain. In the following a single link between two ontology classes is named relation and relation type the kind of the relation. Depending on the system, different nomenclatures have been established for ontology class, attributes or the links between classes. Most systems, however, use triplets to store information. A commonly accepted concept for the use of triplets to define the content of ontologies is the Resource Description Framework (RDF) (Lassila and Swick, 1999). Briefly, each statement is represented as a triplet composed of subject, predicate and object. The above example of the green leaf is thus equivalent to a triplet compiled of the subject “leaf”, the predicate “has” and the object “color green”. The subject and the predicate of such a triplet in RDF have to be an unequivocally defined entity (resource) that is represented by a Uniform Resource Identifier (URI). While this URI does not necessarily have to be a valid World Wide Web (WWW) address, it serves as unique identifier for the resource. An example would be http://tree-ontology#leaf, where the namespace is written before the hash tag and the name after it. This syntax allows an unambiguous identification and clear distinction between entities possessing the same name within different knowledge domains like the heart in anatomy or in poetry. Objects can be defined by resources or literals, i.e. a specific value like a string, a number or a Boolean value. For the introduced example, the color green can be expressed either by the literal “green” or as a resource representing the RGB color space

with attributes to specify its exact composition in terms of primary color intensities and the resource for the color green as subject.

Adhering the concept of RDF, the predicate (the relation type) is a resource, too. The resource for the relation type is often taken from the namespace of OWL or RDF (e.g *rdf:partOf*) or can be defined within the ontology. Therefore, the links between subject and object are members of a well-defined and limited set. For the color of the leaf, a suitable relation type would be http://example.ontology#hasColor.

While RDF is the system that describes how information about real objects or concepts can be expressed using triplets, further specifications are required to define file formats to store ontologies. The most used triplets-based format to describe ontologies is the Web Ontology Language (OWL) (Dean and Schreiber, 2003), which is based on RDF.

In the field of biology, the OBO file format (Open Biomedical Ontologies) is commonly used (Mungall and Ireland, 2012). It was implemented as default file format of the editor OBO-Edit and is now also supported by several other tools (Day-Richter et al., 2007). Ontologies in the OBO format can be interpreted as single statements expressed in triplets, however, the file format is built up by blocks of lines (stanzas), each collecting the information of one ontology class as key-value pairs (e.g. name: coxa, with the key “name” and the value “coxa”). Compared to OWL, the features of the OBO format are limited to the most important ones. Other differences are the use of an ID instead of a name to identify an ontology class or the naming of ontologies’ components.

2.3 Relation types in ontologies

All ontologies enforce a hierarchy of their entities using the class relation, often named *is_a*. Every ontology class, except of the root class, has one or more parent classes. The architecture of the genius-differentia pattern requires that a child class is a specialization of its parent and has one additional characteristic in order to distinguish it from the more generic parent class (Smith et al., 2007). Thus, the child class inherits all the characteristics of its parent, which are defined by their relations to other objects of the ontology. To further develop the already introduced example, an oak is a tree with all the properties of the abstract concept of a tree and with the differentiation that

it is a member of a specific family of trees. The basic properties for all trees are annotated to the entity "tree", e.g. a tree has branches, leafs and is able to perform photosynthesis. The child class "oak" inherits these characteristics and adds special ones like that it has a specific shape of leafs or has acorns as fruit.

Beside of this obligatory *is_a* relation type, ontologies may contain additional ones, like the often-used *partOf* relation. Although this relation type is widely used in ontologies, in natural language the term "part of" is used in various ways. While the user is able to interpret the part of relation for each instance, for automatic processing each relation type should be restricted to a single and well-defined aspect of "part of". For instance, it can be stated that the leaf is part of the tree, with the meaning that it belongs to it. At the same time, the leaf can also be annotated to be part of the photosynthetic system, where the later one is an immaterial entity. In this case two relations with unequal meanings are named by the same (superior) term. Another case is when *partOf* is strictly used in a spatial way. The statement "*trichome part of leaf*" defines the trichome (the hairs on a leaf) as part of the leaf. This statement combined with the statement "*leaf partOf tree*" concludes that the trichome is part of the tree, because the *partOf* relation type can be specified as transitive in this example. The transitivity is a property of the resource used as relation type. Therefore, all relations in an ontology with this relation type are transitive or not. Relation types should therefore be well considered and documented. A possible solution for the above example would be to render the *partOf* relation type more precisely as *insideOf* and *partOfSystem*.

Actually, the mereology is a subpart of the ontology research that has the characterization of *partOf* relations as its sole topic (Varzi, 2015). The crucial point for every ontology and its use is to be aware of the potential discrepancy between machine processable statements and human interpretations.

2.4 Graph representation of ontologies

By combining a myriad of single statements (expressed as triplets) a knowledge domain can be described in a formalized way. A formal representation of an ontology is a graph

with directed edges without cycles (DAG, directed acyclic graph) (Diestel, 2000). Many algorithms exist for fast information retrieval from graphs, like search algorithms: depth-first search(Tarjan, 1972), breadth-first search (Moore, 1959) or algorithms to find the shortest path between two nodes like the Dijkstra algorithm (Dijkstra, 1959). Therefore, many tools are available for efficient processing of ontologies.

The formalized representation of knowledge and the accessibility for automatic processing make ontologies an important pillar of the semantic web. The amount of available knowledge is increasing rapidly. However, most of the information is available as text, readable by humans but hard to process. If pieces of information are linked to ontologies, the meaning of the data is defined, and also if the specific term is unknown to an automatic system, knowledge about the term can be concluded through its relations to other entities.

2.5 Tools for ontologies

The magnitude of biomedical ontologies is used in various ways. OBOEdit can be downloaded and used locally to edit and visualize any ontology in the OBO file format while AmiGO is an online viewer for the Gene Ontology (Carbon et al., 2009). Other tools are using GO to filter information. GoPubMed (Doms and Schroeder, 2005) filters search results from PubMed, Blast2GO (Conesa and Götz, 2008) applies sequence similarity information to annotate new genes and proteins. In the FlyBase Database the anatomical ontology for Drosophila DAO and other controlled vocabularies are used for consistent annotation and crosslinking of the search results (dos Santos et al., 2015). Examples for tools that are using ontologies for statistical methods like the gene set enrichment analysis (GSEA) (Subramanian et al., 2005) are DAVID (Huang et al., 2007) or WebGestalt (Zhang et al., 2005).

While the above-mentioned tools are using ontologies and provide the user with the computed results, there are only a few options to parse and handle ontologies directly. To process ontologies usually the OWL- or Jena library is used (Apache Home, 2015; Horridge and Bechhofer, 2011). These libraries parse the ontology files and provide access to the input by the means of a Java API (Application Programming Interface). These are generic tools that implement the full specifications of OWL, but are challenging for programmers who are not experts in ontologies or OWL.

An alternative approach to access the content of (biological) ontologies is to use tools specific for the OBO format. First of all, the two major web portals NCBO and OBOFoundry list ontologies from the biomedical field with some additional meta-information to download. In addition, the NCBO portal offers an API to access specific classes of the hosted ontologies, without a prior download or local processing of the whole ontology. A similar service is provided by OntoCAT, which is completed by a corresponding client for R programming languages (Adamusiak et al., 2011; Kurbatova et al., 2011).

These portals and tools aim to give access to a large number of ontologies. They provide search function across all available ontologies and give access to specified ontology classes. However, traversing an ontology on the client side is only possible by several and explicit requests to the server after extracting the URI of the parent or child classes from the current ontology class. Also not supported are queries involving the structure of the ontologies, e.g. to ask whether a class is an ancestor of another one. Such queries would be possible using the SPARQL query language (SPARQL Protocol and RDF Query Language) but require again detailed knowledge about ontologies and of this query language (The W3C SPARQL Working Group, 2013). The network services and Java libraries provide no help, and knowledge about a specific ontology is needed instead. In order to answer the question to which organ(s) a specific cell type belongs the user, or a service, has to know how organs are modeled in the ontology and which relation types can be used during the traversal.

2.6 Ontologies in the field of biology

Ontologies are widely used in the bio-medical field. Under the patronage of the OBOFoundry a multitude of ontologies and tools are available (Smith et al., 2007). The most prominent ontology, and basis for many tools, is the Gene Ontology (GO) (The Gene Ontology Consortium, 2015). Aside from this prime example the anatomical ontologies form a major part of the ontologies hosted by the OBOFoundry or on the NCBO portal (National Center of Biomedical Ontolgoies) (Noy et al., 2009). There are species-specific anatomical ontologies like DAO for *Drosophila* (Costa et al., 2013), ontologies for individual taxa like Coleoptera or even abstract ontologies about anatomical entities, like the CARO ontology (Common Anatomical Reference Ontology)

(Haendel et al., 2008).

2.7 The red flour beetle *Tribolium castaneum* as model system

The red flour beetle belongs to the Coleoptera, the most species rich taxon on earth and is emerging as important model organism for insects (Brown et al., 2009; Klingler, 2004). On the one hand, the additional model organism allows comparing gene function between beetle and fly in order to draw conclusions on evolution. On the other hand, processes and structures can be studied which are not represented in the fly. Examples for morphological structures are the stink glands, which do not occur in the fly (Li et al., 2013). Evolutionary differences between the fly and the beetle are studied with respect to the wings (Clark-Hachtel et al., 2013). The development of *Tribolium* follows, like in most insects, the short-germ embryogenesis while *Drosophila* is known for its long-germ embryogenesis (Schröder et al., 2008). In the beetle, the posterior segments form one after the other in the posterior growth process, while in the members of the long-germ development all segments develop at the same time, the blastoderm stage. These two model organisms do not only differ in the embryonic development, but also the larval development of the appendages is not the same. In *Drosophila* the appendages develop during metamorphosis from early anlagen, whereas in *Tribolium* the appendages form during embryogenesis and do not undergo major rebuilding. In many aspects *Tribolium* is more representative for insects than the fruit fly, hence the beetle is a valuable complement.

However, for the emerging model organism *Tribolium* an anatomical ontology was missing. The standard reference for the anatomy of the red flour beetle has been the book “The Biology of Tribolium” from Sokoloff (Sokoloff, 1972). While this work has been the authoritative source, the book is getting outdated with new research activity missing. The need of a current and authoritative definition and naming of morphological structures was highlighted by Yoder and colleagues (Yoder et al., 2010). From various publications they extracted which region the authors denominated as ‘‘paramere’’. The identified structures differed significantly, in some cases they overlapped, in some they even contradicted each other. In order to standardize knowledge and definitions, an electronic resource is preferable because it can be updated easily, it is more easily accessible than a book and it is machine-readable.

2.8 The iBeetle project

Tribolium and the major model organism *Drosophila* do not only differ in their morphology and developmental process; also the available genetic tools are not the same. The fruit fly has a long tradition in forward genetics where phenotypes are induced by mutagenesis and thereafter the affected gene is identified. Many important genes and developmental mechanisms could already be uncovered by this procedure. In *Tribolium* reverse genetics using RNA interference (RNAi) is a well-established method (Brown et al., 1999). Single genes can be silenced using short double stranded RNA (dsRNA), which is complementary to a part of the transcript of the target gene. After injection of dsRNA in the beetle the gene is knocked down ubiquitously. This effect can even be transferred to the offspring generation (Bucher et al., 2002).

However, reverse genetics always start with the gene to be knocked down. Therefore, it is unlikely to identify unexpected gene functions. The iBeetle project is a large-scale RNAi screen in *Tribolium castaneum* to discover novel gene functions in an unbiased way (Schmitt-Engel et al., 2015). dsRNAs for randomly selected genes were injected into *Tribolium*. In the first phase of the project, knockdown was performed by dsRNA injected into larvae and pupae, in the second phase in pupae only. Following a specific protocol a set of possible phenotypes was scanned for and the observed phenotypes were annotated. The annotation of the phenotype adhered to an entity-qualifier-modifier system (EQM) (Mungall et al., 2010). For the morphological structure, the entity, the Tribolium Ontology was used, and controlled vocabularies (CV) for the other fields. This combination of fields and CVs led to readable annotations, e.g. “abdomen number decreased”, and at the same time was suitable for automatic processing. The iBeetle screen was the first large scale RNAi screen in an insect outside *Drosophila*.

2.9 Aims

The amount of biological and medical knowledge is constantly and rapidly increasing. Filtering, preparation and representation of data have to be improved in order to make relevant information easily accessible to the scientific community. Ontologies represent an essential building block for information processing systems and enable automatic processing of the information of knowledge domains by defining their entities and relations in a structured way.

In the context of this thesis two ontologies should be created or substantially extended. For *Tribolium castaneum* an anatomical ontology was missing and should be created with a design suitable to represent the development of morphological structures in the developmental stages and at the same time to provide interconnection points to other insect ontologies. For the human anatomy with Cytomer a hierarchical structure of anatomical structures existed in a relational database. After the migration to the OWL format, the Cytomer ontology consists only of two types of anatomical structures, organs and cells. The aim during this thesis was to define a new architecture with a more detailed hierarchy, also along newly defined relation types. The annotation, done by medical students, should be supervised to achieve the goal.

To make ontologies available to applications a service should be implemented that provides the content of ontologies through a network interface. In addition this service should be able to answer predefined queries with the help of specific ontologies, which assumes detailed knowledge about the semantic of these ontologies.

For the results of the iBeetle project a public web interface should be implemented. A user-friendly search function should use the Tribolium Ontology through the OBA service, in order to bridge the different level of detail used for annotation and for the search. The advantage of ontologies for biological applications should also be demonstrated by means of EndoNet, an information resource about the human intercellular network, TFClass, a classification of transcription factors, and OntoScope, a graphical ontology viewer.

3. Results

This thesis consists of three major parts as they are summarized in Figure 1. Ontologies are the information sources, which serve as input for other applications. Examples for such input source in this thesis are the Tribolium Ontology TrOn and Cytomer for the human anatomy. However, the tools downstream in the processing pipeline are not specific for these two ontologies but can use also alternative ones. The OBA service accesses ontologies and provides their content and derived information to applications for the user. The service can be used via a network interface or the communication can be encapsulated by embedding the provided Java client. The biological applications, in the figure on the right side, can benefit from the ontologies. The OBA service unburden the application developer from being an ontology expert by providing the content and semantics of ontologies in a easy to use way, achieving a separation of concerns. In this thesis this is demonstrated by three web interfaces (iBeetle-Base, EndoNet, TFClass) and a Java application (Ontoscope).

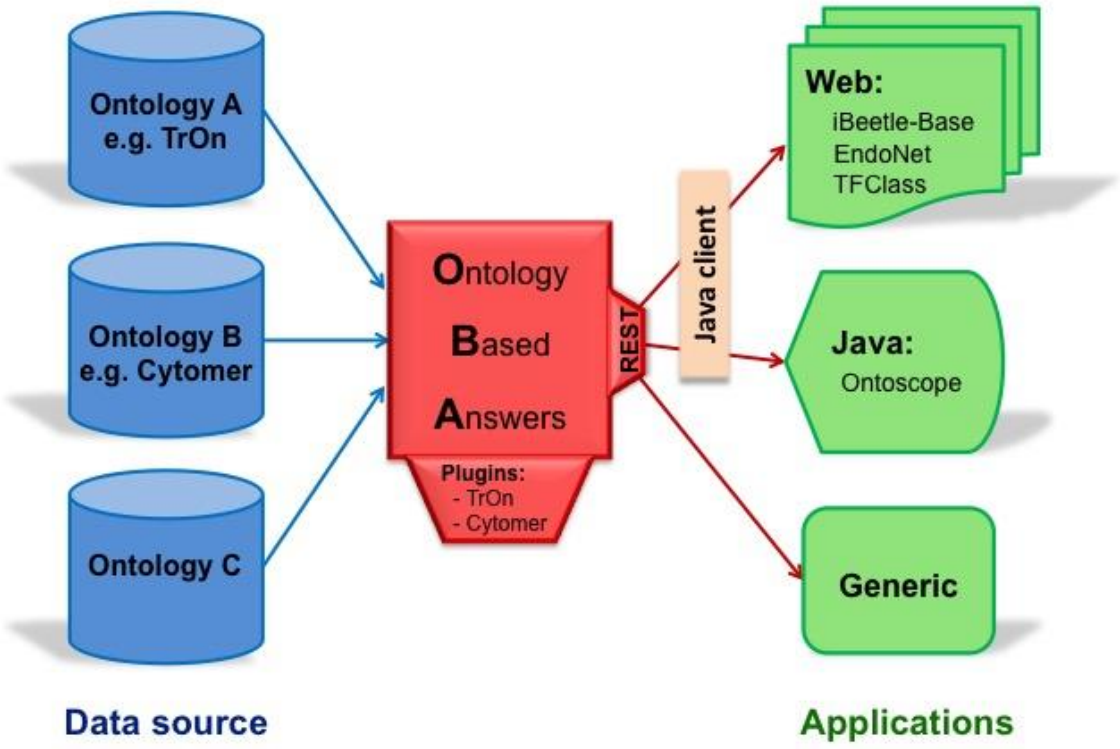

Figure 1

The overall structure of the thesis:

On the left are ontologies listed as input sources for the processing by the OBA service in the middle. The OBA service is used for the two ontologies, which are part of the thesis, but can handle also other ontologies. OBA offers the content of loaded ontologies over a RESTful (Representational State Transfer) to other applications. The generic functions can be complemented by project specific plugins, which implement knowledge about the semantic and annotation policies of specific ontologies. Biological applications can query the network interface of OBA or embed the provided Java client. The Java client provides the functions of the OBA service as Java functions and loads the entities from the server as required while the application is traversing the ontology.

3.1 The Cytomer ontology

Cytomer is an ontology about the human anatomy ranging from the whole body down to the cell level including embryonic structures of the Carnegie stages (O'Rahilly and Müller, 1987; Streeter and Corner, 2013). It comprises about 10,000 ontology classes (Michael et al., 2005). The template for Cytomer is the ideal human body without aberration in anatomy caused by illness. Organs specific for only one sex are modeled as descendant of the nodes female or male genitalia. Differences between the genders or during growth, which are only based on the mass or size, are not further considered. Dimension and absolute position, like left - right or dorsal - ventral are not expressed with relations but noted in the comment where appropriate. Logical spatial relations like *containedIn* or *adjacentTo* can be modeled, if there is a functional relation between the two entities, e.g. to express that the heart is inside of the pericardium. While the growth in size and mass of anatomical structures is not modeled, their development is. The primitive heart tube as well as the heart are modeled and linked by a *differentiatesInto* relation. Structures only available in the embryo are collected below of *embryonic_structure*. Multiple inheritances are used in Cytomer to represent various aspects of anatomy. A muscle may have several parents *flexor_muscle* to define the function, *muscle_of_arm* for its location and *two_headed_muscle* for its anatomical structure. Table 1 gives an overview how often multiple inheritance occurs in Cytomer and how many parents are involved. This equal treatment of the various characteristics is neutral in contrast to use a single parent approach that models other relations as *partOf*.

Table 1: Occurrence of multiple parents in Cytomer classes.

Number of parent classes for a single ontology class	Number of ontology classes
1	9,635
2	1,112
3	80
4	26
5	10

In Cytomer each ontology class has one or more parent classes. In the first column the number of parents is given, in the second the number of ontology classes with this number of parent classes.

The more than 21,000 relations between the classes are modeled with a set of over 30 relation types to describe the gross anatomy (*partOf*, *isCellOf*, *adjacentTo*), anatomical functions (*innervates*, *attachedToOrigin*), physiological systems (*partOf*, *hasPart*), and the development (*differentiatesInto*). Table 2 lists all relation types used in Cytomer together with the number of occurrence and where appropriate their inverse relation type and parent.

With this high numbers of ontology classes, relation types, and relations Cytomer is a complex information resource. The semantic information encoded in the ontology is accessible with the OBA service and complemented by a Cytomer plugin for OBA. Two semantic functions are provided by this plugin. The first is used to query for the organs an arbitrary anatomical structure belongs to, therefor, OBA performs a breath-first search along a specific set of relation types until an organ is found. Already in the case of "hepatocyte" 70 ontology classes have to be visited to get to the organ "liver". For the organs of "sensory_epitelial_cell" 2,497 classes have to be checked to find the organs "lung", "larynx" and "trachea". The second function limits the up- or downstream search to a predefined set. In Cytomer far more structures are listed than are used in a typical application. An unfiltered search for super- or substructures in such an application would reveal many hits where the application has no information about. With this implemented function the result of an up- or downstream search are restricted to those

found in the predefined set before the result is transmitted to the application.

Table 2. Relation types of Cytomer

Relation type	Number of occurrence	Inverse relation type	Parent relation type
receives	9	leadsInto	
bordered_by	13	borders	adjacentTo
borders	15	bordered_by	adjacentTo
leadsInto	16	receives	
containedIn	24	contains	
contains	44	containedIn	
accompanies	48		
developsInto	70	derivesFrom	
becomes	71		
attachedTo	77		
isFedBy	78	drainsInto	
locationOf	81	locatedIn	
forms	81	formedBy	adjacentTo
innervates	93	isInnvervatedBy	
formedBy	97	forms	adjacentTo
isSuppliedBy	122	supplies	
drainsInto	131	isFedBy	
derivesFrom	157	developsInto	
isInnervatedBy	173	innervates	
locatedIn	173	locationOf	
differentiatesInto	183	differentiatesFrom	
supplies	185	isSuppliedBy	
attachedToApproach	198		attachedTo
attachedToOrigin	207		attachedTo
isCellOf	271	hasCell	isPartOf
differentiatedFrom	284	differentiatesInto	

connectWith	301		
adjacentTo	343		
isOriginOf	365	hasOriginIn	
hasOriginIn	879	hasOriginOf	
hasPart	2211	isPartOf	
hasCell	4127	isCellOf	hasPart
isPartOf	8254	hasPart	

The table lists the relation types, which are used in Cytomer sorted by the number of occurrence. The number and the following columns refer to the relation type stated in the first column. If the relation type is defined as the inverse of another one, the inverse relation type is noted. The last column denominates the parent relation type the relation type inherits in case of a child relation type.

Cytomer is available online for free at http://cytomer.bioinf.med.uni-goettingen.de

Contributions:

Cytomer was initially developed by the company BIOBASE GmbH as relational database with tables for organs, cells, developmental stages and physiological systems. The organ table contained also links to the other tables as well as to entries of itself to build a hierarchical structure (Matys et al., 2003). After the transfer to the Department of Bioinformatics at UMG, a first conversion into the OWL format was performed (Michael et al., 2005). After this conversion I improved the performance of the ontology so that a further annotation was possible. At the beginning in cooperation with Holger Michael, I developed the architecture of Cytomer as described in the results and supervised the annotation of the ontology, which was done by medical students.

3.2 *Tribolium* Ontology TrOn

The following manuscript is published as:

Dönitz J, Grossmann D, Schild I, Schmitt-Engel C, Bradler S, Prpic NM and Bucher G.: **TrOn: an anatomical ontology for the beetle Tribolium castaneum.** *PLoS One.* 2013 Jul 30;8(7):e70695.

Authorships:

- Jürgen Dönitz initiated the project, developed the semantic concept, discussed questions according to the annotation and wrote most of the manuscript
- Daniela Grossmann annotated the ontology, did the biological research and helped writing the manuscript.
- Christian Schmitt-Engel supported the project, particularly with regard to the connection to iBeetle-Base, and helped during discussions about the morphology of *Tribolium*.
- Sven Bradler checked the morphological correctness of the ontology.
- Nikola- Michael Prpic contributed the drawings of the morphology of *Tribolium*.
- Gregor Bucher supervised the biological annotation of the ontology and supported the writing of the manuscript, particularly by contributing the introduction.

