ICBO: International Conference on Biomedical Ontology
July 28-30, 2011 · Buffalo, NY, USA

Developing an Application Ontology for Biomedical Resource Annotation and Retrieval: Challenges and Lessons Learned

Carlo Torniai[1], Matthew Brush[1], Nicole Vasilevsky [1], Erik Segerdell[1], Melanie Wilson[1], Tenille Johnson[2], Karen Corday[2], Chris Shaffer[1], Melissa Haendel[1]

[1]Oregon Health & Science University, Portland, OR, USA
[2]Harvard Medical School, Boston, MA, USA

Abstract. The eagle-i project has been developing a semantic search portal for biomedical research resources. A unique feature of eagle-i is that the data collection and search tools are completely driven by ontologies. This has been a source of challenges and opportunities regarding use of biomedical ontologies in real-world applications. In this paper, we address our approach and lessons learned for balancing practical project requirements for design and implementation of an ontology driven application, with a desire to conform to best practices for biomedical ontology development.

Keywords: biomedical ontologies**,** application ontology, resource, eagle-i, ontology reuse.

1 Introduction

An important challenge in biomedical research is the ability to find relevant scientific resources such as reagents, instruments, and protocols, thereby reducing time-consuming and expensive duplication of resource development. For resources that are publicly available, information connecting them to relevant organisms, genotypes, genes, site of action, and other key biological search facets is frequently not available within public databases or from company catalogs. Furthermore, there exist numerous resources that are not published in journals or listed on websites. The eagle-i Consortium (www.eagle-i.org/home) aims to help researchers find biomedical research resources. The goal of eagle-i is to collect data about these "invisible" resources from the labs that use them, and make this information available through a semantic search portal. eagle-i also aims to be interoperable with and contribute to similar ontology-based efforts to represent publicly available research resources in repositories such as the Neuroscience Information Framework (NIF) [1] and the Resource Discovery System (RDS) [2]. These efforts have the main benefit of allowing linkage to very many data sets for gene function, expression, phenotypes, biological pathways, etc. Use of interoperable ontologies will enable understanding of the experimental context in which these data are collected, and support new hypothesis generation.

The architecture of the eagle-i system includes four main components: institutional triple-store repositories; a federated network; a data collection tool, and a central search application. In order to support semantic retrieval of resource data, the underlying data model is based on a modular set of ontologies. A unique feature of the project is that the user interface and logic of both the data collection and search tools are driven by ontologies, allowing these applications to seamlessly change in response to data-driven ontology enhancements [3]. In this paper, we present our approach and lessons learned in the process of developing the eagle-i ontology modules in compliance with project requirements, best practices for biomedical ontology development, and interoperability with other ontology-based resource systems and community ontologies.

2 Modeling Approach

Our modeling approach had three main drivers. The first was to represent real data collected about resources. The second was to have the ontology control the user-interface (UI) and the

logic of the data collection tool and search application. The third was a commitment to build a set of ontologies that could be reusable and interoperable with other ontologies and existing efforts for representing biomedical entities. This latter requirement translated into decisions to a) follow OBO Foundry [4] principles and best practices for biomedical ontology development and b) engage in active discussions within the bio-ontology community in order to provide context for eagle-i interoperability and align with domain-wide standards for resource representation (http://bit.ly/rrcoord).

We began our modeling effort by collecting a preliminary set of data with the goal of identifying key properties for each resource type collected by eagle-i. These include reagents, instruments, services, model and non-model organisms, protocols, biospecimens, human studies, and research opportunities. We then asked the eagle-i team to identify a set of queries relevant to each of these resources. For example, "Which laboratories in the United States are equipped with high-resolution ultrasound machines for brachial artery reactivity testing (BART)?" or "Find *in situ* hybridization protocols for whole-mount preparations of Aplysia." Over 300 queries were generated and analyzed to first identify their relevance and semantic linkage, and then to specify the relations required to answer these queries. Using the preliminary data and the analysis of the queries, we defined a preliminary high-level data model. Next we identified a set of classes and properties from existent biomedical ontologies that could be reused to implement this model, as well as those that had to be created *de novo*. Based on data and functional requirements gathered throughout the project, we expanded on this initial ontology in an iterative approach that involved collaboration with NIF, RDS, Ontology for Biomedical Investigation (OBI) [5], and VIVO [6]. Adherence to our three drivers presented interesting challenges and trade-offs when it came to implementation, which is discussed in the next section.