OPEN ACCESS Freely available online

PLOS ONE

TrOn: An Anatomical Ontology for the Beetle *Tribolium castaneum*

Jürgen Dönitz[1,2*☯], Daniela Grossmann[1☯], Inga Schild[1], Christian Schmitt-Engel[1], Sven Bradler[1], Nikola-Michael Prpic[1], Gregor Bucher[1]

1 Johann-Friedrich-Blumenbach Institute of Zoology and Anthropology, Göttingen, Germany, 2 Department of Bioinformatics, University Medical Center Göttingen, Göttingen, Germany

Abstract

In a morphological ontology the expert's knowledge is represented in terms, which describe morphological structures and how these structures relate to each other. With the assistance of ontologies this expert knowledge is made processable by machines, through a formal and standardized representation of terms and their relations to each other. The red flour beetle *Tribolium castaneum*, a representative of the most species rich animal taxon on earth (the Coleoptera), is an emerging model organism for development, evolution, physiology, and pest control. In order to foster *Tribolium* research, we have initiated the *Tribolium* Ontology (TrOn), which describes the morphology of the red flour beetle. The content of this ontology comprises so far most external morphological structures as well as some internal ones. All modeled structures are consistently annotated for the developmental stages larva, pupa and adult. In TrOn all terms are grouped into three categories: Generic terms represent morphological structures, which are independent of a developmental stage. In contrast, downstream of such terms are concrete terms which stand for a dissectible structure of a beetle at a specific life stage. Finally, there are mixed terms describing structures that are only found at one developmental stage. These terms combine the characteristics of generic and concrete terms with features of both. These annotation principles take into account the changing morphology of the beetle during development and provide generic terms to be used in applications or for cross linking with other ontologies and data resources. We use the ontology for implementing an intuitive search function at the electronic iBeetle-Base, which stores morphological defects found in a genome wide RNA interference (RNAi) screen. The ontology is available for download at http://ibeetle-base.uni-goettingen.de.

Citation: Dönitz J, Grossmann D, Schild I, Schmitt-Engel C, Bradler S, et al. (2013) TrOn: An Anatomical Ontology for the Beetle *Tribolium castaneum*. PLoS ONE 8(7): e70695. doi:10.1371/journal.pone.0070695

Editor: Alistair P. McGregor, Oxford Brookes University, United Kingdom

Received April 3, 2013; **Accepted** June 21, 2013; **Published** July 30, 2013

Funding: The authors acknowledge funding by the DFG research unit FOR1234 iBeetle (http://www.dfg.de/) and the support by the German Research Foundation and the Open Access Publication Funds of the Göttingen University. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: juergen.doenitz@bioinf.med.uni-goettingen.de

☯ These authors contributed equally to this work.

Introduction

Gene knock-down by RNA interference (RNAi) [1] has made it possible to study gene function in many different arthropod model systems. Large scale screens are under way and are used to determine the function of thousands of genes. Due to the ease of culture and its amenability to forward and reverse genetic methods, the red flour beetle *Tribolium castaneum*, a representative of the most species rich taxon on earth, the Coleoptera, has been developed into an insect model system second only to *Drosophila*. It is used for research of different topics such as evolution and development of trunk, head and brain, physiology and pest control [2–9]. In the ongoing large scale RNA interference (RNAi) screen "iBeetle", genes have been systematically knocked down in the red flour beetle *Tribolium castaneum*. Upon injection of dsRNA into larvae and pupae, the respective gene function is knocked down and the resulting morphological phenotypes have been documented in an electronic database using a defined vocabulary. The focus has been on developmental defects during embryogenesis and metamorphosis, on muscle development and on oogenesis. About 5.000 genes have been screened so far (Bucher unpublished). In such projects phenotypic data for thousands of genes is generated which requires a controlled vocabulary describing the wildtype and phenotypic morphology.

In textual form the morphology of *Tribolium* is described in large parts in the text book "The biology of *Tribolium*" from Sokoloff [10]. This work is accepted as reference for the *Tribolium* anatomy. However, being a printed medium, the valuable knowledge cannot be included in automatic data processing or online databases. Some details with respect to morphological descriptions are not covered by Sokoloff but were named when the need arose and the respective information is distributed in various publications. Examples are aspects of central nervous system morphology [8], a set of setae and bristles present on walking legs of first instar larvae [11] and a set of setae and bristles marking the dorsal part of the *Tribolium* head [12,13].

Additional structures have been named in order to allow annotation in the iBeetle screen, like for instance the anterior angle of the pronotum (Klingler, Bucher unpublished). An overview about the morphology of *Tribolium castaneum* at larval, pupal and

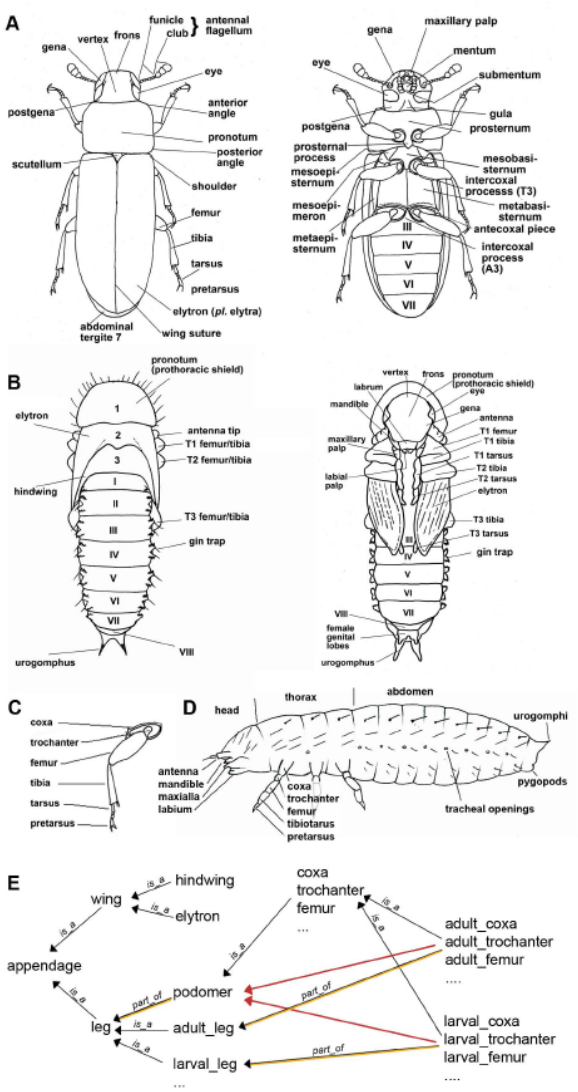

Figure 1. Morphology of the red flour beetle *Tribolium castaneum*. Morphological terms as represented in TrOn are given for three life stages: adult A), pupa B) and first instar larva D). The podomers of the adult walking legs are shown in C). E) Example for the relations `is_a` and `part_of` between terms of the ontology. Note that for simplicity not all terms and not all connections are shown. There are several types of *appendages*, for instance *legs* and *wings*. There are two types of *wings* (*elytron* and *hindwing*). The *legs* are present at all life stages (e.g. `adult_leg` *and* `larval_leg`). Legs are composed of several *podomeres* (see panel C) and each podomer is `part_of` a *leg*. Each life stage has legs which contain podomeres. Hence, the `adult_trochanter` has several relations: it `is_a` *trochanter* and it `is_a` podomer and it is `part_of` the `adult_leg`. In an ontology, `is_a` relations can be used to infer indirect connections. For instance, the adult_trochanter `is_a` trochanter. In addition, the trochanter is defined as a podomer. As consequence, also the `adult_trochanter` is regarded as a podomer although this relation has not been defined directly in the ontology (see red arrow). When implemented in databases, such an ontology can help to make searches more intuitive. A search for "*appendage*" would reveal datasets, with *wing* or *larval_trochanter* phenotypes. A search for datasets affecting *podomer* or *leg* would not consider datasets with annotated wings.
doi:10.1371/journal.pone.0070695.g001

adult life stages is given in Figure 1 together with labels for important anatomical structures.

Due to the distribution of this information in several publications, *Tribolium* research can at the moment not build on a central, searchable and authoritative repository. As the *Tribolium* model system is gaining popularity it is the right moment to initiate such a resource before different terms and definitions of the same structure get established in parallel like the example of the paramere of the hymenoptera as shown by Yoder at al. [14]. During their efforts to build an ontology which covers the lineage of the Hymenoptera (HAO) they faced the difficulty, that the term paramere was used quite different in the available publications. Not only a smaller or larger set of structures of the male genitalia were labeled as paramere, but also the publications which restrict the term to a small subset of structures did not always agree on the same structures. Moreover, even if the morphological knowledge is mapped in a one-to-one correspondence to terms and their definitions, still the relations between them would not be accessible for automatic processing.

The state of the art method to collect and organize a complex resource like that is the use of ontologies. Ontologies are used to represent an expert's knowledge in a formal and standardized way. Each item or concept of the real world is represented in the ontology by one *term*. An ontology term identifies unambiguously one item and describes it. Therefore, a term has a name and may have a list of synonyms, a definition or other properties which help to describe the term, like references to other resources. Furthermore an ontology does not only define a collection of terms, but also a set of *relation types*, to describe the connections between the terms. A highly simplified depiction of an ontology of terms related to walking legs is shown in Fig. 1E (see legend for further details).

The most important connection is the `is_a` relation type. Except for the *root term*, each term has one or more *parents* and may have several *child* terms. Following the concept of *inheritance*, a child term is a specialization of its parents distinguished by a differentiation specific for the child (*genus-differentia* [15]). For example, the antenna is a child of the parent term `appendage` and is connected to its parent by an `is_a` relation. Hence, it inherits the characteristics of the parent (i.e. that it is a moveable outgrowth like other appendages) but it is special in that the antenna is located on the dorsal head and has functions different from other appendages. Besides of this term hierarchy, an ontology can define other relation types, most common is `part_of`. The `part_of` relation type facilitates the representation of logical interconnection between morphological terms, which are clear to the researchers but are not covered by the term hierarchy. For instance, the `antenna` can be linked to the `head` with a `part_of` relation while it is connected to appendage with the `is_a` relation. A child term is always a specialization of its parents and inherits their relations. E.g. `larval_antenna` is a child of `larval_appendage` and inherits the `part_of` relation from `larval_appendage` to larva. While this inheritance of relations is always defined along the axis of the class hierarchy it is not mandatory for the other relation types of an ontology. If such a relation type is not explicitly defined as transitive the term's relations are not inherited along the non-`is_a` relations. This definition is made at the level of the relation type and is obligatory for all relations of this kind. To reuse the example of the antenna, the flagellum has a `part_of` relation to the term antenna and a `is_a` connection to its parent term `multicellular_tissue`. If the relation type in this case would be defined to be transitive, `flagellum` would be part of the head due to two `part_of` relations: The one between `flagellum` and antenna and between `antenna` and `head`. Otherwise the assertion that the flagellum is a part of the head cannot be deduced.

Due to their controlled vocabulary and the clearly defined relations between terms, ontologies are useful to analyze large scale data or can be used to correlate data of different sources. For instance, the RNAi phenotypes of the ongoing iBeetle project and the mutant phenotypes stored in the Göttingen-Erlangen-Kansas-US Department for Agriculture (USDA) (GEKU) database [16] could be correlated. In addition, the correlation can be extended to the cross-species level [17]. *Tribolium* will be the first insect outside *Drosophila* with genome wide functional genetic data, which opens the possibility to do a comprehensive comparison of phenotypes between these two species.

In the biomedical field, ontologies are well established. The Gene Ontology (GO) [18] is the most important ontology in biomedical research, tagging genes with functional annotations and it is integral part of various kinds of automated data analyses and cross linking of data. On ontology portals like the OBO Foundry [12] or the NCBO Bioportal [19] anatomical ontologies form a major category. For the phylum of arthropods there are ontologies describing the morphology of a restricted subset of species like the Hymenopteran Anatomical Ontology HAO [14] or covers an extensive clade like the Arthropod Ontology AO. On the other hand, there are ontologies dealing with one species like the ontology FBbt [20] of the fruit fly *Drosophila melanogaster*. The latter is widely used for instance for the annotation of embryonic expression patterns at the Berkeley *Drosophila* Genome Project (BDGP) expression database and the annotation of neuroanatomy and protein expression of the brain [21–23].

In the scope of the iBeetle project, we have developed the *Tribolium* Ontology (TrOn) describing the morphology of *Tribolium castaneum* as first and only repository for the anatomical structures of the red flour beetle.

The role of TrOn in the iBeetle project is to support the screening procedure and most importantly to allow a semantic search function on the public web interface. The ontology makes it possible to annotate the most specific affected structure and at the same time allows searches for abstract terms comprising several concrete structures. Without an ontology, a user interested in searching for all walking leg phenotypes would have to combine searches for all its substructures with an OR-search. However, the TrOn ontology contains this information in form of the `part_of` relations, such that the user just needs to search for walking leg in order to find all phenotypes related to this structure and all it substructures.

Unfortunately, it was not reasonable to use one of the existing morphological ontologies and adapt it to *Tribolium* in a simple way. The fly ontology is very comprehensive and much of the adult beetle morphology can be aligned to *Drosophila* counterparts. However, there are also many differences, which are too significant to simply transfer the morphological features from one species to the other. For instance, the *Drosophila* leg is defined in the fly ontology as arising from imaginal discs during metamorphosis. While being correct for Dipterans, this is not true for *Tribolium* and most other insects, which do have legs already at the larval stage from which the adult legs develop. In contrast to the FBbt, the AO as well as the HAO are designed to comprise several species, but they lack most terms required to unambiguously identify specific morphological structures in *Tribolium*.

In this work we present TrOn, where most anatomical structures visible from the outside are annotated, defined and interconnected with `part_of` and `is_a` relations. With the help of the ontology based answers service (OBA service) [24] TrOn is already actively used in the public search interface of the iBeetle project. On the web pages of the iBeetle-Base the ontology can be browsed, searched and downloaded.

Materials and Methods

The *Tribolium* anatomical ontology (TrOn) deals with the anatomical structures of the red flour beetle *Tribolium castaneum* at the developmental stages larva, pupa and adult. Where appropriate, the term hierarchy and definitions were taken from the *Drosophila* ontology (FBbt) or the Common Anatomy Reference Ontology (CARO) ontology [25]. The definitions of *Tribolium* specific structures are based on the text book from Sokoloff, respective publications [8,11,12] and the Handbook of Zoology [26]. The ontology is available in the OBO format [15] from the iBeetle-Base web page and the NCBO BioPortal.

The terms, relations and definitions where annotated with the ontology editor OBO-Edit in Version 2.3 [27]. For the OBA service, a plugin was developed to identify the concrete, mixed and generic terms and to provide search functions based on these sets. In the ontology, the three categories are represented as subsets which are retrieved from the OBA service.

For the ontology viewer OntoScope [28] a plugin was developed to visualize TrOn which uses color codes to display the subset of a term and the developmental stage the term belongs to. OntoScope is used for the Figures 2 and 3 and is also available as Java webstart program from the iBeetle-Base web page.

Results

Defined Morphological Structures

The knowledge domain of the newly created *Tribolium* Ontology (TrOn) is the morphology of the red flour beetle *Tribolium castaneum* at the developmental stages larva, pupa and adult. The ontology comprises most morphological structures visible from the outside and some internal structures that are being screened during the iBeetle project. These structures are most of the somatic muscles, the stink glands and the ovary. Apart from stage specific structures, all entities are modeled for all appropriate life stages in the ontology. Substructures and connected structures are considered in all modeled life stages. Symmetrical structures are annotated once, not distinguishing between the left and right side. For symmetric and other multiple structures the singular form is used as name.

Represented Life Stages

Besides of single structures and groups of them, TrOn contains ontology terms representing the beetle organism at each life stage. The anatomical structures specific for a developmental stage are linked with `part_of` relations to their stage, such that the ontology term representing a developmental stage is a collection of anatomical structures. The development of the beetle can be clearly divided into distinct stages separated by molts. These stages share common anatomical structures like legs or antenna. The anatomy of these stages, despite a common function, can be quite dissimilar at different stages. Consequently, we created for each structure in each developmental stage separate ontology terms. The terms are named in the same way for each stage (e.g. `larva_leg`, `adult_leg`). This holds true for concrete morphological structures like `pupa_femur` as well as for more abstract terms like `larva_appendage`.

The seven larval stages are quite similar to each other while pupal and adult morphology differs greatly. Therefore, the structures were not modeled separately for each larval stage but all structures are linked to the main term `larva` from where the

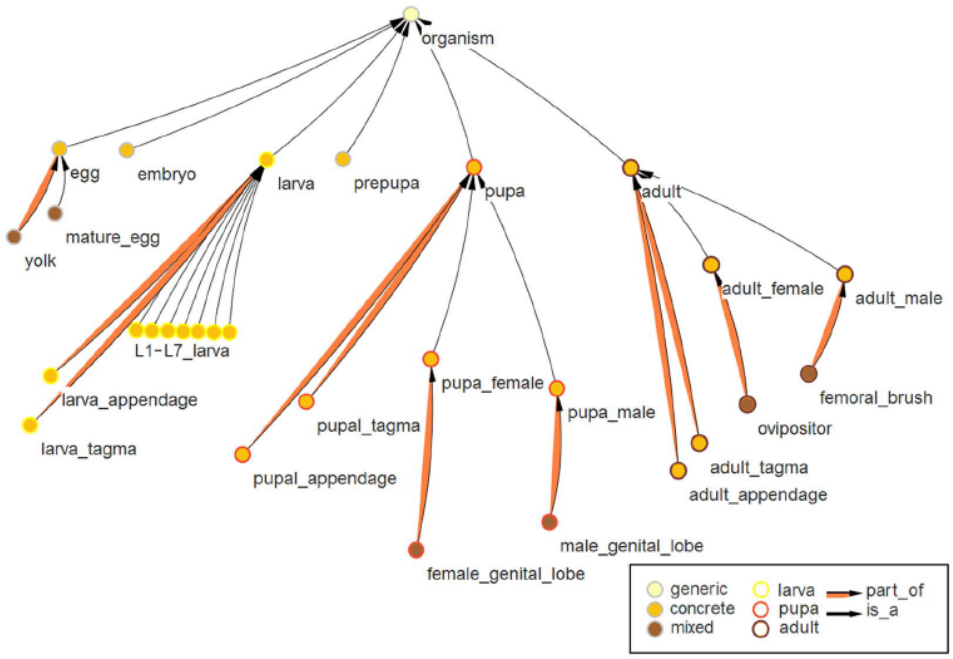

Figure 2. Representation of developmental stages in TrOn. This figure, exported from the ontology viewer OntoScope, displays the terms downstream of organism. `These terms` represent the red flour beetle at a specific developmental stage, which in nature are separated by molting (e.g. larva, pupa, adult). Transition stages, during which dramatic changes of morphology occur (i.e. prepupa and embryo) are included in TrOn to cover all developmental stages but were not linked with morphological structures. The `is_a` relations connect the terms with its parents (see thin black arrows). The anatomical structures are linked with `part_of` relations to the life stages (see orange arrows). The larval stages are divided into sub stages (L1–L7), which inherit the `part_of` relations from the parent term `larva`. Hence, the morphological structures are defined only once for the larval stage. The color of the center of the nodes represent the subset of the ontology the node belongs to, i.e. generic, concrete or mixed class. The border of the node indicates the developmental stage of the represented term.
doi:10.1371/journal.pone.0070695.g002

relations are inherited by the subdivisions L1 to L7. These larval stages are specializations of the more abstract larva stage and therefore subterms of it. In addition, there are intermediate developmental stages, where dramatic changes in morphology occur (embryogenesis in the egg and metamorphosis during the pre-pupal stage). These stages are represented in the ontology but do not have morphological structures assigned to them because the respective dynamic processes are difficult to define using the static anatomical definitions of an anatomical ontology. The developmental stages are children of the ontology term `organism`, which by itself is a specialization of `multicellular_structure` (Figure 2).

The Part_of Relation Type

In addition to the mandatory `is_a` links, the relation type `part_of` is used in TrOn. This relation type is not used in a strictly spatial sense, but also in a functional one, and is not defined as transitive in our ontology [29]. In most cases, the subterm is also a spatial subpart of the larger structure like the `femur` is a spatial part of the `leg`. On the other hand, the `antenna` is `part_of` the `head` but it is discussable if the antenna is inside the space of the head or just appended to the head and functionally related to it. A head without an antenna is still noticed as a head. The `flagellum` is part of the `antenna` but does not inherit the functional `part_of` relation of `antenna` to `head`, because the `part_of` relation type is not defined to be transitive in this case.

Introducing Term Categorization Depending on their Relation to Life Stages

As a feature in TrOn, we introduce the categorization of all terms into concrete, generic and mixed categories (subsets), depending on their relation to a developmental stage. *Concrete terms* are linked to a specific developmental stage and represent anatomical structures that can be, at least theoretically, dissected from a beetle (e.g. the `adult_femur`). Upstream of the concrete ontology terms are the *generic terms*. These are independent of a developmental stage and represent the functional concept of a morphological structure (e.g. the `femur`), which is found at several stages or in several structures of one stage (e.g. `sense_organ`, which is found in many copies on the cuticle). Hence, generic terms usually comprise several, dissectible structures, i.e. they are umbrella terms. The third category subsumes terms of morphologically concrete structures, which are found only at one developmental stage, like the `oocyte` (female adult), the `gin-trap` (pupa) or the `femoral_brush` (male adult). These terms are categorized as *mixed terms* because they combine the characteristics of the generic and concrete category in a single term. Mixed terms are a subset of the concrete terms, because they are dissectible structures of a certain developmental stage. The naming of the mixed terms (i.e. concrete structures, which occur only at one stage) differs in that the stage is not part of the name.

The link of a term to a certain life stage can be direct or indirect. The `part_of` relation to a stage can be inherited through the `is_a` relation from the parents. Also the `part_of` relation type is used in a transitive way in the case of the assignment of a

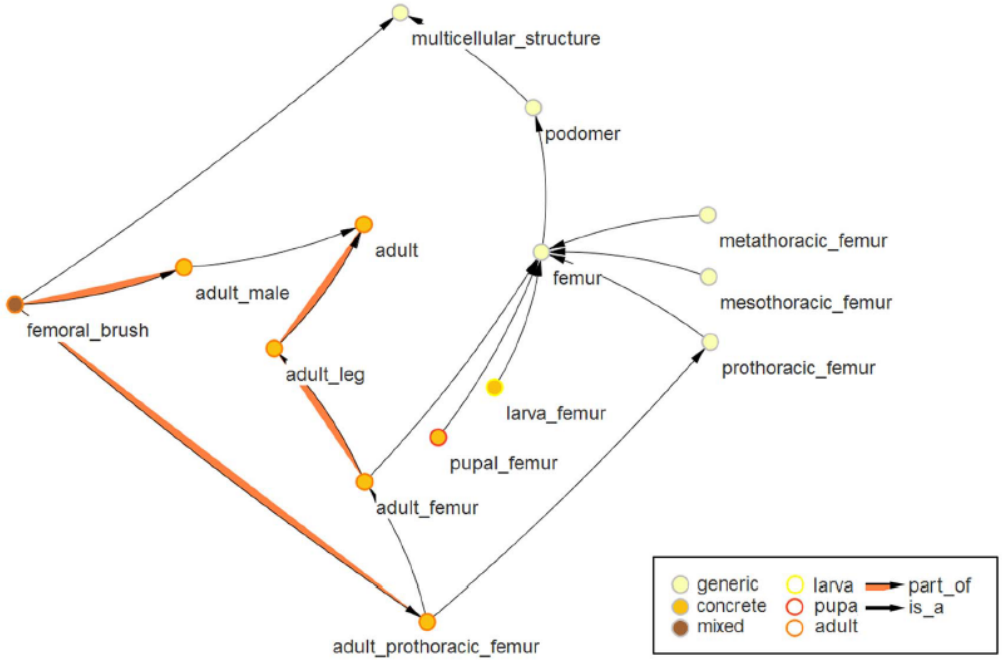

Figure 3. Generic and concrete subsets of terms. Selected terms and relations with respect to the femur are shown. The femur is the third leg segment (or podomer) of insects. It is present at all life stages, but looks quite different between e.g. larval and adult stages. Hence, `femur` is defined to be a generic term, i.e. an umbrella term, which defines morphological concepts, that are not linked to a certain stage or species. The concrete terms in contrast define dissectible structures of a certain life stage of *Tribolium*, which can be very different between life stages or insects species. The center's colors of the nodes represent the subset of the ontology the node belongs to, i.e. generic, concrete or mixed class. The border of the node indicates the developmental stage of the represented term.
doi:10.1371/journal.pone.0070695.g003

concrete term to a developmental stage. For instance, the adult femur is categorized as adult because of the two `part_of` relations between `adult_femur` and `adult_leg` and between `adult_leg` and `adult`. The categorization and the respective links and naming principles are shown for the term femur in Figure 3. The colors of the nodes represent the category they belong to. The generic term `femur` describes the third leg segment (or podomere) of any insect leg. It has children deduced by one of the following specializations. The first differentia is the leg to which the femur belongs, leading to the three children `prothoracic_femur`, `mesothoracic_femur` and `metathoracic_femur`. A distinction between left and right legs is not made because the corresponding structures are symmetrically equivalent. The second differentia added to the abstract term `femur` is the developmental stage. In TrOn this results in `larva_femur`, `pupal_femur` and `adult_femur`. The terms of the next level combine the two differentia of the first step. The concrete term `larva_mesothoracic_femur` has the two generic terms `larva_femur` as well as `mesothoracic_femur` as parent.

Term Definitions

Most of the terms of TrOn were tagged with definitions, which help the user to understand the meaning of the respective term. External sources of information, which were used as base for the definitions, were referenced. We used the definitions of the Arthropod Ontology where possible, the textbook by Sokoloff and the Handbook of Zoology [26] for *Tribolium* specific structures and tried to match the definitions used in FlyBase anatomical ontology where sensible. An expert in insect anatomy was involved in order to finalize the definitions (S.Bradler). Table 1 summarizes the numbers of terms, the respective subsets and the number of relations of the current version of TrOn.

Application of TrOn for a Semantic Search for Morphological Defects at iBeetle-Base

In the ongoing genome wide RNAi screen iBeetle, morphological defects are annotated using a controlled vocabulary. As annotation guideline, the most specific structure affected was to be annotated. For example, when only the most proximal part of the

Table 1. Statistics of TrOn.

	Number
All terms	956
Cross references to other resources for all terms	111
Generic terms	306
Concrete terms	650
Mixed terms	54
Terms with a definition	951
Cross references to other resources for all definitions	2096
`part_of` relations	692
`is_a` relations	1373

doi:10.1371/journal.pone.0070695.t001

walking leg was affected by a knock down, the corresponding term coxa was used. If the entire leg including the coxa was affected, the less specific term `leg` was used. As consequence of this annotation guideline, any simple search using the term "leg" would not identify the more restricted defects e.g. in the coxa or other substructures. Hence, a comprehensive search for all leg defects would have required a complex search combining all substructures with OR, which is not intuitive and error prone.

In order to implement a comprehensive search, the terms of the controlled vocabulary are mapped, in conjunction with the developmental stage, to the concrete terms of the ontology. The relations documented in the ontology allowed finding all phenotypes affecting e.g. the leg and its substructures at all stages by just using the search term "leg". An example is given in Figure 4 where the list of substructures and the number of respective annotations found in a search for "leg" is shown. After a term has been entered in the search field, the number of annotations with respect to this term and its children is given (see black arrow in Figure 4A). This helps the user to quickly assess and optimize the search with respect to specificity.

In detail, the ontology is traversed downstream of the search term in order to identify all of its successors. The search function uses the term hierarchy as well as the `has_part` relationship for the walk downstream. The `has_part` relations are not annotated in the ontology but are transiently generated by the OBA service as the inverse relationship to the annotated `part_of` relation type. The result of the search in the ontology is a list of concrete terms downstream of the search term. If a developmental stage was specified as filter, the search result consists only of the concrete terms, which are linked to this stage.

Another application of the generic terms of TrOn in the search interface of iBeetle-Base is to assist the user to find the appropriate search term. Upon typing a term into the search field, a completion list is displayed, based on the hits in the generic terms (see Figure 4D). As alternative means to identify a search term, an extract of the ontology tree can be displayed, which shows generic terms, the children of which (concrete terms) have been annotated in the screen (see Figure 4B). Together, the completion list and the suggestion tree support the user in finding the right name and level of abstraction for searching the anatomical structure.

After a search, the result list is accompanied by an extract of the ontology with all ontology terms which contributed to the result list. The number next to the term indicates the number of occurrences of the respective term in the results (see Figure 4C).

The OBA service is used to access the ontology in the web application. An OBA module for the iBeetle project implements the described functions and encapsulates the logic to process the ontology in the OBA service.

Discussion

The *Tribolium* Ontology consists of nearly one thousand ontology terms describing most of the external morphology and a, so far, limited set of the internal structures of the red flour beetle. This focus reflects its initiation from the iBeetle project. However, the structures were defined more comprehensively than required for the iBeetle project and include substructures as well as developmental stages which are not annotated in the screen. While most of the external morphology is, hence, represented in TrOn, our systematic approach will allow expanding the ontology as the need arises. For instance, in contrast to the external morphology, internal structures like the central nervous system, musculature and digestive systems were not included or limited to subsets. The public availability of the ontology and the open OBO format will allow the community to adopt and use TrOn for future *Tribolium* projects. The team of iBeetle Base will endeavor to incorporate all comments and extensions of the community to keep a current and central knowledge resource about the red flour beetle.

Our effort to provide definitions for each term will help to use TrOn for additional purposes beyond a controlled vocabulary or a classification of morphological structures. For instance, definitions can be used to add information to applications and web sites which deal with morphological structures of *Tribolium* but do not require functions like the semantic search. TrOn as central repository of morphological terms and definitions may also help to agree on common definitions of morphological structures and will help to avoid divergent definitions of the same structure like the example of the paramere of the Hymenoptera shown by Yoder et al. [14].

Other important extensions to be included in future versions of TrOn are additional relation types. Corresponding structures of the different developmental stages can be connected with the relationship `develops_to` (e.g. the `larva_leg` `develops_to` the `adult_leg` passing through the `pupal_leg`). A `develops_to` relation type will help to clarify situations where one structure gives rise to two new ones and to unambiguously define development of the beetle for electronic systems. For instance, the single structure `larva_tibiotarsus` develops into the separate structures `adult_tibia` and `adult_tarsus`. Other possible relationships like `attached_to`, `bordered_by` would describe the spatial relations in a more sophisticated way. Functional aspects could be introduced with relations like `innervated_by` to denominate the target structures of nerves. Additional facets of the `part_of` relation could assign structures to physiological systems like the fat body or processes like communication and olfaction.

One of the most exciting possibilities opened up by ontologies describing the morphology of different insects is their use for automated cross species comparisons [17]. Together with the prospect of a genome wide analysis of gene function in *Tribolium*, this will allow for the first time to automatically compare the function of genes across insect taxa. In order to realize this potential, the terms of the *Tribolium* ontology need to be linked to the corresponding term of the ontologies in other insects. The basic architectures of the insect bodies have a common blueprint but during evolution, body parts have diversified their morphologies and functions. The wings serve as one example: Ancestrally, winged insects have two pairs of membranaceous wings used for flight. Both *Drosophila* and *Tribolium* have only one pair of membranaceous "wings". In both cases the membranaceous wings are used to generate the required uplift and drive for flight. Also the morphology is comparable, they consist of a double layer of cuticle, and both have a complex and specific pattern of veins. However, these functionally and morphologically comparable structures are located on different segments. In the beetle, the membranaceous wings develop on the third thoracic segment, while in *Drosophila* they belong to the second thoracic segment. With the elytra (second thoracic segment of beetles) and the halteres (third thoracic segment on dipterans) both species have modified the other pair of wings both functionally and morphologically. Hence, crosslinking based on equivalent function of structures may lead to different connections than links based on homology of the structures (i.e. common evolutionary origin). Due to such functional evolution of structures, the cross link between the anatomical ontologies TrOn and FBbt need to be based on the evolutionary criterion of common descent (i.e. homology of the structures). See Bertone et al. for an effort to link the ontologies of different species [17].

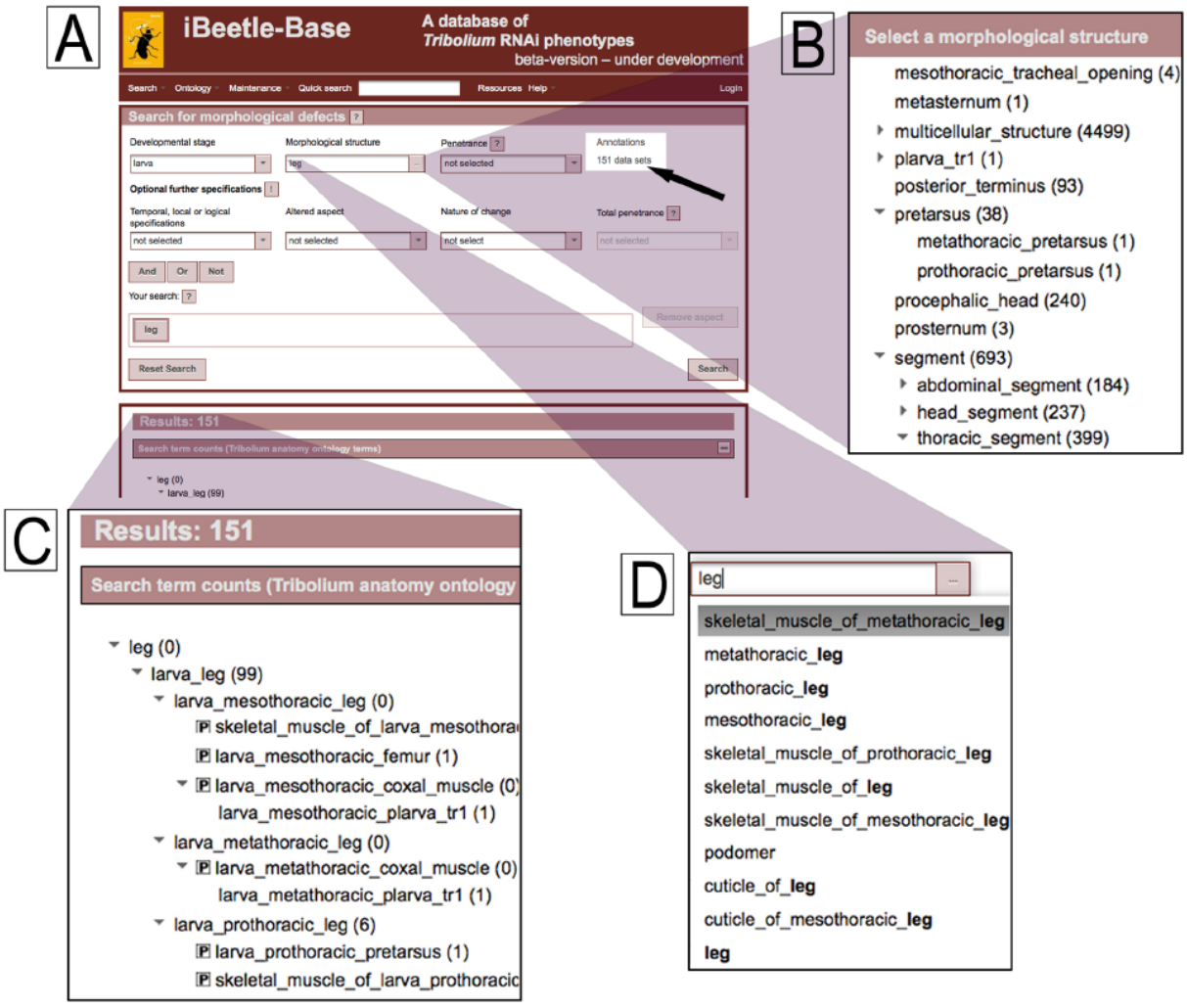

Figure 4. Application of TrOn at the iBeetle-Base. The figure shows a survey and detail views of the search interface of iBeetle Base that can be found at http://ibeetle-base.uni-goettingen.de. A) An overview of the search page is depicted. The black arrow points to the number of search results for the entered search terms. B) A suggestion tree is displayed upon click on the respective button, showing the generic terms of the ontology with annotated features of the iBeetle screen downstream. In brackets, the number of occurrences of a given term in the database is given. C) After a search, an extract of the ontology can be displayed, which represents the terms which contributed to the search result. In this case the number in brackets indicates the number of hits for this term in the search results. D) Upon typing a search term, a completion list is displayed. The displayed terms are generic terms which match the entered term or its synonyms.
doi:10.1371/journal.pone.0070695.g004

In contrast to most anatomical ontologies of insects, in TrOn nearly all morphological structures have a generic term and a separate term for each developmental stage. This emphasizes the different morphology of insects at their different life stages. Upstream of the concrete terms, those that are linked to a development stage, the generic term represents the concept of the morphological structure independent of any stage. This conceptual separation allows unequivocally labeling the unique structure of a given insect with concrete terms, while the generic terms enable cross reference to other ontologies.

A major benefit of a cross linking between ontologies of different species is, among others, the possibility of a phenotypic search across several species. An example is the interconnection of fish phenotypes in the Phenoscape Knowledgebase [30]. However, to realize this aim, a mapping of the morphological ontologies is not sufficient. In addition, also the nature of morphological phenotypes needs to be mapped by a phenotypic ontology, like the Phenotype and Trait Ontology (PATO) [31]. Due to the availability of phenotypic data on a genome wide scale, a comparison between *Drosophila* and *Tribolium* represent a unique opportunity.

TrOn is already at its current stage a useful resource for the *Tribolium* research and a fundamental part of the iBeetle Base. Future priorities for the ontology are: (1) more relations types to describe functional and further connections (e.g. `develops_to`) (2) extend the scope of the covered internal morphological (3) cross-connection to other ontologies like FBbt from FlyBase and other insect ontologies.

Acknowledgments

We thank Edgar Wingender for support and Maike Tech for discussion.

Author Contributions

Conceived and designed the experiments: JD DG CSE GB. Performed the experiments: DG IS CSE. Contributed reagents/materials/analysis tools: NMP. Wrote the paper: JD DG GB. Revised the definitions: SB.

References

1. Bucher G, Scholten J, Klingler M (2002) Parental RNAi in Tribolium (Coleoptera). Curr Biol 12(3): R85–6.
2. Brown SJ, Shippy TD, Miller S, Bolognesi R, Beeman RW, et al. (2009) The red flour beetle, Tribolium castaneum (Coleoptera): a model for studies of development and pest biology. Cold Spring Harb Protoc (8): pdb.emo126. doi: 10.1101/pdb.emo126.
3. Fu J, Posnien N, Bolognesi R, Fischer TD, Rayl P, et al. (2012) Asymmetrically expressed axin required for anterior development in Tribolium. Proc Natl Acad Sci U S A. 109(20): 7782–6. doi: 10.1073/pnas.1116641109.
4. Posnien N, Schinko JB, Kittelmann S, Bucher G (2010) Genetics, development and composition of the insect head–a beetle's view. Arthropod Struct Dev 39(6): 399–410. doi: 10.1016/j.asd.2010.08.002.
5. Wohlfrom H, Schinko JB, Klingler M, Bucher G (2006) Maintenance of segment and appendage primordia by the Tribolium gene knödel. Mech Dev 123(6): 430–9.
6. Konopova B, Jindra M (2008) Broad-Complex acts downstream of Met in juvenile hormone signaling to coordinate primitive holometabolan metamorphosis. Development 135(3): 559–68. doi: 10.1242/dev.016097.
7. Mitten EK, Jing D, Suzuki Y M (2012) Matrix metalloproteinases (MMPs) are required for wound closure and healing during larval leg regeneration in the flour beetle, Tribolium castaneum. Insect Biochem Mol Biol 42(11): 854–64. doi: 10.1016/j.ibmb.2012.08.001.
8. Dreyer D, Vitt H, Dippel S, Goetz B, El Jundi B, et al. (2010) 3D Standard Brain of the Red Flour Beetle Tribolium Castaneum: A Tool to Study Metamorphic Development and Adult Plasticity. Front Syst Neurosci 4: 3. doi: 10.3389/neuro.06.003.2010.
9. Arakane Y, Specht CA, Kramer KJ, Muthukrishnan S, Beeman RW (2008) Chitin synthases are required for survival, fecundity and egg hatch in the red flour beetle, Tribolium castaneum. Insect Biochem Mol Biol 38(10): 959–62. doi: 10.1016/j.ibmb.2008.07.006.
10. Sokoloff A (1974) The Biology of Tribolium. Oxford: Clarendon Press.
11. Grossmann D, Scholten J, Prpic NM (2009) Separable functions of wingless in distal and ventral patterning of the Tribolium leg. Dev Genes Evol 219(9–10): 469–79. doi: 10.1007/s00427-009-0310-z.
12. Schinko JB, Kreuzer N, Offen N, Posnien N, Wimmer EA, et al. (2008) Divergent functions of orthodenticle, empty spiracles and buttonhead in early head patterning of the beetle Tribolium castaneum (Coleoptera). Dev Biol 317(2): 600–13. doi: 10.1016/j.ydbio.2008.03.005.
13. Posnien N, Koniszewski ND, Hein HJ, Bucher G (2011) Candidate gene screen in the red flour beetle Tribolium reveals six3 as ancient regulator of anterior median head and central complex development. PLoS Genet 7(12): e1002416. doi: 10.1371/journal.pgen.1002416.
14. Yoder MJ, Mikó I, Seltmann KC, Bertone MA, Deans AR (2010) A Gross Anatomy Ontology for Hymenoptera. PLoS ONE 5(12): e15991. doi:10.1371/journal.pone.0015991.
15. Smith B, Ashburner M, Rosse C, Bard J, Bug W, et al. (2007) The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotechnol 25(11): 1251–5.
16. Trauner J, Schinko J, Lorenzen MD, Shippy TD, Wimmer EA, et al. (2009) Large-scale insertional mutagenesis of a coleopteran stored grain pest, the red flour beetle Tribolium castaneum, identifies embryonic lethal mutations and enhancer traps. BMC Biol.7: 73. doi: 10.1186/1741-7007-7-73.
17. Bertone MA, Mikó I, Yoder MJ, Seltmann KC, Balhoff JP, et al. (2013) Matching arthropod anatomy ontologies to the Hymenoptera Anatomy Ontology: results from a manual alignment. Database (Oxford) bas057. doi: 10.1093/database/bas057.
18. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, et al. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 25(1) 25–9.
19. Whetzel PL, Noy NF, Shah NH, Alexander PR, Nyulas C, et al. (2011) BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Res W541–5. doi: 10.1093/nar/gkr469.
20. Tweedie S, Ashburner M, Falls K, Leyland P, McQuilton P, et al. (2009) FlyBase: enhancing Drosophila Gene Ontology annotations. Nucleic Acids Res. 37: D555–9. doi: 10.1093/nar/gkn788.
21. Tomancak P, Beaton A, Weiszmann R, Kwan E, Shu S, et al. (2002) Systematic determination of patterns of gene expression during Drosophila embryogenesis. Genome Biol. 3(12): RESEARCH0088.
22. Osumi-Sutherland D, Reeve S, Mungall CJ, Neuhaus F, Ruttenberg A, et al. (2012) A strategy for building neuroanatomy ontologies. Bioinformatics 28(9): 1262–9. doi: 10.1093/bioinformatics/bts113.
23. Knowles-Barley S, Longair M, Armstrong JD (2010) BrainTrap: a database of 3D protein expression patterns in the Drosophila brain. Database (Oxford) baq005. doi: 10.1093/database/baq005.
24. Dönitz J, Wingender E (2012) The ontology-based answers (OBA) service: a connector for embedded usage of ontologies in applications. Front Genet. 3: 197. doi: 10.3389/fgene.2012.00197.
25. Haendel MA, Neuhaus F, Osumi-Sutherland D, Mabee PM, Mejino JLV, et al. (2008) in Burger A, Davidson D and Baldock R. Anatomy Ontologies for Bioinformatics New York: Springer. 327–349.
26. Steinmann H, Zombori L (1999) Handbook of Zoology. A Natural History of the Phyla of the Animal Kingdom. De Gruyter. Vol 4 pt.34.
27. Day-Richter J, Harris MA, Haendel M, Gene Ontology OBO-Edit Working Group, Lewis S (2007) OBO-Edit–an ontology editor for biologists. Bioinformatics 23(16): 2198–200.
28. OntoScope. Available: http://www.bioinf.med.uni-goettingen.de/projects/ontoscope/. Accessed 2013 Jul 3.
29. Winston ME, Chaffin R, Herrmann D (1987) A Taxonomy of Part-Whole Relations. Cogn Sci 11, 417–444.
30. Mabee BP, Balhoff JP, Dahdul WM, Lapp H, Midford PE, et al. (2012) 500,000 fish phenotypes: The new informatics landscape for evolutionary and developmental biology of the vertebrate skeleton. J Appl Ichthyol 28(3): 300–305.
31. Mungall CJ, Gkoutos GV, Smith CL, Haendel MA, Lewis SE, et al. (2010) Integrating phenotype ontologies across multiple species. Genome Biol 11(1): R2. doi: 10.1186/gb-2010-11-1-r2.

3.3 The ontology-based answers (OBA) service

The following manuscript is published as:

Dönitz, J. and Wingender, E.: The ontology-based answers (OBA) service: A connector for embedded usage of ontologies in applications *Front. Gene.* 3, 197 (2012).

Authorships:

- Jürgen Dönitz initiated the project, implemented the service and wrote the manuscript.
- Edgar Wingender supported the project with discussions and helped writing the manuscript.

frontiers in GENETICS

METHODS ARTICLE
published: 05 October 2012
doi: 10.3389/fgene.2012.00197

The ontology-based answers (OBA) service: a connector for embedded usage of ontologies in applications

Jürgen Dönitz[1,2]* and Edgar Wingender[1]

[1] Department of Bioinformatics, University Medical Center Göttingen, Göttingen, Germany
[2] Department of Developmental Biology, Johann-Friedrich-Blumenbach Institute for Zoology and Anthropology, Georg-August University Göttingen, Göttingen, Germany

Edited by:
John Hancock, Medical Research Council, UK

Reviewed by:
Katy Wolstencroft, University of Manchester, UK
Damian Smedley, Sanger Institute, UK

***Correspondence:**
Jürgen Dönitz, Department of Bioinformatics, University Medical Center Göttingen, Goldschmidtstrasse 1, 37077 Göttingen, Germany.
e-mail: juergen.doenitz@bioinf.med.uni-goettingen.de

The semantic web depends on the use of ontologies to let electronic systems interpret contextual information. Optimally, the handling and access of ontologies should be completely transparent to the user. As a means to this end, we have developed a service that attempts to bridge the gap between experts in a certain knowledge domain, ontologists, and application developers. The ontology-based answers (OBA) service introduced here can be embedded into custom applications to grant access to the classes of ontologies and their relations as most important structural features as well as to information encoded in the relations between ontology classes. Thus computational biologists can benefit from ontologies without detailed knowledge about the respective ontology. The content of ontologies is mapped to a graph of connected objects which is compatible to the object-oriented programming style in Java. Semantic functions implement knowledge about the complex semantics of an ontology beyond the class hierarchy and "partOf" relations. By using these OBA functions an application can, for example, provide a semantic search function, or (in the examples outlined) map an anatomical structure to the organs it belongs to. The semantic functions relieve the application developer from the necessity of acquiring in-depth knowledge about the semantics and curation guidelines of the used ontologies by implementing the required knowledge. The architecture of the OBA service encapsulates the logic to process ontologies in order to achieve a separation from the application logic. A public server with the current plugins is available and can be used with the provided connector in a custom application in scenarios analogous to the presented use cases. The server and the client are freely available if a project requires the use of custom plugins or non-public ontologies. The OBA service and further documentation is available at http://www.bioinf.med.uni-goettingen.de/projects/oba

Keywords: ontology, semantic function, ontology-based answers, OBA

INTRODUCTION

Ontologies play a major role in the semantic web (Berners-Lee et al., 2001). Running in the background they provide electronic systems with the expertise of a knowledge domain. Through formal and logical statements ontologies are useful to unambiguously identify and define entities representing material objects as well as abstract concepts and their mutual relations. By connecting unknown terms with known ones through defined statements, new knowledge can be deduced. This knowledge can be used to provide the user with information that he/she is seeking but could not exactly specify. This is achieved by means of a mandatory class hierarchy, using the "is_a" relation, and other relations, connecting the ontology classes to each other. Supplementary data can be added to each ontology class by annotations. While the meaning of relations is comprehensible to human users so that they can select the right one for traversing the graph, it is a particular challenge to transfer the logical axioms defined in an ontology into an object-oriented view that is common to most applications (Winston et al., 1987; Burger et al., 2008).

A multitude of tools and web services dealing with ontologies are available in the biomedical field. Ontology browsers like Amigo (Carbon et al., 2009) for the Gene Ontology (GO; Ashburner et al., 2000) or ontology editors (OBOEdit: Day-Richter et al., 2007; Protégé[1]) let the user work interactively with an ontology. The web services Ontology Lookup Service (OLS; Côté et al., 2008), the NCBO BioPortal (Noy et al., 2009) and OntoCAT (Adamusiak et al., 2011) facilitate the search function covering all ontologies publicly available at the NCBO portal or the OBO-Foundry (Smith et al., 2007) and provide access to their content. OntoCAT and the BioPortal also offer an interface to be queried by electronic systems over the network. By doing so OntoCAT additionally offers a Java and R client (Kurbatova et al., 2011) for communication with the service.

The listed portals offer services which are highly valuable to the community. However, they fall short in two aspects: by approaching the access of a collection of ontologies in a standardized way, the portals lack functions that are specific for individual ontologies, leaving the information encoded in the diverse relationships unattended. An automated system does not allow the user to decide when to use which relationship, the algorithm

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

has to solve this problem. The application developer is required to be familiar with the annotation guidelines and implement the required algorithm.

If a search interface allows the user to enter or select an anatomical structure, for which data should be displayed, the user will expect results not only for the selected structures, but also for substructures and perhaps functionally related structures. With the use of ontologies this challenge can be met. The different sets of available relations used in ontologies like "part_of," "contained," or "bordered_by" require an implementation of such a search algorithm to be ontology specific.

The other challenge is between the semantics of ontologies, consisting of a set of axioms, and the modern style of object-oriented programming. In an ontology the classes and their relations are stored in separate axioms while in an object graph the objects themselves have knowledge about the links to their neighbors. APIs like OWL-API (Horridge and Bechhofer, 2011) or Jena-API (Jena – A Semantic Web Framework for Java[2]) facilitate full access to ontologies and follow their design principles. They disclose any information and logic of the supported ontology format to the user. The resulting complexity prevents a straight way to get, e.g., neighbors of a class from the ontology. To get the subclasses of an ontology class with the OWL-API the axioms for the superclass has to be fetched and the right axioms have to be selected. Also when using the ontology portals an additional request to the portal is required because the ontology classes fetched from the portals lack a method to access their own subclasses.

As an alternative way we suggest a service providing ontology-based answers (OBA service). To benefit from ontologies the OBA service can be embedded in applications and workflows. The OBA project's goal is to make knowledge, which a user can intuitively retrieve from ontologies, available to applications or to workflows processing high-throughput data. The service provides semantic functions that implement knowledge about the curation guidelines as well as the used relations and their interpretation. The client of the service can be embedded into custom applications and maps the service's responses to a graph of Java objects. The OBA service provides the main information stored in ontologies to computational biologist not familiar with ontologies. The developers are enabled to concentrate on their research topic while working with the familiar object-oriented programming style.

Use cases and projects are presented to demonstrate the concept and advantages of OBA. In the use cases the Cytomer ontology and the iBeetle project are used. Cytomer[3] is an ontology concerning anatomical structures of humans in adults and during the fetal development (Heinemeyer et al., 1999; Michael et al., 2005). Specific relations describe the progenitor, the derivation and the appearance in the Carnegie stages.

The iBeetle project[4] aims to identify genes essential to insect development and physiology by genome wide gene silencing in the red flour beetle *Tribolium castaneum* (Schröder et al., 2008) using parental and larval RNA interference (Bucher et al., 2002; Tomoyasu and Denell, 2004). During the first part of the iBeetle project, several thousand genes have been silenced and the observed phenotypes are stored in a database and linked to an anatomical ontology for *Tribolium* (Bucher and Klinger, personal communication).

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

[3]http://cytomer.bioinf.med.uni-goettingen.de

[4]http://ibeetle-base.uni-goettingen.de/

MATERIALS AND METHODS

A service which helps to bridge the shortcomings of existing tools, as it is described in Section "Introduction," should fulfill the following requirements:

- The service should enable an application developer to deal with the ontology in a transparent manner rather than enforcing him to deal with different ontology formats or low level APIs.
- The service should map the ontology classes and their connections to a graph consisting of Java objects.
- The part processing the ontologies should be separated from the part which is embedded in the application. A server process would in addition offer a central ontology server.
- The communication with the server should be encapsulated by a connector on the client side to provide network transparency for the custom application.
- The service should implement knowledge about the used ontologies and provide the information deduced from the ontologies by simple Java methods to a computational biologist.
- With more in-depth knowledge about the used network interface or ontologies the service should be extensible to match the requirements of new or custom ontologies and projects.

The OBA service consists of a server and a client part, which communicate using the Representational State Transfer (REST) architecture (Fielding, 2000). **Figure 1** gives an overview of the OBA service design. The server can load any ontology in the OWL (Lacy, 2005) or OBO format (Smith et al., 2007) and host semantic functions. For every ontology a basic part of the server provides access to the entities, connected entities and lists of entities. Each entity is accessed by a unique Uniform Resource Locator (URL). Entities linked to another entity, like its child or parent classes, can also be accessed by a URL denoting the required subresource. Like the content of the ontologies, the semantic functions are available through URLs and return entities or a list of entities as answer.

A list of entities can be stored on the server in order to facilitate the work on more comprehensive input. This data can be used, for example, to limit the results of a search to members of a list of entities used in an application. To manage resource allocation, the storage area is divided into partitions. A user or a work group can create their own partition to store one or more lists. Such a partition is only accessible through its assigned name, allowing a basic access control.

The server uses a REST interface and provides the data in the "application/json," "text/plain," and "text/html" format (MIME-types). The open architecture allows the user to communicate with the server via a command line client, a web browser or with any custom client. The preferred form is the embedding in custom applications. For easy integration into applications a Java client is provided. The client encapsulates the network communication and facilitates access to the semantic functions of the server and to the entities of the respective ontology by Java functions.

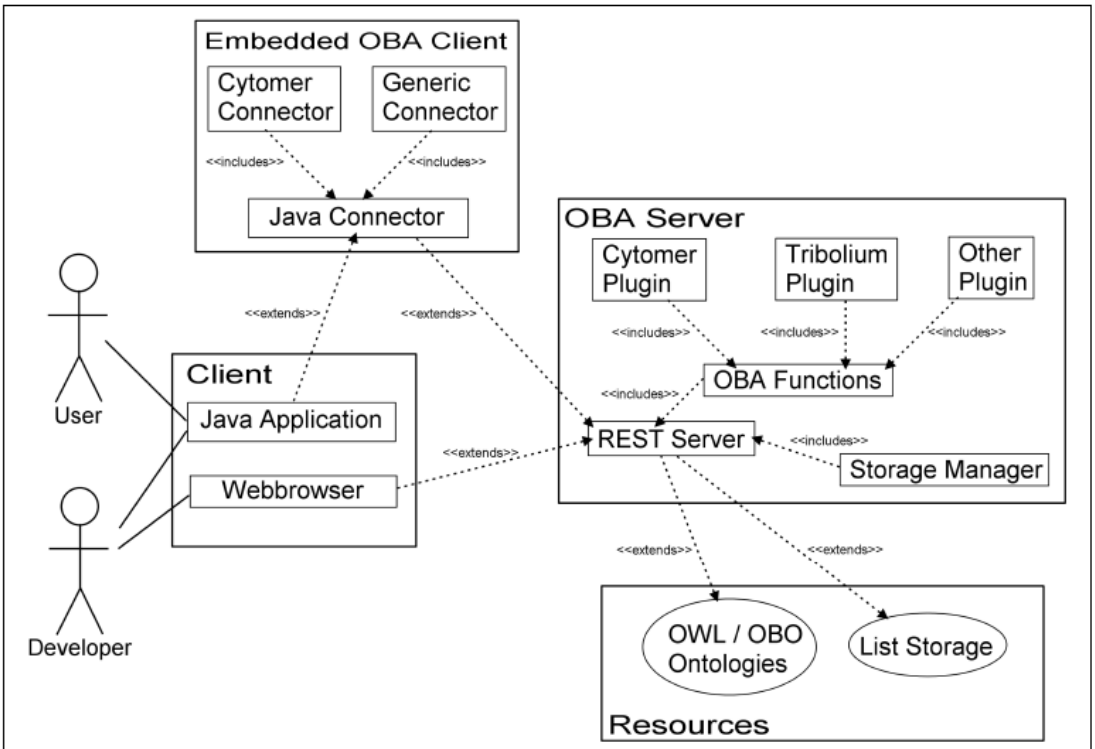

FIGURE 1 | The components of the OBA project and their relations. The components of the OBA project are displayed together with their connections to external parts, to ontology files or to other applications.

The server's response is converted into Java objects, containing methods to access super- and subclasses as well as annotations and relations. To avoid loading the whole ontology upon the first request, the Java objects representing the ontology classes function as proxies that load connected objects upon the first access. This lazy loading is completely transparent to the application.

By default, the client uses the public server available at http://oba.sybig.de. Currently, this server provides access to the Cytomer ontology, the *Tribolium* anatomical ontology (TrOn) and the GO with ontology specific functions for the first two and generic semantic functions for all ontologies. To access custom ontologies or to implement individual OBA functions, the server and the client can be downloaded and extended. The server can load plugins to add custom OBA functions to meet new requirements of a specific project or ontology. The module containing the basic functions implements the plugin interface and can be deemed as built-in plugin. Two additional plugins, one for the Cytomer ontology and one for the iBeetle project, are already available and can serve as templates for the development of new plugins. Client extension is achieved by subclassing. These subclasses can provide Java functions to access semantic functions of a custom plugin or provide convenient functions to access annotations or relations of the ontology's classes. Each ontology has its own defined set of relations and annotations. The generic client has no knowledge of the specific sets of annotations and relations for an ontology and enables access to the annotation and relations as two-dimensional lists containing the type of the annotation or relation and the respective values. To get the synonyms annotations of an ontology class, the application has to iterate the list of annotations until the desired one is found. A custom client can provide the method "getSynonyms()" encapsulating this iteration.

The OBA server and the example client are implemented using the Java Platform. The OWL-API is used to access ontologies in OBO or OWL format. To implement the REST-protocol the Jersey library was selected[5]. The Grizzly HTTP container handles the network communication on the server side[6]. To index the ontology's classes the Lucene library[7] is used. To store the metadata of the uploaded data HSQLDB[8] was selected.

[5] http://jersey.java.net/
[6] http://grizzly.java.net/
[7] http://lucene.apache.org/
[8] http://hsqldb.org/

RESULTS

With the OBA service a software application was developed to fulfill the requirements listed above (see Materials and Methods). The division into a server and a client component allows the separation of processing the ontologies from the specific custom applications. The server has access to the ontologies and hosts plugins with the OBA functions. These functions make intensive use of the ontologies and transfer the processed results to the client. The plugins encapsulate the implementation details to process the ontologies and reduce the complexity to a single function call on the client's side. The concept of the OBA functions as a server side component is a new concept not known to the existing ontology portals.

The OBA client maps the OBA functions to Java functions and the ontology classes to Java objects. The objects representing the ontology classes have functions implemented to access their parents, children, and connected ontology classes. To avoid loading the complete ontology from the very beginning the neighboring classes are loaded upon the first access by a proxy functionality. The Java objects created by the OBA client are internally equipped with a link to the Java connector to load missing neighboring classes in the background. In contrast to existing solutions this loading process is completely transparent to the user. The developer is able to accesses the neighboring classes through Java methods and does not have to be concerned about their loading from the backend. The OBA client facilitates also access to the OBA functions by simple Java methods. Using the OBA client the network access and the implementation details of the OBA functions are transparent to the application developer, who can thus focus on the scope of his custom application.

The following use cases illustrate some OBA functions and how OBA is already used in some upcoming projects. The description of the OBA functions reveals the implementation details of these functions to show how the ontology is processed. The application developer can use these functions with a single function call and is not required to reimplement the logic.

OBA FUNCTION: GENERIC SEARCH

The function "searchCls" is used to search for an ontology class matching a pattern that has been specified by the user. The search is not limited to the name of the ontology class, but the annotation fields of the class are included. On the client side the annotation fields to be used for the search can be specified.

Table 1 shows the result of a search for "cistern" in the Cytomer ontology. In the second case the search is restricted to the annotation "definitionEnglish." The search function of the Java client also provides the possibility of limiting the search to selected annotation fields. This possibility is not common in existing tools but is a powerful filter to get more precise search results.

The search functionality uses the name of the ontology class as well as its annotation fields and works with any loaded ontology. The classes returned by the search function can serve as starting point for traversing the graph or as input for other OBA functions.

OBA FUNCTION: MAP ONTOLOGY CLASSES TO ANCESTORS

The goal of the following two functions is to map ontology classes to more abstract ancestors. The function "reduceToLevel" requires the input of a level and a single ontology class or a list of them. Each one of the classes from the input is mapped to all ancestors at the given level beneath the root node. To determine the ancestors of a class, all paths between the start class and the root class are considered. Due to the fact that an ontology class can have more than one parent, there might be more than one path, resulting in multiple ancestors for a single class at a specific level. If the node "negative regulation of binding" in **Figure 2** is mapped to level five, the two nodes "negative regulation of molecular function" and "regulation of binding" are returned. The function can also be called with a reference to a previously uploaded list of ontology classes. In doing so a list of classes with different levels of abstraction are mapped to classes at a constant and equal level below the root node.

A similar approach is implemented in the function "reduceToClusterSize." In this case the ontology classes are successively mapped to their parents. In each iteration only those classes with the greatest distance to the root class are mapped to their parents. The process is finished when the number of resulting ontology classes is not larger than the specified number. The result is a list of clusters, each with a list of ontology classes from the input list, mapped to this class. Due to the specification of a maximum number of clusters instead of a concrete level, the resulting clusters may have varying distances to the root class. However, by processing the farthest ontology classes in each step, this effect is minimized. The marked nodes in the example of **Figure 2** will be mapped to the nodes "regulation of signaling" and "regulation of protein binding" if the maximum number of clusters is set to the value of two. The node "regulation of cytokine activity" is mapped in

Table 1 | Generic search with a limitation to an annotation field.

http://oba.sybig.de/cytomer/functions/basic/searchCls/cistern	http://oba.sybig.de/cytomer/functions/basic/searchCls;field=definitionEnglish/cistern
cistern, pontocerebellar_cistern, chyle_cistern, ambient_cistern, lumbar_cistern, quadrigeminal_cistern, interpeduncular_cistern, chiasmatic_cistern, pericallosal_cistern, cistern_of_lamina_terminalis, lateral_cerebellomedullary_cistern, vein_of_cerebellomedullary_cistern, posterior_cerebellomedullary_cistern, cistern_of_lateral_cerebral_fossa, basilar_artery	pontocerebellar_cistern, basilar_artery

Result of a search for "cistern" in the Cytomer ontology. In the first case the pattern is searched in the class name and all annotation fields including the comment field. In the right column the search is limited to the annotation "definitionEnglish" by a matrix parameter in the URL.

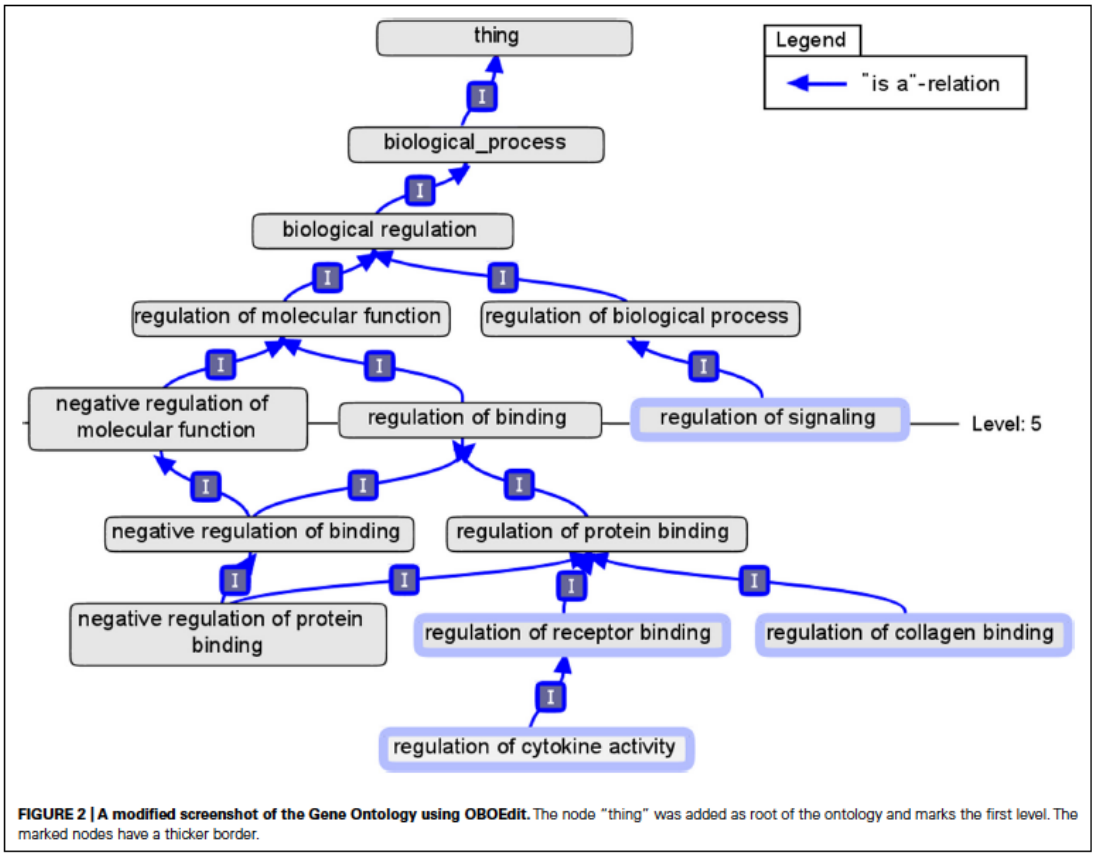

FIGURE 2 | A modified screenshot of the Gene Ontology using OBOEdit. The node "thing" was added as root of the ontology and marks the first level. The marked nodes have a thicker border.

each step, while "regulation of signaling" is just copied to the result set. The classes representing the final cluster do also have different distances to the root node, five and six in this case.

The functions described in this section relay on the class hierarchy and are therefore not ontology specific, they can process any currently loaded ontology as well as the ontologies added in the future. When the described function has to be implemented with existing tools the effort is larger. To map ontology classes to a given level all classes from the starting class up to the root node have to be fetched to determine the classes on the required level. The other classes can be dismissed afterward. The OBA functions simplify the tasks by hiding the processing step behind a function call provided by the OBA client.

The result of a gene expression experiment is a list of differentially expressed genes. A common way to analyze this gene list is to map the genes to the corresponding terms of the GO. The mapping can be done for example with the help of BioMart from Ensembl (Kinsella et al., 2011). Apart from a statistical analysis the resulting list of GO terms can be mapped to more abstract terms until the list is short enough to give an overview of the main processes the GO terms belong to. This can easily be achieved with the two OBA functions "reduceToClusterSize" and "reduceToLevel" and gives a first and intuitive impression of the experiment's outcome.

USE CASE: CYTOMER-SPECIFIC FUNCTIONS

In the following the advantages of OBA functions provided by the service are demonstrated using the anatomical ontology Cytomer. In biomedical research different anatomical structures are investigated. These anatomical structures can be cells, tissues, organs, and entire body parts. A common example is the handling of gene or protein expression data derived from cells, organs, or biopsies (Uhlen et al., 2010). For an analysis on an equal level of abstraction, it is preferable to map all anatomical structures to the level of organs. These steps need to be automated for high-throughput data.

OBA function: get organs of an anatomical entity

The function "organsOf" of the OBA service accepts an arbitrary class of the Cytomer ontology, which represents an anatomical structure as input and returns its respective organs. Inside this function the organs are searched along the class hierarchy and along the selected relations "isPartOf," "isPartOfOrgan," and

"isCellOf." Other relations, for example relations describing the development, are ignored in this case. **Figure 3** shows a simplified, abstract section of Cytomer. Using the function "organsOf" on "Cell 1" "Organ 3" is found using the two relations "isPartOf" and "isCellOf." For "Cell 2" the two nodes "Organ 1" and "Organ 2" are found. "Organ 3" is not part of the result, because the relation "differentiatesInto" between the nodes "Cell 2" and "Cell 1" is not considered for the search of the organs of an anatomical entity. To retrieve the physiological system of an anatomical entity the function "physiologicalSystemsOf" can be used, which works in an analogous way.

OBA FUNCTION: MAP TO A PREDEFINED LIST

An alternative approach is to store the data linked to the most precise anatomical entities, even if these entities do not belong to the same level. In this case a user needs help to draft a request. If the request is on another level than the stored data, no match may be found, although there are relevant entries on a more abstract or more concrete level. The two functions "findUpstreamInSet" and "findDownstreamInSet" of the OBA service provide a solution for this use case. In a set-up step the list of anatomical structures represented by the input data, is stored on the OBA server. The list can be reused for each user's request. Starting from the class, which has been requested by the user, the ontology is searched until a class in the previously uploaded set is found. For an illustration of these two functions please refer to **Figure 4**. The graph is a simplified view of the Cytomer ontology. The yellow nodes are anatomical structures used in EndoNet and uploaded to the OBA service. In the first example the user is searching information on nephron, which would give no result in EndoNet. The function "findUpStreamInSet" searches upstream of the start class "nephron," until a class is found which is also in the previously uploaded list. In this case, following the "isPartOf" relation "kidney" is found, to which EndoNet can provide information to the user. The example of the function "findDownStreamInSet" starts with the abstract term "digestive_organ" and returns "liver" and "pancreas" as matching classes in EndoNet, by following the class hierarchy. The nodes and edges marked with a green shape are the entities processed during the mapping. The search only stops when a member of the predefined list is found, or no more nodes up- or downstream along the class hierarchy or the used relations are available.

These two functions contain a list of relations usable for the up- and downstream search. The path from the starting class to the result nodes may contain any mixture of the intended relations for the requested search direction. The length of the path is not limited, the breadth-first search stops in the iteration step with the first match and returns all matches found in this step.

The OBA functions presented above process a graph's representation of the Cytomer ontology containing the ontology classes, the class hierarchy, and other relationships between the classes. As ontology specific information the functions have the knowledge implemented when to use which relation and how organs or physiological systems can be identified. Processing the graph's

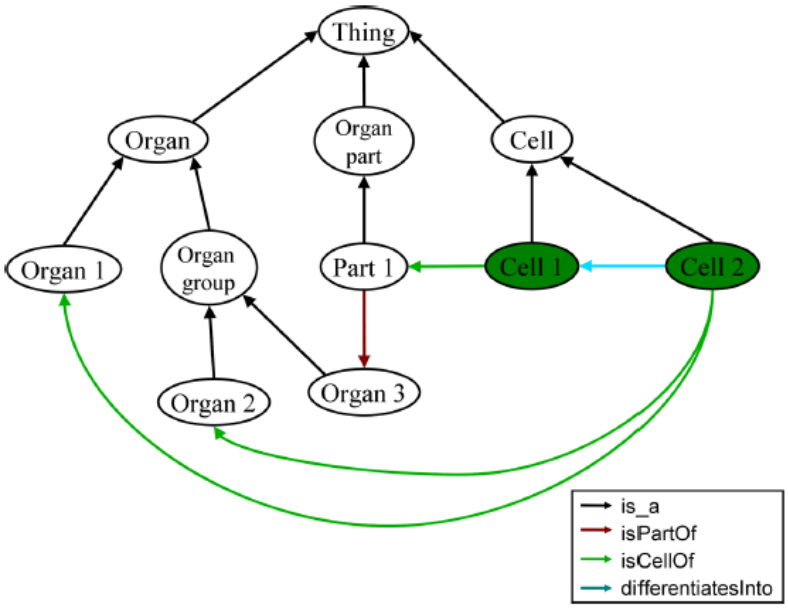

FIGURE 3 | Abstracted and simplified view of the Cytomer ontology illustrating the handling of organs of an anatomical entity. The green nodes are the start nodes for the search function specified in the text. In this section, the entities are connected by four different relations given in the legend.

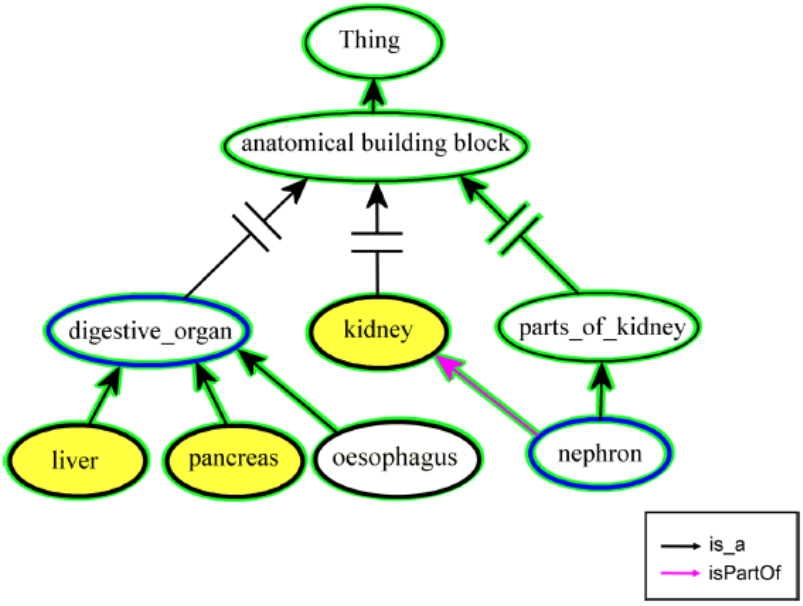

FIGURE 4 | Mapping entities to a predefined list. The nodes with the blue border represent the start nodes for the functions "findUpstreamInSet" and "findDownstreamInSet," respectively. The nodes and edges marked with a green background shape are processed during the mapping. The classes of the result set have to be members of the predefined set which contains the yellow nodes. A predefined list can be used by a project to limit the result of a up- or downstream search to a set of classes used in the project.

representation is done by the OBA framework, to implement analogous functions for other ontologies or similar tasks, a new plugin can reuse this existing logic and only the ontology or task specific knowledge needs to be added, i.e., the relations to use and the key classes.

To achieve a comparable result with existing ontology portals is much more complex. In order to retrieve all organs for an arbitrary anatomical structure using the existing ontology portals the user has to decide which of the relations of the starting class could be used to traverse the ontology graph to some organ. In the next step, all neighboring classes linked by the selected relations have to be queried from the portal. The last two steps have to be repeated for every fetched intermediate class multiplying the number of classes in each step. Whether one of the processed classes represents an organ has to be decided by the users based on their medical knowledge or based on rules deduced from the curation guideline of the ontology. Using the OBA function "organsOf" all these steps are executed on the server where the knowledge is implemented which ontology classes represent the concrete organs. Due to the multitude of relations to consider, 70 ontology classes are processed to return "liver" as organ for the ontology class "hepatocyte." To get the organs lung, larynx, and trachea for the ontology class "sensory_epithelial_cell" 2,497classes are needed to be checked. Without OBA each of these classes has to be downloaded from an ontology portal and processed locally. The numbers are dependent on the starting class and the version of the used ontology. New or removed relations can have a great impact on the number of processed ontology classes. However, for simple queries like the example of the hepatocyte cell, a considerable number of ontology classes already have to be processed. Using OBA the result is always achievable with one single function call. Even changes in the annotation guidelines, like new relations' types, of the used ontology would be encapsulated in the plugin and hidden from the application developer.

PROJECT: iBeetle

In the iBeetle project genes are silenced by RNAi and the observed phenotypes for several stages are annotated into a database following the Entity–Quality (EQ) system (Washington et al., 2009). During the project a detailed ontology about the anatomical structures of *Tribolium* in different developmental stages has been created. There is an ontology class for each structure at every developmental stage where this structure exists. Thus there are distinguished classes for the pupal and the larval antenna. Both are linked with an "isPartOf" relation to the corresponding developmental stages and share the same generic superclass "antenna". The annotations are linked to the classes connected to a developmental stage instead of being linked to generic ones. The most detailed level in the ontology is chosen for the annotation, i.e., flagellum is used if the phenotype affects only the flagellum and not the whole antenna. For the search interface the requirements are different. A typical input is the developmental stage and a generic

and rather abstract morphological structure, e.g., antenna instead of flagellum. To fulfill the demands and provide a general access to the *Tribolium* ontology the OBA service is embedded into the search interface and a server plugin with specific semantic functions has been implemented.

Upon startup the OBA service scans the ontology for concrete classes (these connected to a developmental stage) and generic classes, respectively. The concrete classes do not necessarily have a direct relation to a developmental stage, the path to the stage may be a collection of "is_a" and "isPartOf" links. The generated list of generic classes is used as a suggestion list for the user while typing into the search form. When the user has chosen a developmental stage and an anatomical structure, the OBA service selects all concrete classes downstream of the selected structures and connected to the appropriate stage. Because "isPartOf" is used in the *Tribolium* ontology to describe meronomic relation, the inverse "hasPart" relation is generated on the fly. The list of ontology classes is used as input for the search in the database of the iBeetle project. As add-on on the result page a tree with the subsections of the ontology that were used for the search is displayed. **Figure 5** shows a screenshot of this ontology tree. The semantic search started with the search term "head" and added all ontology classes representing head and its parts.

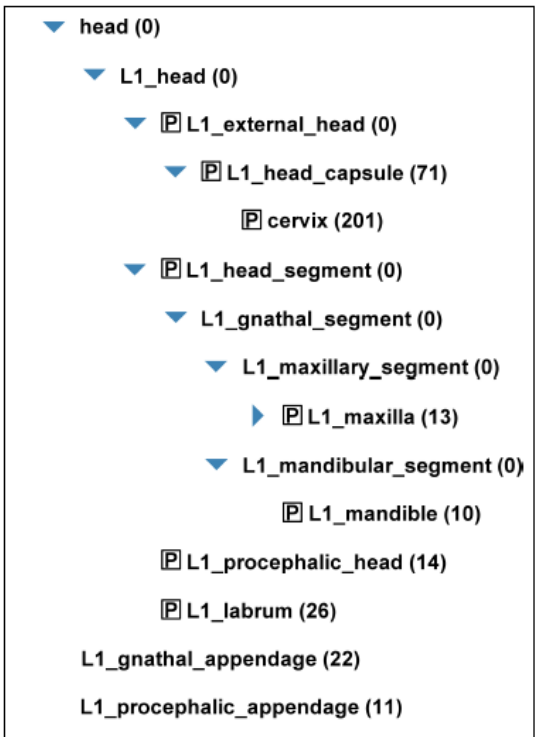

FIGURE 5 | Ontology tree from the result page of a search in the iBeetle database. The tree shows the classes of the *Tribolium* ontology downstream of the searched structure and linked to the queried developmental stage. In this example, the user has selected "head" as anatomical structure and "larva" as stage. All ontology classes shown in the tree where used for a search in the iBeetle database. The numbers in parentheses indicate the number of hits linked to this node.

PROJECT: EndoNet

For the upcoming new web interface for EndoNet, an information resource of the human endocrine system (Dönitz et al., 2008), a semantic search, similar to the search function described above is used. As ontological data source the anatomical ontology Cytomer is utilized. In this case the focus is not on developmental stages but on grouping the annotated cells and tissues at the level of organs in order to generate a survey map of general pathways. To limit the search result to anatomical structures used in EndoNet a predefined list containing the anatomical structures used in EndoNet is stored on the OBA server.

PROJECT: OntoScope

Another type of application using the OBA service is the ontology viewer OntoScope[9]. OntoScope visualizes ontologies as a graph extending the common tree like view of ontologies. The representation as a graph enables the user to explore ontologies along arbitrary relations. OntoScope uses from the OBA service the object graph and the access to the ontologies without any knowledge about the format or semantics of the ontology. OBA functions are used in the background, so that for example the nodes of the Cytomer ontology can be displayed in a color code according to the physiological system. **Figure 6** shows a screenshot of OntoScope with several nodes and relations.

Table 2 summarizes the OBA functions used in the projects. The plugin containing the function is named and a short description is given.

INSTALLATION AND EXTENSION OF OBA

For the use of OBA in a new application the Java client has to be downloaded and added to the class path of the application. After the initialization of the connector, all OBA functions are accessible as Java methods through the connector. The OBA functions will return single ontology classes or lists of them. These ontology classes are mapped to Java objects by the connector and returned by the Java methods of the connector. The Java objects provide functions to access the annotations and neighboring classes of the represented ontology class. If necessary missing information is queried internally from the OBA server. The application developer does not have to be concerned about the retrieval of neighboring classes.

If a required ontology is not available on the public OBA server, it can be downloaded and started locally. After the extraction of the zip file default directories for ontologies, plugins, and the storage area are available. New ontologies can be copied to the ontology directory together with a short property file. The property file defines under which name the ontology will be available from the OBA server and which annotation fields should be indexed for the

[9]http://www.bioinf.med.uni-goettingen.de/projects/ontoscope/

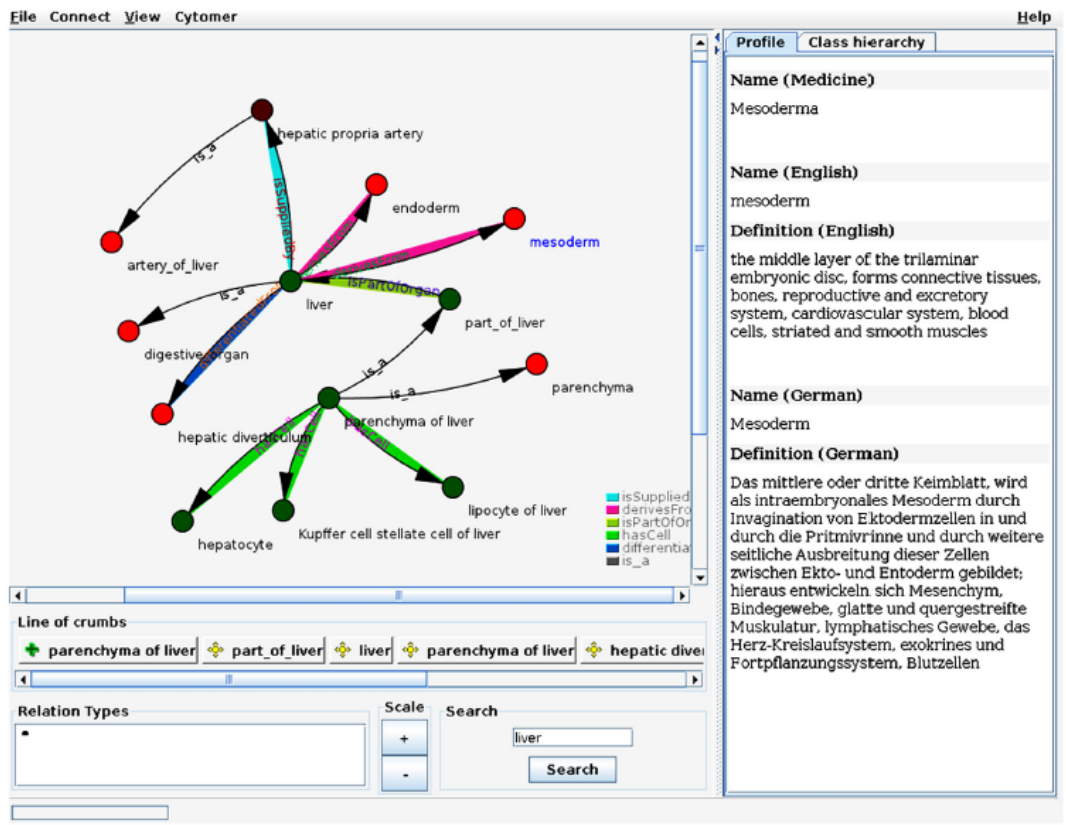

FIGURE 6 | Screenshot of OntoScope. The ontology viewer OntoScope uses the object graph from OBA to let the user browse the graph. The ontology classes are represented by the nodes of the graph. Each relation type has an own color and are displayed as edges connecting the nodes. The color of the nodes indicates the physiological system the class belongs to and is retrieved from the OBA service. On the tabs of the right side additional information of the selected ontology class is displayed, or the class hierarchy of the classes in the graph (tab is hidden in the screenshot).

search function. The property file can be copied from the provided examples and is described in the manual.

SUMMARY

The OBA service is available online at http://oba.sybig.de. Upon pointing a web browser to this URL an overview is given as a list of loaded ontologies as well as the available plugins and the OBA functions implemented by them. The object graph of the ontologies can be browsed by following the links of the HTML representation of the ontology classes. The syntax to access the OBA functions is described in the manual available at the home page of the project: http://www.bioinf.med.uni-goettingen.de/projects/oba. Located on the home page of the project is the Java connector as well as all sources and jar files for the server and currently available plugins. The Cytomer connector contains a test client, which is executed when the client is run on the command line. This client calls some functions on the server and prints the results to the console in order to validate the OBA service's function. The client's sources can serve as a template for a usage of OBA in a custom application.

To give the user a first impression of the function of the OBA service, a web demo is available at http://webdemo.oba.sybig.de/ implementing some of the provided functions for manual tests. For each step the example source code is noted, which is needed to implement the corresponding step in a custom application.

DISCUSSION

Ontologies are powerful and also complex tools. This is especially true for the OWL format. Parsers like the Jena-API (Jena – A Semantic Web Framework for Java[10]) or the OWL-API (Horridge and Bechhofer, 2011), take care of parsing ontologies but do not intend to hide the semantics of ontologies. The same is true for OBO ontologies, although they have a more finite

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

Table 2 | Overview of the OBA functions used in the projects.

Project	Used OBA function	Plugin	Functionality
iBeetle	concreteClasses	Tribolium	Returns all classes linked to a developmental stage. The annotated phenotypes are linked to these classes
	genericClasses	Tribolum	Returns all classes not related to a developmental stage, used for the auto-complete function of the search interface
	findInGeneric	Tribolium	Searches in the labels and synonyms of generic classes and an additional previous generated list for classes matching the search string. Used for the auto-complete function in the search interface
	concreteForDevStage	Tribolium	Returns the class downstream of the given generic class and linked to the given developmental stage. Used to map the user query to the annotations stored in the database.
EndoNet	findUpStreamInSet findDownStreamInSet	Cytomer	Used to find entities from EndoNet related to the search term
OntoScope	physiologicalSystemOf	Cytomer	Returns all physiological systems of an ontology class, used for coloring in the graph
	searchCls	built-in	Searches ontology classes matching a text pattern in the class name or annotation field

The table summarizes the use of OBA in the projects listed in the first column. The second and the third column denominate the OBA function name and the plugin containing the function. The last column describes the functionality of the OBA service the projects benefits from.

structure. If a developer plans to include information deduced from ontologies in an application, a time for training is needed to learn the semantics of ontologies and the framework's design. The basic tutorial of the OWL-API already consists of over 100 slides and deals with a semantic most computational biologists are unfamiliar with. The OBA service maps the relevant parts of ontologies to the world of object-oriented programming and provides semantic functions. The usage of the OBA service does not call for intensive training time to work with different topics and programming paradigms. The simplification to an object graph is oblivious to advanced features of OWL like cardinalities or different OWL dialects. If such a full access is needed, it can be achieved with the very good ontology APIs, i.e., Jena-API or OWL-API, with the query language SPARQL or Protege for interactive work. However, the OBA service can load and process any ontology in the OBO or OWL format, giving access to their fundamental information to developers who otherwise would probably not use ontologies.

Portals like OntoCAT (Adamusiak et al., 2011), the OLS (Côté et al., 2008), or the NCBO BioPortal (Noy et al., 2009) aim to provide access to huge collections of ontologies in a standardized manner. This is the preferred way if the unique definitions of terms in ontologies take precedence over the complex relations. Like the OBA service, OntoCAT and the NCBO ontology portal allow the user to access ontologies using the REST-protocol. OntoCAT also provides basic clients for different programming languages. In addition to the functions of the OntoCAT client, the Java objects of the OBA service provide the required functions to access the super- and subclasses as well as classes which are linked by relations. Together with the proxy function, the basis of the new feature in the OBA service is to map ontology classes to an object graph, traversable by Java methods. The required network communication with the service is encapsulated by the OBA client and transparent to the user. The feature to grant access to the neighbors of an object, representing an ontology class, by Java methods is beyond the function provided by the clients of the existing ontology portals. Together with the proxy function of the OBA client the developer is now enabled to access ontology classes and traverse the graph using only Java methods. Network access and parsing of the ontology is transparent.

One intention of the OBA service is to relieve the user from ontology specific demands by encapsulating the logic in a service. With the OBA functions the developer benefits from the rich information of a specific ontology encoded in the relations without the detailed knowledge about these semantics. The goal of the OBA service is not primarily to provide network access to ontologies, but to add additional functions to help a developer to solve a subtask of an application based on information available in ontologies without being familiar with ontologies, APIs, or query languages to process them.

The OBA service's concept of semantic functions is distinct from the goal of ontology portals like OBO-Foundry (Smith et al., 2007), NCBI, or OntoCAT. The portals focus on accessing as many ontologies as possible. This approach is very well suited for an ontology overarching search and access. The OBA service provides access to a set of specific ontologies with matching semantic functions. If a plugin with the required semantic function is already available the developer saves time for training and programming. Even if the required function is not available, the developer benefits from the framework of the OBA service and the advantages of the client described above. The OBA framework and the open architecture minimize the effort of extending the service to fit the requirements of a specific project. A new plugin relays on the existing functions to access the ontology, marshal the objects for the network transfer as well as the proxy functionality of the client. A new plugin only has to implement knowledge about a custom ontology or the logic to solve a new question. Due to the provided framework the already supplied plugins are very small and easy to implement. The developer of a new plugin needs to be familiar

with the curation guideline of the used ontologies. Further expertise about ontologies, like the different formats and ontology internals like Frames, Description Logic are not required.

Under the umbrella of the OBO-Foundry a collection of tools handling ontologies has evolved. There is a number of tools supporting the annotation process or focusing on statistical analysis of data based on ontologies, examples are the tool DAVID (Huang et al., 2009) and tools for the gene set enrichment analysis (GSEA) method (Subramanian et al., 2005). Like the functions of the OBA service, these tools make intensive use of the GO or other ontologies. The advantage of the OBA service is that it is easily extendible. The server can load plugins for any ontology. The service is designed to be embedded into applications and workflows to minimize interaction with external tools.

The design of the OBA service has several advantages. A public server is the central contact point and serves a growing collection of publicly available ontologies and plugins. Developers and maintainers of an ontology are welcome to submit new plugins, which enables the scientific community to profit. Alternatively, the server can be downloaded and run locally if the required ontology is not available in the public repositories, or if the developed plugin is not to be published.

The new features of OBA are the seamless mapping of ontologies to a connected object graph for object-oriented programming and the implementation of the OBA functions.

The server side plugins can make intensive use of the ontologies loaded by the server and return the computed results back to the client. The round-trips between client and server are reduced to a minimum and the logic is encapsulated in a reusable plugin. This new features enables computational biologists to use the basic information from ontologies in their applications, who would otherwise avoid ontologies.

ACKNOWLEDGMENTS

This work was supported by the Seventh Framework Program of the EU-funded "LipidomicNet" (grant no. 202272). We also acknowledge the support by Deutsche Forschungsgemeinschaft and Open Access Publication Funds of Goettingen University. The authors wish to acknowledge the help of Pia Franziska Kohlbecker with linguistic corrections.

REFERENCES

Adamusiak, T., Burdett, T., Kurbatova, N., van der Velde, K. J., Abeygunawardena, N., Antonakaki, D., et al. (2011). OntoCAT – simple ontology search and integration in Java, R and REST/JavaScript. *BMC Bioinformatics* 12, 218. doi: 10.1186/1471-2105-12-218

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., et al. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.* 25, 25–29.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Sci. Am.* 284, 34–43.

Bucher, G., Scholten, J., and Klingler, M. (2002)). Parental RNAi in *Tribolium* (Coleoptera). *Curr. Biol.* 12, R85–R86.

Burger, A., Davidson, D., and Baldock, R. (eds). (2008). *Anatomy Ontologies for Bioinformatics.* New York: Springer.

Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., Lewis, S., et al. (2009). AmiGO: online access to ontology and annotation data. *Bioinformatics* 25, 288–289.

Côté, R. G., Jones, P., Martens, L., Apweiler, R., and Hermjakob, H. (2008). The Ontology Lookup Service: more data and better tools for controlled vocabulary queries. *Nucleic Acids Res.* 36, W372–W376.

Day-Richter, J., Harris, M. A., Haendel, M., Gene Ontology OBO-Edit Working Group, and Lewis, S. (2007). OBO-Edit – an ontology editor for biologists. *Bioinformatics* 23, 2198–2200.

Dönitz, J., Goemann, B., Lizé, M., Michael, H., Sasse, N., Wingender, E., et al. (2008). EndoNet: an information resource about regulatory networks of cell-to-cell communication. *Nucleic Acids Res.* 36, D689–D694.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Ph.D. thesis, University of California, Irvine.

Heinemeyer, T., Chen, X., Karas, H., Kel, A. E., Kel, O. V., Liebich, I., et al. (1999). Expanding the TRANSFAC database towards an expert system of regulatory molecular mechanisms. *Nucleic Acids Res.* 27, 318–322.

Horridge, M., and Bechhofer, S. (2011). The OWL API: a Java API for OWL ontologies. *Semant. Web* 2, 11–21.

Huang, D. W., Sherman, B. T., and Lempicki, R. A. (2009). Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.* 37, 1–13.

Kinsella, R. J., Kähäri, A., Haider, S., Zamora, J., Proctor, G., Spudich, G., et al. (2011). Ensembl BioMarts: a hub for data retrieval across taxonomic space. *Database (Oxford)* 2011, bar030.

Kurbatova, N., Adamusiak, T., Kurnosov, P., Swertz, M. A., and Kapushesky, M. (2011). ontoCAT: an R package for ontology traversal and search. *Bioinformatics* 27, 2468–2470.

Lacy, L. W. (2005). *OWL: Representing Information Using the Web Ontology Language.* Victoria: Trafford Publishing.

Michael, H., Chen, X., Fricke, E., Haubrock, M., Ricanek, R., and Wingender, E. (2005). Deriving an ontology for human gene expression sources from the CYTOMER database on human organs and cell types. *In Silico Biol.* 5, 61–66.

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., et al. (2009). BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res.* 37, W170–W173.

Schröder, R., Beermann, A., Wittkopp, N., and Lutz, R. (2008). From development to biodiversity–Tribolium castaneum, an insect model organism for short germband development. *Dev. Genes Evol.* 218, 119–126.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., et al. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.* 25, 1251–1255.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., et al. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. U.S.A.* 102, 15545–15550.

Tomoyasu, Y., and Denell, R. E. (2004). Larval RNAi in *Tribolium* (Coleoptera) for analyzing adult development. *Dev. Genes Evol.* 214, 575–578.

Uhlen, M., Oksvold, P., Fagerberg, L., Lundberg, E., Jonasson, K., Forsberg, M., et al. (2010). Towards a knowledge-based Human Protein Atlas. *Nat. Biotechnol.* 28, 1248–1250.

Washington, N. L., Haendel, M. A., Mungall, C. J., Ashburner, M., Westerfield, M., and Lewis, S. E. (2009). Linking human diseases to animal models using ontology-based phenotype annotation. *PLoS Biol.* 7, e1000247. doi: 10.1371/journal.pbio.1000247

Winston, M. E., Chaffin, R., and Herrmann, D. (1987). A taxonomy of part-whole relations. *Cogn. Sci.* 11, 417–444.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 28 March 2012; accepted: 14 September 2012; published online: 05 October 2012.

Citation: Dönitz J and Wingender E (2012) The ontology-based answers (OBA) service: a connector for embedded usage of ontologies in applications. Front. Gene. 3:197. doi: 10.3389/fgene.2012.00197

This article was submitted to Frontiers in Bioinformatics and Computational Biology, a specialty of Frontiers in Genetics.

3.4 iBeetle-Base

The following manuscript is published as:

Dönitz J, Schmitt-Engel C, Grossmann D, Gerischer L, Tech M, Schoppmeier M, Klingler M and Bucher G.: **iBeetle-Base: a database for RNAi phenotypes in the red flour beetle Tribolium castaneum.** *Nucleic Acids Res.* 2015 Jan;43(Database issue):D720-5.

Authorships:

- Jürgen Dönitz designed the architecture and implemented most of iBeetle-Base as well as composed major parts of the manuscript.
- Christian Schmitt-Engel planned the layout of the web site and the general concept.
- Daniela Grossmann supported the writing of the manuscript.
- Lizy Gerischer processed the genomic data and prepared them for the import to iBeetle-Base.
- Maike Tech participated in the implementation during the beginning of the project,
- Michael Schoppmeier & Martin Klinger supported the project with the general concept and feedback.
- Gregor Bucher led the iBeetle project, supported the writing of the manuscript and contributed the introduction.

D720–D725 Nucleic Acids Research, 2015, Vol. 43, Database issue
doi: 10.1093/nar/gku1054
Published online 05 November 2014

iBeetle-Base: a database for RNAi phenotypes in the red flour beetle *Tribolium castaneum*

Jürgen Dönitz[1,*], Christian Schmitt-Engel[1,3], Daniela Grossmann[1], Lizzy Gerischer[4], Maike Tech[2], Michael Schoppmeier[3], Martin Klingler[3] and Gregor Bucher[1]

[1]Johann-Friedrich-Blumenbach Institute of Zoology and Anthropology, GZMB, Department of Evolutionary Developmental Genetics, Georg-August-University Göttingen, 37075 Göttingen, Germany, [2]Institute of Microbiology and Genetics, Department of Bioinformatics, Georg-August-University Göttingen, 37073 Göttingen, Germany, [3]Department of Biology, Friedrich-Alexander University, 91058 Erlangen, Germany and [4]Institute for Mathematics and Computer Science, Ernst Moritz Arndt University, 17487 Greifswald, Germany

Received August 25, 2014; Revised September 29, 2014; Accepted October 14, 2014

ABSTRACT

The iBeetle-Base (http://ibeetle-base.uni-goettingen.de) makes available annotations of RNAi phenotypes, which were gathered in a large scale RNAi screen in the red flour beetle *Tribolium castaneum* (iBeetle screen). In addition, it provides access to sequence information and links for all *Tribolium castaneum* genes. The iBeetle-Base contains the annotations of phenotypes of several thousands of genes knocked down during embryonic and metamorphic epidermis and muscle development in addition to phenotypes linked to oogenesis and stink gland biology. The phenotypes are described according to the EQM (entity, quality, modifier) system using controlled vocabularies and the *Tribolium* morphological ontology (TrOn). Furthermore, images linked to the respective annotations are provided. The data are searchable either for specific phenotypes using a complex 'search for morphological defects' or a 'quick search' for gene names and IDs. The red flour beetle *Tribolium castaneum* has become an important model system for insect functional genetics and is a representative of the most species rich taxon, the Coleoptera, which comprise several devastating pests. It is used for studying insect typical development, the evolution of development and for research on metabolism and pest control. Besides *Drosophila*, *Tribolium* is the first insect model organism where large scale unbiased screens have been performed.

INTRODUCTION

Next generation sequencing has been used for the identification of gene sequences in a plethora of insect species. Hence, our view on the gene complements present in insects is becoming ever more comprehensive. However, our knowledge on the function of these genes lags far behind because functional studies have long been restricted to organisms with a highly sophisticated genetic tool kit and a short generation time. These are the prerequisites for forward genetic screens, which have been used to identify gene function by large scale mutagenesis. So far, within insects, only the fruit fly *Drosophila melanogaster* has offered the possibility of saturated forward genetic screens and, hence, most of what we know about insect gene functions is derived from this species (1,2).

Recently, novel model species have become amenable to reverse genetic screens. Specifically, RNA interference (RNAi) has been widely used in a broad range of insects for knocking down gene function by injecting respective double stranded RNA (dsRNA) (3–6). So far, research has been focusing on genes, which were likely to be involved in a certain process based on *Drosophila* data like studies of gap gene (7–9) or pair rule gene orthologs (10) or based on vertebrates like studies on the Wnt pathway or head development (11–13). This candidate gene approach has been very fruitful but is biased towards highly conserved gene functions.

The iBeetle-Screen has been performed in order to overcome this bias and to identify unexpected gene functions. In this large scale RNAi screen, more than approximately one-third of the 16 505 genes were picked at random for knockdown allowing the detection of unexpected gene functions. The red flour beetle *Tribolium castaneum* has been selected for this endeavor because it has a strong and systemic RNAi response, which is transferred from the mother to its offspring (14). Further, in many respects *Tribolium* shows a more insect typical development than the fruit fly *Drosophila*. This includes axis formation, segmentation in an elongating germ band, head and leg development and oogenesis (15,16). Also metamorphosis reflects a more in-

*To whom correspondence should be addressed. Tel: +49 551 395426; Fax: +49 551 395416; Email: contact@ibeetle-base.uni-goettingen.de

sect typical situation in that most larval cells are re-used for the adult epidermis instead of being replaced by imaginal cells (17). Furthermore, *Tribolium* is used as model for cuticle synthesis (18), hormonal control (19) and for as yet unstudied processes like stink gland biology (20).

The phenotypes listed in the iBeetle-Base will help scientists in the *Tribolium* community to identify novel genes involved in the process under study. For scientists from other fields, it will be a complementary source of information for putative functions of a given gene. While iBeetle-Base makes available gene-related information (like RNAi phenotypes, sequences and orthologs), the BeetleBase (hosted at Kansas State University, USA) (21) offers access to extensive genomic data with respect to *Tribolium castaneum* and related beetles. Together, these interlinked databases will foster research in *Tribolium* and other insects.

DATA SOURCE AND ANNOTATION

dsRNAs targeting several thousand genes were injected into female pupae and the resulting RNAi phenotypes were scored. This included phenotypes of the injected animal like morphological phenotypes arising during metamorphosis or sterility. The offspring of the injected animals was scored for defects in the muscles and the first instar larval cuticle. In parallel, the same dsRNAs were injected into 5th/6th instar larvae and resulting cuticle and muscle phenotypes arising during metamorphosis were annotated along with other aspects of metamorphic development.

The iBeetle Screen was designed as a first pass screen, i.e. the experiments were not repeated and the annotation policy was to avoid false negatives with the trade-off of elevated false positive annotations.

Phenotype annotation using the EQM system

For the annotation of phenotypic changes the EQM (entity, quality, modifier) system (22–24) was previously developed and implemented in several phenotypic databases (25,26). This system defines the use of controlled vocabularies for the affected entity E (e.g. *leg*), the quality of the change Q (e.g. *number*) and the nature of the change using a modifier M (e.g. *increased*). A larva in which more legs have developed after RNAi treatment was annotated with 'larval_leg; number; increased'. Adhering to the EQM system has several advantages: First, it facilitates a systematic annotation of phenotypes. Second, it allows reading the descriptions of the phenotypes in phrases, which are well readable by both humans and machines. Third, the use of the same system in different phenotypic databases will facilitate data interconnection (23).

Structuring morphological information by the anatomical ontology TrOn

The *Tribolium* ontology (TrOn) (27) defines unique terms for the morphological entities of *Tribolium castaneum* (e.g. prothoracic_leg for the legs on the first thoracic segment). An ontology class comprises one such term together with its definition and `is_a` as well as `part_of` relations to other classes. TrOn covers all structures that were scored in the iBeetle screen. The annotation policy for the iBeetle screen was to annotate the most specifically affected (sub-) structure while the relations annotated in TrOn allow finding these phenotypes in searches using more general terms. For instance, an annotated defect in the specific structure 'pretarsus' (the most distal part of the larval leg) is found in a general search for 'leg' because TrOn stores the information that pretarsus along with coxa, tibia, tarsus, claw and other structures constitute a leg.

In TrOn all anatomical structures are modeled independently of a specific developmental stage (e.g. 'leg', which actually exists in different forms in embryos, larvae and adults). This category is called the *generic subset* and reflects the abstract anatomical concept of a certain structure. As such, it comprises many diverse realizations of that concept at different developmental stages (e.g. larva versus adult) or locations in different segments (e.g. fore- and hindwings) within one taxon or even in different species. A second category of ontology classes represents the visible occurrence of the respective structure at specific developmental stages (e.g. 'larval_prothoracic_leg' or 'adult_leg'). The classes of this category, the *concrete subset*, reflect an anatomical entity at a specific developmental stage, i.e. a unique dissectible structure. The concrete subset comprises child terms of the generic classes. For example, the anatomical concept wing is expressed in TrOn with the *generic class* 'wing'. Its children comprise *concrete classes* like 'adult_wing' or 'pupal_wing'. Within the iBeetle-Base the annotated phenotypes are always linked to concrete classes while the generic classes can be used in the search.

The OBA service (ontology based answers) (28) was used to process the ontology and to integrate it in iBeetle-Base. The service was used to assign the ontology classes to the right subset and to transiently add the inverse relation `has_part` to the annotated `part_of` relation. During the search it is the task of the OBA service to retrieve the relevant downstream substructures (concrete classes) of the morphological structure the user selected (generic class).

THE WEB INTERFACE

Search for phenotypic data

In order to identify all genes required for the formation of a certain structure a search for morphological phenotypes is implemented on the start page of iBeetle-Base (Figure 1). Typically the user would specify the developmental stage and the morphological structure of interest. iBeetle-Base provides two functions to enable the user to pick the right morphological term. First, the input field has an autocomplete feature that suggests terms or synonyms represented in TrOn based on the typed characters (Figure 1A, arrow a). Second, a click on the three dots at the end of the input field (Figure 1A, arrow b) opens a dialog (Figure 1B) where all terms used in the screen are displayed and can be selected. The numbers following these terms indicate the number of annotations connected to these structures (independently of the selected developmental stage or other selections). An important additional search criterion is the penetrance with which a phenotype was detected during the screen. High penetrance phenotypes (>80%) are less likely to be false positives than low penetrance phenotypes

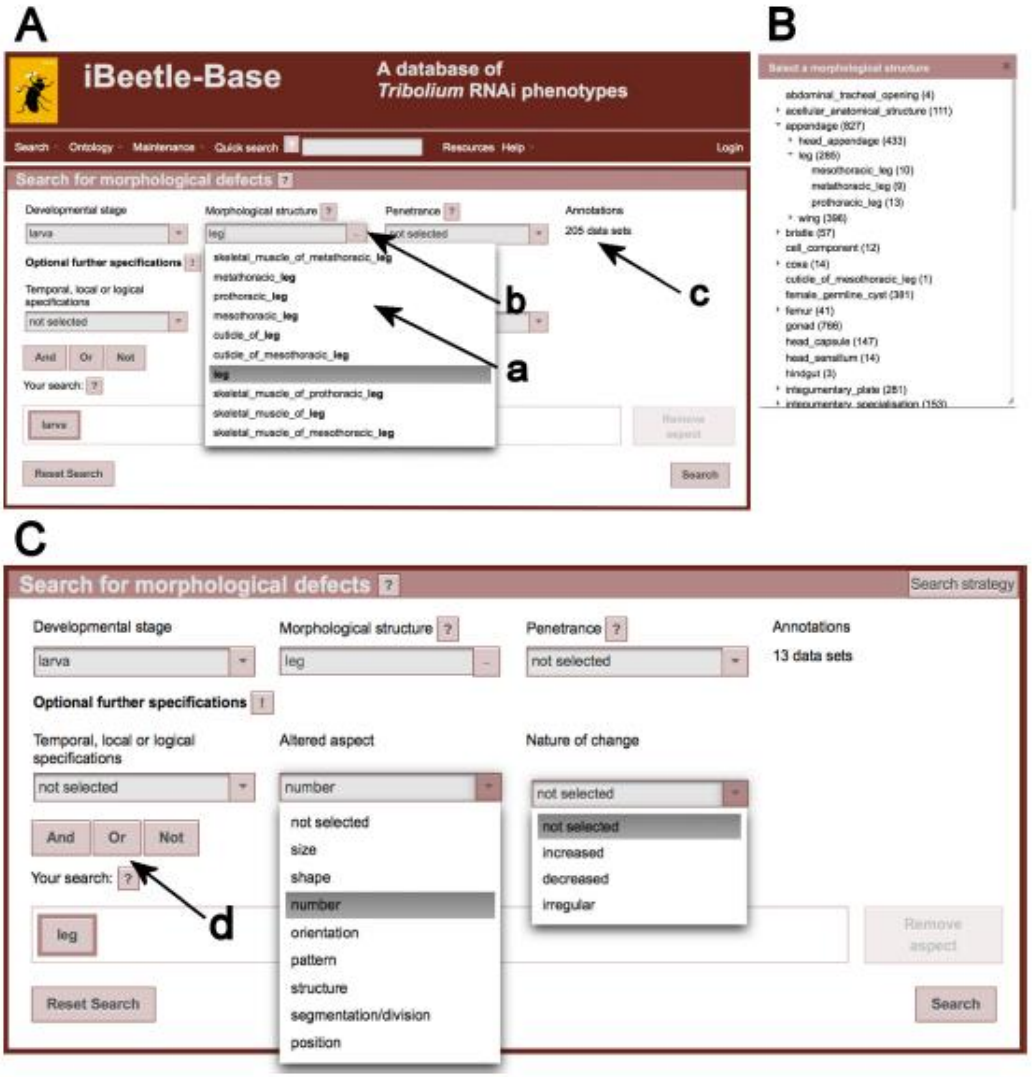

Figure 1. The start page of iBeetle Base. **(A)** An overview of the start page, where the user can either perform a quick search for a gene, ID's or a morphological structure of interest. A search for data sets affecting the leg is shown. To facilitate the search the input field suggests terms or synonyms based on the typed characters (black arrow a). **(B)** Clicks on the button at the end of the input field (black arrow b) show all terms used in the screen providing an alternative way to find terms. **(C)** A detailed view of different drop-down menus is shown ('Temporal, local or logical specifications', 'Altered aspect' and 'Nature of change'), which further specify the nature of the phenotype. The respective lists are reduced to the terms that are used or combined in the screen. Arrow d points to the buttons, which enables the user to combine two or more search aspects.

(<30%). The number of data sets fulfilling the selected criteria are displayed right of the penetrance dropdown menu (Figure 1A, arrow c). In addition to this search for affected structures, also the nature of the phenotype can be specified by using the dropdown menus for 'Temporal, local or logical specifications', 'Altered aspect' and 'Nature of change'. The content of these lists is reduced to those terms that result in at least one hit when combined with the previous selections (Figure 1C). For instance, after 'number' was selected as 'Altered aspect' the list of 'Nature of change' has been reduced from 37 terms to three. At the same time, the number of data sets that have been found has been updated from 205 (Figure 1A, arrow c) to 13 (Figure 1C).

Complex queries can be assembled by combining several search aspects with logical operators. After the search criterion for the first aspect is entered in the search form, a new aspect can be added by choosing one of the buttons 'And', 'Or', 'Not' (Figure 1, arrow d). In the 'Your search' field all aspects are shown and can be selected for editing the search form.

To identify most relevant phenotypes with low number of false positives, it is recommended to start with a rather broad search for an affected anatomical entity at a certain stage and a penetrance >50%. Any further restriction will make the hits more specific but will also lead to an increase of false negatives.

Clicking the search button will retrieve an overview of the search results (Figure 2). In the resulting table all annotations fitting the search criteria are listed along with the EQM annotations (Figure 2, arrow a) and respective penetrance (Figure 2, arrow b). Previews of the images documenting the respective annotations can be displayed in different sizes. Further information on the gene and additional phenotypes can be found upon click on the details page (see below). Based on these pieces of information the user can deselect data sets (Figure 2, arrow c) and a consolidated list is provided upon clicking on 'Refresh'. The resulting list can be exported or saved (Figure 2, arrow d).

For further information or optimization of the search strategy, the number of annotations retrieved for a specific morphological term is displayed by expanding the panel 'Search terms count' (Figure 2, arrow e).

Quick search for genes

A complementary way of using iBeetle-Base is a search for the functions of a given gene. To this end, gene identifier (i.e. TC number) or *Drosophila* ortholog (e.g. CG number or name) are entered in the 'Quick search' field in the menu bar (Figure 1A). If the search term is non-ambiguous the user is redirected to the corresponding details page (Supplementary Material, Figure S1). If several data sets fit the search term, a list is displayed where the user is able to choose the preferred ones.

The quick search can also be used to directly jump to specific data sets using the iBeetle identifier (e.g. iB_01757). For searches in the quick search, the prefix 'iB_' can be omitted.

Details pages for comprehensive gene and phenotypic information

The details pages of iBeetle-Base display information about a given gene and the iBeetle RNAi phenotypes associated with it (Supplementary Figure S1). The top section (Gene information) (Supplementary Figure S1A) provides access to gene and protein sequence data of the respective *Tribolium* gene. Furthermore, it provides links to the BeetleBase (21) genome browser where the genomic structure and location are displayed. A link to OrthoDB (29) allows the identification of orthologs of the respective gene. Where appropriate, information about the *Drosophila* ortholog is provided by a link to FlyBase (30).

If a certain gene was included in the iBeetle screen, the sequence of the respective dsRNA fragment is shown below of the gene information (Supplementary Figure S1B) ('iBeetle fragment') along with the RNAi phenotypes elicited with this iBeetle fragment. The observed phenotypes are presented with the annotations in the EQM scheme, pictures, free text comments and technical remarks. The results of the two parts of the iBeetle screen (pupal injection and larval injection screens) are shown in separate sections (Figure 2 C and D).

The dsRNA sequences of the iBeetle screen were mapped to the current gene models after a major re-annotation effort (Stanke, personal communication). Due to the changes in the gene predictions the association between an iB ID and a gene may change. For example, if two genes covered with iB fragments were merged in the new annotation, there are now two iB IDs assigned to the same gene. In such cases, both sets of results are displayed below the same TC number (Supplementary Figure S1E) and the former TC number is given in the section 'Gene information' (Supplementary Figure S1A). An example is the gene TC032760 (also used for Supplementary Figure S1).

Links to morphological definitions provided by TrOn and additional resources

The *Tribolium* Ontology TrOn is not only utilized in the background to improve the search algorithm but can also be searched and browsed on the web interface. This allows the user to access the definitions of morphological structures and their relations, which will foster the use of common terms in the community. The classes of the ontology are displayed as a tree which shows the hierarchy of the `is_a` relations (Supplementary Material, Figure S2, left column). Alternatively, a term can specifically be searched for ('Search & control'). For a selected node in the tree additional information is shown like synonyms, the definition (including links to the respective source of information), or relations to other ontology classes from TrOn (Supplementary Figure S2, arrow a). In addition, each morphological term is linked to those details pages, which mention this term (Supplementary Figure S2, arrow b). Further, the number of links of this class and all of its downstream classes is given (Supplementary Figure S2, arrow c).

Statistics

Basic information about all the 16 505 *Tribolium* genes of the current gene set is available in iBeetle-Base. 5180 different dsRNAs have been injected leading to 39 156 EQM annotations, documented with 14 487 pictures. 1007 morphological terms and definitions provided by TrOn can be browsed in the iBeetle-Base. Two-hundred-and-seven of them are used in the screen to describe morphological structures.

Linking to and from other resources

All data are available without restriction. Only for saving individual searches, a login is required. iBeetle-Base is intended to become the main gene-centered information hub for *Tribolium* research. Therefore, it provides links to additional resources like blast, genome browser, a community listserver and a link collection to some *Tribolium* labs.

Besides the possibility to search and browse the data of iBeetle-Base, it is welcomed that other projects cross link to this data resource. To link to a gene, iBeetle ID or TrOn ID stable URL patterns are administered. In detail these are:

iBeetle IDs: http://ibeetle-base.uni-goettingen.de/details/iB_00000

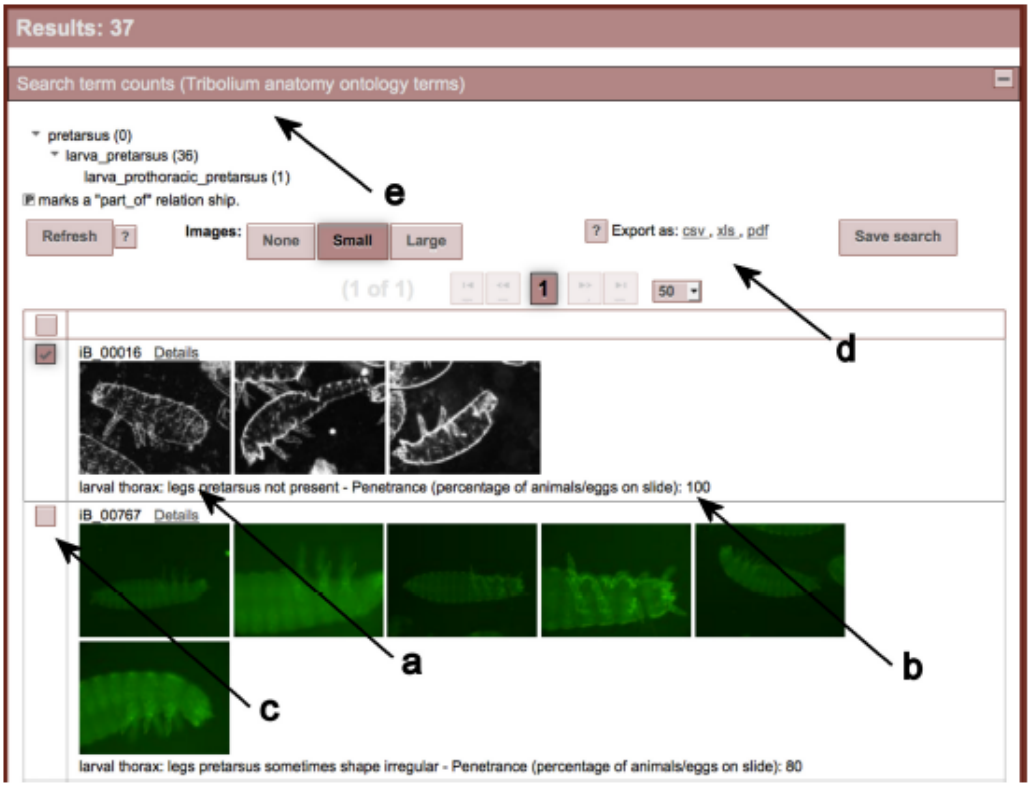

Figure 2. Partial view of the table with the search results. All research results are shown, which fulfil the selected criteria. In the shown example, the search for data sets affecting the pretarus leads to 37 results (not all data sets are shown). The results are listed including the EQM annotations (arrow a) and the respective penetrance (arrow b). Images can be displayed in different sizes. Arrow c points to the button, which allows the user to deselect specific data sets. The corresponding results list can be exported or saved (arrow d). Arrow e points to the search term counts, which includes all terms used for the semantic search. The numbers indicate how many annotations were retrieved for a specific morphological term.

TC numbers: http://ibeetle-base.uni-goettingen.de/details/TC000000
Ontology terms: http://ibeetle-base.uni-goettingen.de/ontology/tron/TrOn_0000000

FUTURE DIRECTIONS

iBeetle-Base is the first and currently only resource giving access to RNAi phenotypes gathered in an hypothesis independent way in any insect outside *Drosophila*. It is intended as a central resource for the increasing community working on aspects of the emerging model organism *Tribolium castaneum*. Beyond the *Tribolium* community, it will be used by scientists working on other insects in order to gather additional information on genes they are working on. To this end we plan to include links from FlyBase to iBeetle-Base. As part of the continuation of the iBeetle screen, additional data are currently being produced and will be added to the iBeetle-Base in the future. Finally, iBeetle-Base will be extended in a way to allow the community to contribute data. These data will be: complementary information to the already described phenotypes (e.g. confirmation), additional RNAi phenotypes (published or unpublished) and additional data, like gene expression data or literature.

iBeetle-Base is hosted at the computing center of the University of Göttingen (GWDG). The University of Göttingen and the GWDG will provide long term hosting and maintenance of iBeetle-Base as part of the eResearch initiative of the university.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENT

We thank the screeners of the first phase of the iBeetle project for their accurate and laborious work: Van Anh Dao, Daniela Grossmann, Upalparna Majumdar, Christian Schmitt-Engel, Dorothea Schultheis, Nadi Stroehlein, Jonas Schwirz, Nicole Troelenberg and Tobias Richter. Further, we thank Dr Mario Stanke for providing the new gene set prior to publication and Marita Büscher for her linguistic advice.

FUNDING

German Research Foundation (DFG) [BU1443/6-1]. Funding for open access charge: German Research Foundation; Open Access Publication Funds of the Göttingen University.
Conflict of interest statement. None declared.

REFERENCES

1. Nüsslein-Volhard,C. and Wieschaus,E. (1980) Mutations affecting segment number and polarity in *Drosophila*. *Nature*, **287**, 795–801.
2. St Johnston,D. (2002) The art and design of genetic screens: *Drosophila melanogaster*. *Nat. Rev. Genet.*, **3**, 176–188.
3. Brown,S.J., Mahaffey,J.P., Lorenzen,M.D., Denell,R.E. and Mahaffey,J.W. (1999) Using RNAi to investigate orthologous homeotic gene function during development of distantly related insects. *Evol. Dev.*, **1**, 11–15.
4. Lynch,J.A. and Desplan,C. (2006) A method for parental RNA interference in the wasp Nasonia vitripennis. *Nat. Protoc.*, **1**, 486–494.
5. Hughes,C.L. and Kaufman,T.C. (2000) RNAi analysis of Deformed, proboscipedia and Sex combs reduced in the milkweed bug Oncopeltus fasciatus: novel roles for Hox genes in the hemipteran head. *Development*, **127**, 3683–3694.
6. Miyawaki,K., Mito,T., Sarashina,I., Zhang,H., Shinmyo,Y., Ohuchi,H. and Noji,S. (2004) Involvement of Wingless/Armadillo signaling in the posterior sequential segmentation in the cricket, Gryllus bimaculatus (Orthoptera), as revealed by RNAi analysis. *Mech. Dev.*, **121**, 119–130.
7. Marques-Souza,H., Aranda,M. and Tautz,D. (2008) Delimiting the conserved features of hunchback function for the trunk organization of insects. *Dev. Camb. Engl.*, **135**, 881–888.
8. Cerny,A.C., Grossmann,D., Bucher,G. and Klingler,M. (2008) The *Tribolium* ortholog of knirps and knirps-related is crucial for head segmentation but plays a minor role during abdominal patterning. *Dev. Biol.*, **321**, 284–294.
9. Bucher,G. and Klingler,M. (2004) Divergent segmentation mechanism in the short germ insect *Tribolium* revealed by giant expression and function. *Dev. Camb. Engl.*, **131**, 1729–1740.
10. Choe,C.P., Miller,S.C. and Brown,S.J. (2006) A pair-rule gene circuit defines segments sequentially in the short-germ insect *Tribolium* castaneum. *Proc. Natl. Acad. Sci. U.S.A.*, **103**, 6560–6564.
11. Bolognesi,R., Farzana,L., Fischer,T.D. and Brown,S.J. (2008) Multiple Wnt genes are required for segmentation in the short-germ embryo of *Tribolium* castaneum. *Curr. Biol. CB*, **18**, 1624–1629.
12. Fu,J., Posnien,N., Bolognesi,R., Fischer,T.D., Rayl,P., Oberhofer,G., Kitzmann,P., Brown,S.J. and Bucher,G. (2012) Asymmetrically expressed axin required for anterior development in *Tribolium*. *Proc. Natl. Acad. Sci. U.S.A.*, **109**, 7782–7786.
13. Posnien,N., Koniszewski,N.D.B., Hein,H.J. and Bucher,G. (2011) Candidate gene screen in the red flour beetle *Tribolium* reveals six3 as ancient regulator of anterior median head and central complex development. *PLoS Genet.*, **7**, e1002418.
14. Bucher,G., Scholten,J. and Klingler,M. (2002) Parental RNAi in *Tribolium* (Coleoptera). *Curr. Biol.*, **12**, R85–R86.
15. Klingler,M. (2004) *Tribolium*. *Curr. Biol.*, **14**, R639–R640.
16. Schröder,R., Beermann,A., Wittkopp,N. and Lutz,R. (2008) From development to biodiversity—*Tribolium castaneum*, an insect model organism for short germband development. *Dev. Genes Evol.*, **218**, 119–126.
17. Snodgrass,R. (1954) *Insect Metamorphosis: Smithsonian Miscellaneous Collections*, V122, No. 9 Literary Licensing, Washington, DC.
18. Chaudhari,S.S., Arakane,Y., Specht,C.A., Moussian,B., Boyle,D.L., Park,Y., Kramer,K.J., Beeman,R.W. and Muthukrishnan,S. (2011) Knickkopf protein protects and organizes chitin in the newly synthesized insect exoskeleton. *Proc. Natl. Acad. Sci. U.S.A.*, **108**, 17028–17033.
19. Konopova,B. and Jindra,M. (2007) Juvenile hormone resistance gene Methoprene-tolerant controls entry into metamorphosis in the beetle *Tribolium castaneum*. *Proc. Natl. Acad. Sci. U.S.A.*, **104**, 10488–10493.
20. Li,J., Lehmann,S., Weißbecker,B., Naharros,I., Schütz,S., Joop,G. and Wimmer,E.A. (2013) Odoriferous Defensive Stink Gland Transcriptome to Identify Novel Genes Necessary for Quinone Synthesis in the Red Flour Beetle, *Tribolium* castaneum. *PLoS Genet*, **9**, e1003596.
21. Kim,H.S., Murphy,T., Xia,J., Caragea,D., Park,Y., Beeman,R.W., Lorenzen,M.D., Butcher,S., Manak,J.R. and Brown,S.J. (2010) BeetleBase in 2010: revisions to provide comprehensive genomic information for *Tribolium castaneum*. *Nucleic Acids Res.*, **38**, D437–D442.
22. Mungall,C.J., Gkoutos,G.V., Smith,C.L., Haendel,M.A., Lewis,S.E. and Ashburner,M. (2010) Integrating phenotype ontologies across multiple species. *Genome Biol.*, **11**, R2.
23. Washington,N.L., Haendel,M.A., Mungall,C.J., Ashburner,M., Westerfield,M. and Lewis,S.E. (2009) Linking human diseases to animal models using ontology-based phenotype annotation. *PLoS Biol.*, **7**, e1000247.
24. Beck,T., Morgan,H., Blake,A., Wells,S., Hancock,J.M. and Mallon,A.-M. (2009) Practical application of ontologies to annotate and analyse large scale raw mouse phenotype data. *BMC Bioinformatics*, **10**, S2.
25. Sprague,J., Bayraktaroglu,L., Bradford,Y., Conlin,T., Dunn,N., Fashena,D., Frazer,K., Haendel,M., Howe,D.G., Knight,J. *et al.* (2008) The Zebrafish Information Network: the zebrafish model organism database provides expanded support for genotypes and phenotypes. *Nucleic Acids Res.*, **36**, D768–D772.
26. Grumbling,G. and Strelets,V. (2006) FlyBase: anatomical data, images and queries. *Nucleic Acids Res.*, **34**, D484–D488.
27. Dönitz,J., Grossmann,D., Schild,I., Schmitt-Engel,C., Bradler,S., Prpic,N.-M. and Bucher,G. (2013) TrOn: an anatomical ontology for the beetle *Tribolium castaneum*. *PLoS One*, **8**, e70695.
28. Dönitz,J. and Wingender,E. (2012) The ontology-based answers (OBA) service: a connector for embedded usage of ontologies in applications. *Bioinforma. Comput. Biol.*, **3**, 197.
29. Waterhouse,R.M., Tegenfeldt,F., Li,J., Zdobnov,E.M. and Kriventseva,E.V. (2013) OrthoDB: a hierarchical catalog of animal, fungal and bacterial orthologs. *Nucleic Acids Res.*, **41**, D358–D365.
30. McQuilton,P., St Pierre,S.E., Thurmond,J. and FlyBase Consortium (2012) FlyBase 101–the basics of navigating FlyBase. *Nucleic Acids Res.*, **40**, D706–D714.

3.5 Other ontology enhanced applications

Aside of iBeetle-Base, there are more use cases of the OBA service to enhance applications.

3.5.1 EndoNet

Besides the well-studied intracellular signaling networks, intercellular networks are of crucial importance for multicellular organisms. The cells of a tissue have to be synchronized and complex processes such as the coordinated development of a whole organism have to be tightly orchestrated as they can involve entities ranging from single cells to complete organs.

EndoNet is an information resource about the endocrine network in human and its components, namely messengers (e.g. hormones), anatomical structures (e.g. cells, tissues, organs), and receptors (Potapov et al., 2006). These components are connected by events like binding, influence of the downstream secretion or the secretion itself to build up the intercellular signaling network. The anatomical entities were first linked to the Cytomer ontology (Dönitz et al., 2008). In medical terminology, several names are often attributed to anatomical structures. The OBA service is used on the public web page of EndoNet to add the superior and subordinated anatomical structures to the search query of the user as well as to the resulting detail pages. The search term “liver” will therefore also reveal data annotated to the hepatocyte cell type (Dönitz and Wingender, 2014).

Contributions:
The need for an information resource about the endocrine network was first recognized by Edgar Wingender; Ines Liebich drafted a first design of a relational database. Afterwards Anatolij Potapov was the responsible researcher and supervised the annotation. I developed the data model, implemented the webinterface, including the embedding of the Cytomer ontology and also took over the supervision of the annotation.

3.5.2 TFClass

Transcription factors play a major role in gene regulation by binding to a specific sequence pattern on the DNA. Transcription factors can be classified based on their DNA binding motifs and homology, which allows a hypothesis about their biological functions. A hierarchical classification of transcription factors and its proper visualization represent an essential support for research.

The transcription factor classification (TFClass) was modeled as separate ontologies for human and mouse, where the hierarchy was mapped to the class relations (Wingender et al., 2013, 2015). On the web page both classifications are synchronized during the browsing through the basic levels. (See also

Figure 2) The OBA service is used for a search function in the current selected classification. Another task fulfilled by OBA is the expansion of the tree, or parts of it, to a given level, e.g. family or genus. In a first step, all target classes of the required level are retrieved, either for a complete tree or the current sub-tree. In a second step for each target class the path from the root node to the respective class is queried. This data is added to the visualization and the tree is expanded. The semantic functions of OBA used for TFClass are all implemented by the generic part of OBA relying on the universal properties of ontologies. Besides a well-defined structure of the ontologies used as input, no project specific functions are needed.

Contributions:

The classification of the transcription factors was completely done by Edgar Wingender. The original data format is an html page, for the new web interface this page is parsed by Torsten Schöps and the data transferred into an OBO ontology following my specifications. I implemented the web interface including the connection to the OBA service.

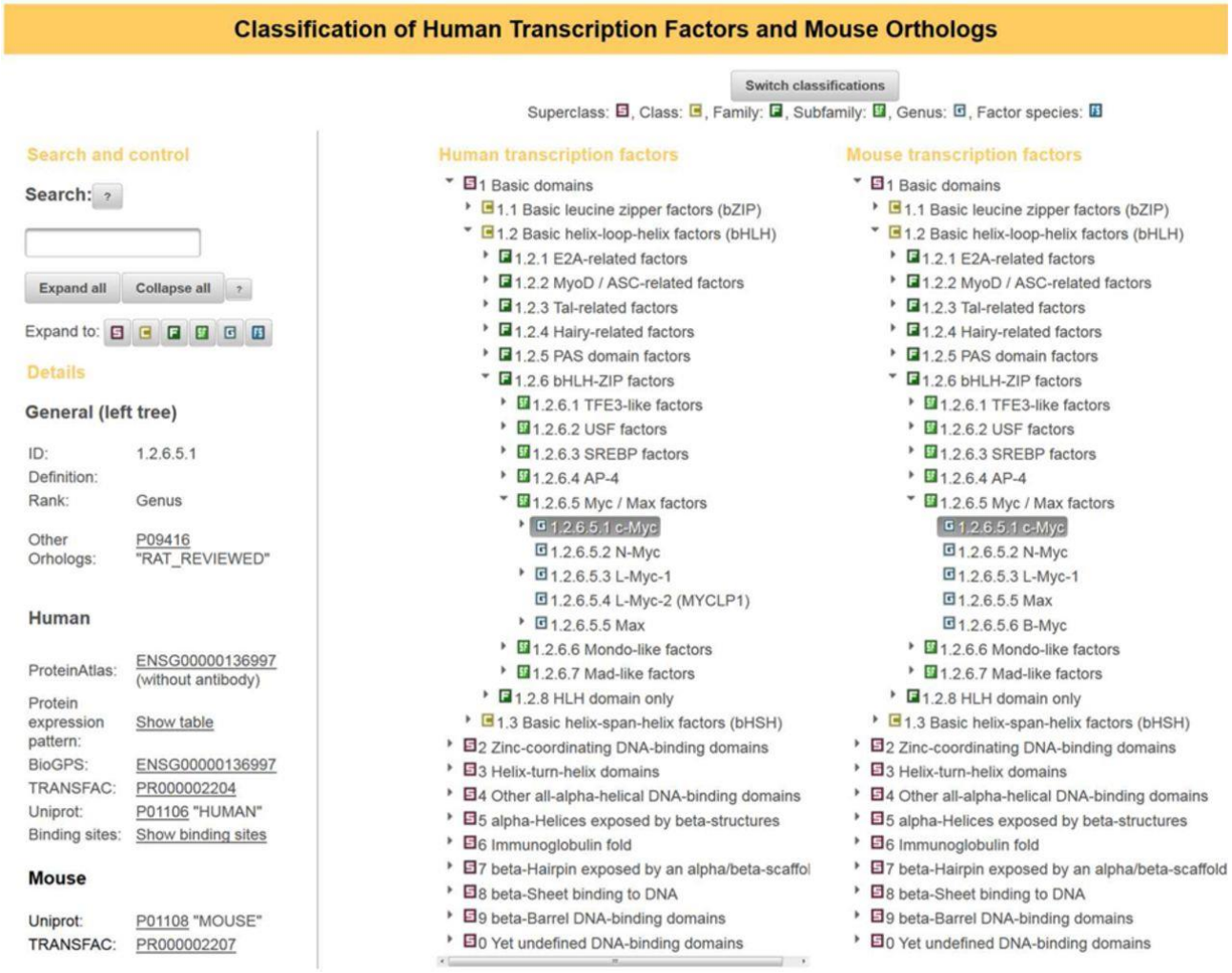

Figure 2

Screenshot of the web interface of TFClass: The main tree in the center is the classification of human TFs. Navigating to a TF in this tree automatically opens the orthologous factor in the tree for mouse on the right side. The panel on the left gives detailed information to the factor selected in the main tree and controls for search and expansion. The trees can be switched to make the mouse the main classification in the center.

The figure was taken from Wingender et al., 2015.

The web interface of TFClass is available at: http://tfclass.bioinf.med.uni-goettingen.de.

3.5.3 OntoScope

Ontologies are very well suitable to define and describe a knowledge domain to humans and machines. However, with several hundred or thousands of classes and a multitude of relations between them, the information has to be visualized in a comprehensible way. Every ontology can be represented as a graph, with the ontology classes as nodes and their relations as edges. Such visualization is advantageous as the user is free to choose the relation to follow for every node (as opposed to tree-like layouts). The class hierarchy has a strong relevance for the semantics of an ontology. For a human user, in particular for an advanced one, the class hierarchy if often not the most important subject, because it is mostly evident. Using an alternative relation between the nodes to build a tree has the drawback that the relation type of interest is depending on the context. For the hepatocyte cell the relation type *cellOf* and *partOf* might be of interest to browse to the organ liver. From here on the relation *derivesFrom* might be of more interest to get to the germinal sheet the liver originates from.

The ontology viewer OntoScope has been implemented to facilitate interactive exploration of an ontology. The ontology is represented as graph and only the nodes actively searched for or expanded by the user are displayed. The node can be expanded separately along each relation type, displayed as colored edges. OntoScope uses the OBA service as backend. The generic functions of the server are used for search, display and browsing. Small, project-specific plugins for OntoScope and OBA can enhance OntoScope with additional features. Cytomer nodes can be color-coded according to the physiological system the represented structure belongs to. In the case of iBeetle-Base, the node and its border are color-coded by developmental stage and type of the ontology class (generic, abstract, mixed). Figure 3 depicts an exemplary screenshot of OntoScope using the Cytomer ontology.

OntoScope can be used as standalone Java application to visualize arbitrary ontologies or as pre-configured Java applet to enhance the webpage of a project with the visualization of a specific ontology as demonstrated on iBeetle-Base.

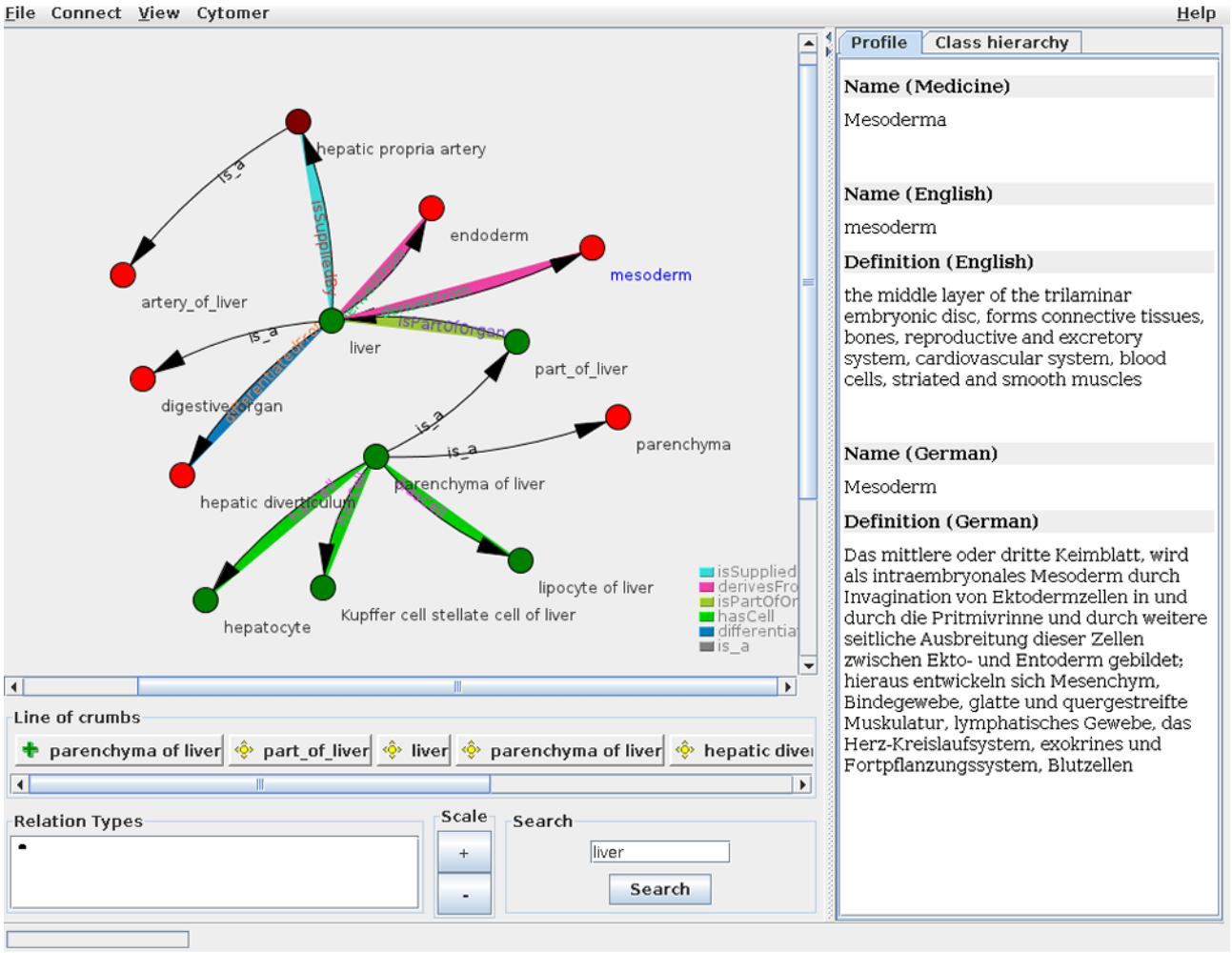

Figure 4

Screenshot of OntoScope displaying nodes of the Cytomer ontology: The screenshot shows the parts of OntoScope. In the main window the nodes of the ontology are visualized as graph. The color of the nodes can be mapped to a semantic property of the used ontology, in case of Cytomer they are colored according to their physiological system. Each relation type of the ontology is assigned to a color and the vertices are drawn in this color. For the currently selected ontology class detail information is shown in the panel on the right. At the bottom is an input box to search for nodes and the line of crumbs as history of the browsing activity (adding, expanding and deleting of nodes).

The OntoScope executable is available, together with the source code, at http://ontoscope.bioinf.med.uni-goettingen.de .

Contributions:

Remko Ricanek implemented a first version of OntoScope in his master thesis under a joined supervision of Holger Michael and myself. Later, I rewrote OntoScope, partially

with the support of Ralph Krimmel during a practical course. The framework for the graph visualization was updated to a new major version. I changed the methods for reading the data to use the OBA service and added new features, e.g. the “line of crumbs” as history for the executed commands of the user.

4. Discussion

4.1 Ontologies, the data source

4.1.1 The Tribolium Ontology TrOn

The ontology TrOn about the morphological structures of *Tribolium castaneum* is the first anatomical ontology about this emerging model organism. This adds another valuable resource to the collection of biomedical ontologies. However, in the opinion of the OBO Foundry community it is important that the need of a new resource is considered its creation, in particular it should be avoided that new ontologies cover the same knowledge domain as an existing one. No anatomical ontology exists for *Tribolium* prior to TrOn, but two ontologies should be considered to be used for the iBeetle project: the Coleoptera Anatomy Ontology (CalAO) (Nico Franz and Aaron Smith, personal communication) and the Fly ontology DAO. The beetle ontology aims to be valid for the species rich taxon of the Coleoptera. Huge efforts have to be done to define ontology classes that are true for all beetles. Therefore, the progress of the ontology was not far enough to be used in the iBeetle project and still the ontology is not publicly available. Also with the complete Coleoptera ontology *Tribolium* specific entities would have been missing and still would have to be added to a species-specific resource. The second option could have been to use the fly ontology for the annotation in the iBeetle screen and only add some *Tribolium* specific entities. Many projects and tools already rely on the DAO, a reuse of this ontology could for example facilitate a cross species search for phenotypes. However, *Drosophila* and *Tribolium* are different species and even homologous structures are quite different like the fore- and hindwings, which makes the existing definitions of the fly useless for the beetle. Importantly, using the same ontology for both species would contradict the convention that an ontology class represents an entity of a specific knowledge domain.

The option chosen for TrOn was to create a new ontology for *Tribolium*, based on the Common Anatomical Reference Ontology (CARO) (Haendel et al., 2008) and later on crosslink common entities with other ontologies like DAO or ColAO. CARO defines anatomical concepts like “multicellular tissue”, “anatomical line” or “epithelium” and should be reused in anatomical ontologies. For the corresponding entities in TrOn,

ontology classes with the same label and position in the hierarchy were created. The respective description was copied and linked to the related class in CARO to prove the origin and to cross link TrOn and CARO. An alternative way could be to import the CARO ontology into TrOn and to replace the basic TrOn ontology classes with their CARO equivalents. Below of the generic CARO classes, the *Tribolium* specific classes would then be child classes of the ones from the CARO ontology. This approach would have the advantage that the universal CARO ontology would be embedded in the *Tribolium* ontology instead of a crosslinking in the definitions. The entity "multicellular_tissue" would then be represented by a single ontology class instead of two interlinked classes in two namespaces. Both approaches are valid and commonly used. In order to strengthen the interconnection, ontology classes of different ontologies can be annotated with *sameAs* instead of adding a reference link in the description of the class. While this concept does not add additional information, it would enhance the option to automatically parse information.

However, the aim should be to achieve a maximum of crosslinking between TrOn and other ontologies. In TrOn the biological concepts (e.g. leg or coxa) are independent from dissectible and species-specific entities. These abstract classes are predestinated for linking to other species, while the child classes are *Tribolium* specifc (e.g. adult_leg or pupa_coxa). Unfortunately, the *Drosophila* ontology does not have a similar architecture. Most entities that are equivalent to the abstract classes in TrOn are part of the adult organism in DAO, but dissectible structures from two different species are not the same, only their biological concept. Non species-specific ontologies will be better candidates as they focus on abstract concepts, more equal to the abstract classes of TrOn.

Future extension of TrOn

TrOn was initiated for the iBeetle screen. Therefore the morphological structures covered by TrOn are limited to those that were in the focus of the screen. In order to fulfill the objective to be a common and central resource for the morphology of *Tribolium*, the ontology has to be extended with further structures. TrOn was already recognized in other projects as useful resource (Ramírez and Michalik, 2014; Tarasov and Génier, 2015). With an extension of the ontology's content it could be expected that more projects will use TrOn to annotate their data. The improved content would

facilitate TrOn to become the authoritative resource for the morphology of *Tribolium* and could prevent the community from conflicting definitions of terms as exposed by Yoder et al. for the paramere (Yoder et al., 2010). Currently, the Virtual Fly Brain project aims to define names and structures of the nervous system for all insects. *Drosophila* is the best-studied insect and therefore is the basis of the project (Ito et al., 2014). A consistent and thorough structure for all components of the nerve system would be a significant assistance for the research in *Tribolium*. The approach to create one resource for all insects will assist cross species research and data processing. However, *Tribolium* specific structures, names or synonyms should be considered, too. An ontology, particulary TrOn, is well suited to integrate the results of the Virtual Fly Brain in the *Tribolium* research and preserve the beetle specific parts. Common structures can be copied or imported into TrOn. *Tribolium* specific structures can be added by extending the common hierarchy, established names can be added as synonyms. The muscle system is another field to be solved for *Tribolium*. Currently, the majority of muscles in *Tribolium* are not defined and named yet. A central resource would support a consistent naming.

4.1.2 The Cytomer ontology

The importance of an ontology for the human anatomy ranges from interested layman over the clinical field to biomedical research. More than any other organism the human body is of interest of laymen, whether based on curiosity or to gain better knowledge about medical topics. Cytomer provides, in addition to the medical term, name, synonyms and description in English and German. The variety of relations entraps the user to explore the ontology with different questions in mind. In the clinical field Cytomer can be embedded into services and applications to bridge the different abstraction layers of anatomy and to profit from the information encoded in the semantic of the ontology. Examples for this are EndoNet, which queries related anatomical structures from the ontology, or a more complex function of the OBA service like “getOrgans”.

With the Foundational Model of Anatomy (FMA) (Rosse and Mejino, 2007) another ontology about the human anatomy was created. In principle, both ontologies have the same aim but are following different strategies and, hence, have different strengths and

weaknesses. On the first sight, the number of classes are eight times higher than the number of Cytomer. However, this can be explained by different annotation policies. Symmetric structures are modeled twice in FMA. Also parts of single structures are modeled individually like each belly and head of muscles. With respect to absolute numbers of relationships, FMA has nearly twice as many relations as Cytomer. But again the symmetric structures and part of structures were included twice, such that Cytomer actually has the higher information content in the relations of the ontology. The double amount of relation types in Cytomer also confirms this. Finally, the architecture followed different principles. While Cytomer uses multiple inheritance to model the different nature of anatomical structures (e.g. the biceps has muscle of arm, abductor, two headed muscle as parent), in FMA each ontology class has only a single parent class while the other classes are linked using relations. This approach prefers arbitrarily a single parent class. This hinders the semantic processing. Finally, although FMA also started to include embryonic structures, they are not as comprehensive as the modeling of the embryonic development in Cytomer. The extensive use of relation types and relations and the modeling of the embryonic structures make Cytomer an important option as an ontology for the human anatomy.

In 2003, the BRENDA Tissue Ontology (BTO) was launched (Gremse et al., 2011). With currently around 5,700 ontology classes, the ontology seems to be quite large, but it covers animals, plants and fungi as well as cell lines. Also aberrant anatomical structures like tumor cells are included. BTO uses the class hierarchy and three additional relation types (part_of, develops_from/derives_from and related_to). This is much less than in Cytomer and although BTO has the structure of a DAG, the semantics of BTO are not so far developed. The ontology class “heart” has a `part_of` relation to “cardiovascular system” which in turn is part of “whole body”, but the “heart” has no `is_a` relation and is therefore assumed to be directly below of the root node. A class for “organ” is missing at all. BTO was designed as encyclopedia for the enzyme information system BRENDA (Chang et al., 2015). It is useful as stable reference for the location of enzyme activities, but not so much as semantic resource.

4.2 OBA service, processing of ontologies

The OBA service is a useful link between ontologies and applications. While it is possible

to implement the algorithms to parse ontologies separately in every application, this contradicts two central design patterns “Don’t repeat yourself “ (DRY) (Hunt and Thomas, 1999) and the “separation of concerns” (Dijkstra, 1982). The algorithm for solving a certain task should be implemented only once. This is achieved by outsourcing it to a separate service, which is used by all other applications. The OBA service addresses both issues by encapsulation and provides the content of ontologies in a way that an application developer is able to use it without detailed knowledge of ontologies. Many applications only need a limited subset of the possibilities of ontologies but have the focus on other topics. OBA facilitates to implement aspects of ontology functionality without requiring the application developer to get familiar with the advanced internals of ontologies. There are already libraries, like the OWL-API, that provide functions to parse ontology files in different formats. However, in these libraries the complexity of the semantic of ontologies or the non-intuitive naming is not hidden from the application developer. Hence, detailed knowledge of ontologies and the annotation policy of specific ontologies are required to use them. OBA is an important tool to enable non-specialists to include ontologies in their projects.

The OBA service has already been implemented in several projects to solve different tasks. In the two web interfaces iBeetle-Base and EndoNet, OBA is utilized to improve the search function. For a search term on an abstract level, more concrete terms are fetched from the ontology to expand the search results with related hits.

To achieve ontology specific tasks, like to determine the developmental stage of an anatomical structure, the generic semantic functions of OBA could be easily extended by project-specific plugins. The plugin for iBeetle-Base implements how the link of a morphological structure to a developmental stage is annotated in TrOn. This allows combining a queried morphological structure with a developmental stage. The EndoNet plugin limits the up- and downstream search to a few predefined relation types of Cytomer and the anatomical structures available in EndoNet. In case of the classification of transcription factors (TFClass), OBA is not only used for the search function but also to retrieve the required information to expand or collapse the hierarchy to a level requested by the user.

Another example for implementation of OBA is OntoScope. This graphical viewer for ontologies represents an ontology as graph instead of the more common tree like

structure. The use of OBA in this project differs from that in the web interfaces. The search function is used to get an initial ontology class. Starting with this class interactive browsing along arbitrary relations is enabled. The required function of the OBA service is the transparent lazy loading of entities over the network from the server. Besides mapping ontologies to a graph structure, OntoScope uses ontology-specific functions from the OBA server to add information to the graph, e.g. by coloring the nodes. The OBA plugin for the Tribolium Ontology implements the knowledge about which relations it has to follow to determine the respective developmental stage of a given concrete class as well as the algorithm to assign the ontology classes to one of the three sets of concrete, abstract und mixed classes. This information is used to colorize the nodes and their borders. For the human anatomical ontology Cytomer, the nodes are colored representing the physiological system the represented structure is assigned to.

OntoScope in its interplay with the OBA service is a prime example to provide the user with information from the semantics encoded in the ontology instead of just displaying information from the currently selected ontology class. The graphical and intuitive browsing along the edges (i.e. relations of the ontology classes) invites the user to explore the content of ontologies, independent of his or her previous knowledge.

4.3 Biological applications enhanced by ontologies

In the iBeetle project a significant gap exists between the level of abstraction used for the annotation of morphological structures or for the search for them. Following good scientific procedure, phenotypes are annotated as precisely as possible. If only a part of the mouthparts are affected, only this part is annotated as affected structure. If further structures are involved (e.g. the labrum as well as the antenna), they are noted in separate annotation statements. Only if the head as a whole is affected, the term “head” will be found as annotation in the data. A typical search, in contrast, is on a more abstract level (e.g. “head” instead of its substructure “labrum”). A researcher may start a search with a general morphological term like head. Based on the ontology, the results will contain annotations for the “head” as well as for all its substructures like “labrum”. Without the usage of an ontology, dozens of terms for all parts would have to be combined. This is a laborious and error-prone approach. An alternative would be to add keywords for the search to each dataset manually. Such an approach is time consuming

and potentially biased.

Another useful option to be implemented in iBeetle-Base in the future would be to add an ontology that represents the nature of the phenotype. Several terms used for annotation (e.g. “shorter”, “shortened” or “wider”) describe an alteration of the size of a shape. Again, the level of detail of annotation and the search pattern will be different. For a search all phenotypes may be of interest where the size of a morphological structure is affected independently of the amount or an in- or decrease. With the PATO ontology (Phenotype and Trait Ontology) the required ontology already exists. To improve iBeetle-Base with another ontology, an ontology about phenotypes has to be chosen or created and the annotations of the iBeetle screen have to be mapped to the selected ontology. For the search function an integration of the ontology with the support of the OBA service would be feasible.

5. References

Adamusiak, T., Burdett, T., Kurbatova, N., Joeri van der Velde, K., Abeygunawardena, N., Antonakaki, D., et al. (2011). OntoCAT -- simple ontology search and integration in Java, R and REST/JavaScript. *BMC Bioinformatics* 12, 218. doi:10.1186/1471-2105-12-218.

Apache Home (2015). Apache Jena - Home. *Apache Jena*. Available at: https://jena.apache.org/ [Accessed August 14, 2015].

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Sci. Am.* 284, 34–43. doi:10.1038/scientificamerican0501-34.

Brown, S. J., Mahaffey, J. P., Lorenzen, M. D., Denell, R. E., and Mahaffey, J. W. (1999). Using RNAi to investigate orthologous homeotic gene function during development of distantly related insects. *Evol. Dev.* 1, 11–15. doi:10.1046/j.1525-142x.1999.99013.x.

Brown, S. J., Shippy, T. D., Miller, S., Bolognesi, R., Beeman, R. W., Lorenzen, M. D., et al. (2009). The Red Flour Beetle, Tribolium castaneum (Coleoptera): A Model for Studies of Development and Pest Biology. *Cold Spring Harb. Protoc.* 2009, pdb.emo126. doi:10.1101/pdb.emo126.

Bucher, G., Scholten, J., and Klingler, M. (2002). Parental RNAi in Tribolium (Coleoptera). *Curr. Biol.* 12, R85–R86. doi:10.1016/S0960-9822(02)00666-8.

Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., Lewis, S., et al. (2009). AmiGO: online access to ontology and annotation data. *Bioinformatics* 25, 288–289. doi:10.1093/bioinformatics/btn615.

Chang, A., Schomburg, I., Placzek, S., Jeske, L., Ulbrich, M., Xiao, M., et al. (2015). BRENDA in 2015: exciting developments in its 25th year of existence. *Nucleic Acids Res.* 43, D439–D446. doi:10.1093/nar/gku1068.

Clark-Hachtel, C. M., Linz, D. M., and Tomoyasu, Y. (2013). Insights into insect wing origin provided by functional analysis of vestigial in the red flour beetle, Tribolium castaneum. *Proc. Natl. Acad. Sci.* 110, 16951–16956. doi:10.1073/pnas.1304332110.

Conesa, A., and Götz, S. (2008). Blast2GO: A Comprehensive Suite for Functional Analysis in Plant Genomics. *Int. J. Plant Genomics* 2008. doi:10.1155/2008/619832.

Costa, M., Reeve, S., Grumbling, G., and Osumi-Sutherland, D. (2013). The Drosophila anatomy ontology. *J. Biomed. Semant.* 4, 32. doi:10.1186/2041-1480-4-32.

Day-Richter, J., Harris, M. A., Haendel, M., The Gene Ontology OBO-Edit Working Group, and Lewis, S. (2007). OBO-Edit an ontology editor for biologists. *Bioinformatics* 23, 2198–2200. doi:10.1093/bioinformatics/btm112.

Dean, M., and Schreiber, G. (2003). OWL Web Ontology Language Reference. *OWL Web Ontol. Lang. Ref.* Available at: http://www.w3.org/TR/2003/WD-owl-ref-20030331/ [Accessed August 10, 2015].

Diestel, R. (2000). *Graph theory*. 2nd ed. New York: Springer.

Dijkstra, E. (1982). "On the Role of Scientific Thought," in *Selected Writings on Computing: A personal Perspective* Texts and Monographs in Computer Science. (Springer New York), 60–66. Available at: http://dx.doi.org/10.1007/978-1-4612-5695-3_12.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.* 1, 269–271. doi:10.1007/BF01386390.

Doms, A., and Schroeder, M. (2005). GoPubMed: exploring PubMed with the Gene Ontology. *Nucleic Acids Res.* 33, W783–786. doi:10.1093/nar/gki470.

Dönitz, J., Goemann, B., Lizé, M., Michael, H., Sasse, N., Wingender, E., et al. (2008). EndoNet: an information resource about regulatory networks of cell-to-cell communication†. *Nucleic Acids Res.* 36, D689–D694. doi:10.1093/nar/gkm940.

Dönitz, J., and Wingender, E. (2014). EndoNet: an information resource about the intercellular signaling network. *BMC Syst. Biol.* 8, 49. doi:10.1186/1752-0509-8-49.

Goclenius, R. (1613). Lexicon philosophicum, quo tanquam clave philosophiae fores aperiuntur. Informatum opera studio Rodolphi Goclenii senioris in Academia Marchioburgi philosophiae professoris. Francofurti.

Gremse, M., Chang, A., Schomburg, I., Grote, A., Scheer, M., Ebeling, C., et al. (2011). The BRENDA Tissue Ontology (BTO): the first all-integrating ontology of all organisms for enzyme sources. *Nucleic Acids Res.* 39, D507–513. doi:10.1093/nar/gkq968.

Haendel, M. A., Neuhaus, F., Osumi-Sutherland, D., Mabee, P. M., Mejino, J. L. V., Mungall, C. J., et al. (2008). "CARO - The Common Anatomy Reference Ontology," in *Anatomy Ontologies for Bioinformatics* (New York: Springer), 327 – 349.

Horridge, M., and Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. *Semantic Web*, 11–21. doi:10.3233/SW-2011-0025.

Huang, D. W., Sherman, B. T., Tan, Q., Kir, J., Liu, D., Bryant, D., et al. (2007). DAVID Bioinformatics Resources: expanded annotation database and novel algorithms to

better extract biology from large gene lists. *Nucleic Acids Res.* 35, W169–175. doi:10.1093/nar/gkm415.

Hunt, A., and Thomas, D. (1999). *The Pragmatic Programmer: From Journeyman to Master*. 1 edition. Reading, Mass: Addison-Wesley Professional.

Ito, K., Shinomiya, K., Ito, M., Armstrong, J. D., Boyan, G., Hartenstein, V., et al. (2014). A Systematic Nomenclature for the Insect Brain. *Neuron* 81, 755–765. doi:10.1016/j.neuron.2013.12.017.

Klingler, M. (2004). Tribolium. *Curr. Biol.* 14, R639–R640. doi:10.1016/j.cub.2004.08.004.

Kurbatova, N., Adamusiak, T., Kurnosov, P., Swertz, M. A., and Kapushesky, M. (2011). ontoCAT: an R package for ontology traversal and search. *Bioinformatics* 27, 2468–2470. doi:10.1093/bioinformatics/btr375.

Lassila, O., and Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. *Resour. Descr. Framew. RDF Model Syntax Specif.* Available at: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ [Accessed May 11, 2015].

Li, J., Lehmann, S., Weißbecker, B., Ojeda Naharros, I., Schütz, S., Joop, G., et al. (2013). Odoriferous Defensive Stink Gland Transcriptome to Identify Novel Genes Necessary for Quinone Synthesis in the Red Flour Beetle, Tribolium castaneum. *PLoS Genet* 9, e1003596. doi:10.1371/journal.pgen.1003596.

Matys, V., Fricke, E., Geffers, R., Gößling, E., Haubrock, M., Hehl, R., et al. (2003). TRANSFAC®: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res.* 31, 374–378. Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC165555/ [Accessed September 18, 2015].

Michael, H., Chen, X., Fricke, E., Haubrock, M., Ricanek, R., and Wingender, E. (2005). Deriving an ontology for human gene expression sources from the CYTOMER database on human organs and cell types. *In Silico Biol.* 5, 61–66.

Moore, E. F. (1959). The shortest path through a maze. 285–292.

Mungall, C., and Ireland, A. (2012). The OBO Flat File Format Guide, version 1.4. *OBO Flat File Format Guide Version 14*. Available at: http://oboformat.googlecode.com/svn/trunk/doc/GO.format.obo-1_4.html [Accessed August 21, 2015].

Mungall, C. J., Gkoutos, G. V., Smith, C. L., Haendel, M. A., Lewis, S. E., and Ashburner, M. (2010). Integrating phenotype ontologies across multiple species. *Genome Biol.* 11, R2.

doi:10.1186/gb-2010-11-1-r2.

Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., et al. (2009). BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res.* 37, W170–173. doi:10.1093/nar/gkp440.

O'Rahilly, R., and Müller, F. (1987). Developmental Stages in Human Embryos: Including a Revision of Streeter's Horizons and a Survey of the Carnegie Collection. Washington, D.C.: Carnegie Inst of Washington.

Palmer, J. (2012). "Parmenides," in *The Stanford Encyclopedia of Philosophy*, ed. E. N. Zalta Available at: http://plato.stanford.edu/archives/sum2012/entries/parmenides/.

Potapov, A., Liebich, I., Dönitz, J., Schwarzer, K., Sasse, N., Schoeps, T., et al. (2006). EndoNet: an information resource about endocrine networks. *Nucleic Acids Res.* 34, D540–545. doi:10.1093/nar/gkj121.

Ramírez, M. J., and Michalik, P. (2014). Calculating structural complexity in phylogenies using ancestral ontologies. *Cladistics* 30, 635–649. doi:10.1111/cla.12075.

Rosse, C., and Mejino, J. L. V. (2007). "The Foundational Model of Anatomy Ontology," in *Anatomy Ontologies for Bioinformatics: Principles and Practice*, eds. A. Burger, D. Davidson, and R. Baldock (Springer), 59–17. Available at: http://sigpubs.biostr.washington.edu/archive/00000204/.

dos Santos, G., Schroeder, A. J., Goodman, J. L., Strelets, V. B., Crosby, M. A., Thurmond, J., et al. (2015). FlyBase: introduction of the Drosophila melanogaster Release 6 reference genome assembly and large-scale migration of genome annotations. *Nucleic Acids Res.* 43, D690–D697. doi:10.1093/nar/gku1099.

Schmitt-Engel, C., Schultheis, D., Schwirz, J., Ströhlein, N., Troelenberg, N., Majumdar, U., et al. (2015). The iBeetle large-scale RNAi screen reveals gene functions for insect development and physiology. *Nat. Commun.* 6, 7822. doi:10.1038/ncomms8822.

Schröder, R., Beermann, A., Wittkopp, N., and Lutz, R. (2008). From development to biodiversity—Tribolium castaneum, an insect model organism for short germband development. *Dev. Genes Evol.* 218, 119–126. doi:10.1007/s00427-008-0214-3.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., et al. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol* 25, 1251–5. Available at: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=17989687.

Sokoloff, A. (1972). The biology of Tribolium: with special emphasis on genetic aspects. Oxford: Clarendon Press.

Streeter, G. L., and Corner, G. W. (2013). *Developmental Horizons in Human Embryos: Age Groups XI to XXIII*. Literary Licensing, LLC.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., et al. (2005). Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. U. S. A.* 102, 15545–15550. doi:10.1073/pnas.0506580102.

Tarasov, S., and Génier, F. (2015). Innovative Bayesian and Parsimony Phylogeny of Dung Beetles (Coleoptera, Scarabaeidae, Scarabaeinae) Enhanced by Ontology-Based Partitioning of Morphological Characters. *PLoS ONE* 10, e0116671. doi:10.1371/journal.pone.0116671.

Tarjan, R. (1972). Depth first search and linear graph algorithms. *SIAM J. Comput.*

The Gene Ontology Consortium (2015). Gene Ontology Consortium: going forward. *Nucleic Acids Res.* 43, D1049–D1056. doi:10.1093/nar/gku1179.

The W3C SPARQL Working Group (2013). SPARQL 1.1 Overview. *SPARQL 11 Overv.* Available at: http://www.w3.org/TR/sparql11-overview/ [Accessed August 31, 2015].

Varzi, A. (2015). "Mereology," in *The Stanford Encyclopedia of Philosophy*, ed. E. N. Zalta Available at: http://plato.stanford.edu/archives/sum2015/entries/mereology/.

Wingender, E., Schoeps, T., and Dönitz, J. (2013). TFClass: an expandable hierarchical classification of human transcription factors. *Nucleic Acids Res.* 41, D165–170. doi:10.1093/nar/gks1123.

Wingender, E., Schoeps, T., Haubrock, M., and Dönitz, J. (2015). TFClass: a classification of human transcription factors and their rodent orthologs. *Nucleic Acids Res.* 43, D97–102. doi:10.1093/nar/gku1064.

Yoder, M. J., Mikó, I., Seltmann, K. C., Bertone, M. A., and Deans, A. R. (2010). A Gross Anatomy Ontology for Hymenoptera. *PLoS ONE* 5, e15991. doi:10.1371/journal.pone.0015991.

Zhang, B., Kirov, S., and Snoddy, J. (2005). WebGestalt: an integrated system for exploring gene sets in various biological contexts. *Nucleic Acids Res.* 33, W741–748. doi:10.1093/nar/gki475.

6. Acknowledgements

First of all, I would like to thank my two supervisors Prof. Dr. Edgar Wingender and Prof. Dr. Gregor Bucher for their great support and the possibility to work on this thesis. They helped me with vivid discussions and by giving me the possibility to work on interesting projects. Edgar Wingender gave me the room and inspiration to develop my own ideas and to improve my skills in bioinformatics. Gregor Bucher gave my work an additional input, motivated and encouraged me to finish this thesis.

Important parts of this thesis would have not been possible without the manual annotation of data. I appreciate very much the annotation of the Tribolium Ontology by Daniela Grossmann and her teaching me about Tribolium. Many hard-working students, thanks to them all, have annotated EndoNet and Cytomer from literature.

Further, I thank all members of the Institute of Bioinformatics and the Department of Developmental Biology, particularly lab 3, for the good working atmosphere. It was always a pleasure to work in these two groups. Scientific assistance and non-scientific discussion .I thank Julia Ulrich und Susanne Al-Eryani for constant motivation and helpful proofreading.

Finally, I would like to thank my family. They enabled me to achieve my goals and are always there for me.

Publikationen

Large scale RNAi screen in Tribolium reveals novel target genes for pest control and the proteasome as prime target.
Ulrich J, Dao VA, Majumdar U, Schmitt-Engel C, Schwirz J, Schultheis D, Ströhlein N, Troelenberg N, Grossmann D, Richter T, Dönitz J, Gerischer L, Leboulle G, Vilcinskas A, Stanke M and Bucher G.
BMC Genomics. 2015 Sep 3;16:674.

The iBeetle large-scale RNAi screen reveals gene functions for insect development and physiology.
Schmitt-Engel C, Schultheis D, Schwirz J, Ströhlein N, Troelenberg N, Majumdar U, Dao VA, Grossmann D, Richter T, Tech M, Dönitz J, Gerischer L, Theis M, Schild I, Trauner J, Koniszewski ND, Küster E, Kittelmann S, Hu Y, Lehmann S, Siemanowski J, Ulrich J, Panfilio KA, Schröder R, Morgenstern B, Stanke M, Buchhholz F, Frasch M, Roth S, Wimmer EA, Schoppmeier M, Klingler M and Bucher G.
Nat Commun. 2015 Jul 28;6:7822

iBeetle-Base: a database for RNAi phenotypes in the red flour beetle Tribolium castaneum.
Dönitz J, Schmitt-Engel C, Grossmann D, Gerischer L, Tech M, Schoppmeier M, Klingler M and Bucher G.
Nucleic Acids Res. 2015 Jan;43(Database issue):D720-5.

TFClass: a classification of human transcription factors and their rodent orthologs.
Wingender E, Schoeps T, Haubrock M and Dönitz J.
Nucleic Acids Res. 2015 Jan;43(Database issue):D97-102.

EndoNet: an information resource about the intercellular signaling network.
Dönitz J and Wingender E.
BMC Syst Biol. 2014 Apr 24;8:49.

TrOn: an anatomical ontology for the beetle Tribolium castaneum.
Dönitz J, Grossmann D, Schild I, Schmitt-Engel C, Bradler S, Prpic NM and Bucher G.
PLoS One. 2013 Jul 30;8(7):e70695.

TFClass: an expandable hierarchical classification of human transcription factors.
Wingender E, Schoeps T and Dönitz J.
Nucleic Acids Res. 2013 Jan;41(Database issue):D165-70

The ontology-based answers (OBA) service: A connector for embedded usage of ontologies in applications.
Dönitz, J. and Wingender, E.
Front. Gene. 3, 197 (2012).

Planar cell movements and oriented cell division during early primitive streak formation in the mammalian embryo.
Halacheva, V., Fuchs, M., Dönitz, J., Reupke, T., Püschel, B. and Viebahn, C.
Dev. Dyn. 240, 1905-1916 (2011).

EndoNet: An information resource about regulatory networks of cell-to-cell communication.
Dönitz, J., Goemann, B., Lizé, M., Michael, H., Sasse, N., Wingender, E. and Potapov, A. P.
Nucleic Acids Res. 36, D689-D694 (2008).

DEEP--A tool for differential expression effector prediction.
Degenhardt, J., Haubrock, M., Dönitz, J., Wingender, E. and Crass, T.
Nucleic Acids Res. 35, W619-W624 (2007)

EndoNet: An information resource about endocrine networks.
Potapov, A., Liebich, I., Dönitz, J., Schwarzer, K., Sasse, N., Schoeps, T., Crass, T. and Wingender, E.
Nucleic Acids Res. 34, D540-D545 (2006).