3 Implementation

3.1 Ontology Reuse and Development Practices

We implemented the eagle-i ontology modules in the Web Ontology Language (OWL) [7] to comply with the *de facto* standard for ontology representation and to exploit its reasoning capabilities. Upon analysis of existing ontologies, we came to the conclusion that those showing the most promising high-level classes and design principles for resource representation belonged primarily to the OBO Foundry constellation. Because of our choice to reuse portions of certain OBO ontologies (OBI [5], Uberon (http://bit.ly/ubernat), the Gene Ontology (GO; http://www.geneontology.org), the Software Ontology (SWO; http://www.ebi.ac.uk/efo/swo), the NIF Standard Ontology (NIFstd), Biomedical Resource Ontology (BRO) [2], etc.), we chose to follow several key OBO Foundry principles for ontology development and reuse. These included:

(a) Adoption of the Basic Formal Ontology (BFO) [8] as upper level ontology and the Information Artifact Ontology (IAO) [9] for representing ontology metadata.

(b) Use of the Relation Ontology (RO) [10] for basic properties.

(c) Adherence to the Minimum Information to Reference an External Ontology Term (MIREOT) [11] principle to reference terms and axioms already defined in other ontologies. MIREOT is a standard whereby a subset of classes and related axioms can be referenced from an ontology, without importing the whole source ontology.

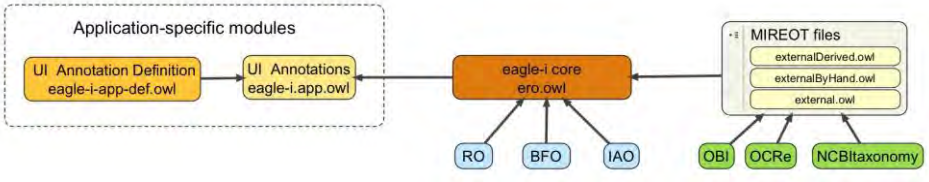

Figure 1. The eagle-i layered, modular ontology structure.
The core module imports BFO, IAO, and RO in their entirety, as well as files containing portions of external ontologies (MIREOT; shown are portions of OBI, Ontology of Clinical Research (OCRe), and the NCBI taxonomy). The application-specific modules define properties, instance values for annotation properties, and property and class annotations used by the eagle-i applications.

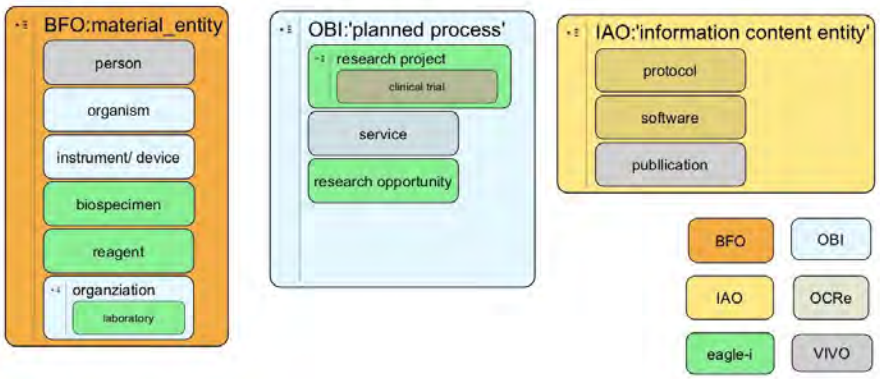

Figure 2. Research resources represented in eagle-i.
The classes from external ontologies referenced through MIREOT are as follows: 507 from OBI, 182 from NCBI Taxon, 53 from SWO, 20 from VIVO, 19 from OCRe, 13 from BRO.

3.2 Layered eagle-i Ontology Modules

We used an approach in which the representation of research resource data is decoupled from the representation of application-specific data used to control the appearance and behavior of the tooling UI. The result is a collection of ontology modules that are encoded and maintained separately, but assemble into a single ontology to support the eagle-i data collection tool and search application (Fig. 1).

The eagle-i core Ontology. The eagle-i core module contains classes and properties used to represent, logically define, and retrieve biomedical research resources (Fig. 2). The eagle-i core has its own unique namespace identifier (ERO) as required by the OBO Foundry. The eagle-i core module imports some external ontologies in their entirety, as well as a set of files containing individual classes and properties collected from external ontologies using MIREOT (Fig. 2). As of April 2011, the eagle-i core module contains 1059 ERO classes (excluding classes imported directly or referenced with MIREOT), 56 object properties, and 59 data properties. The current version of the core ontology under development is available at http://bit.ly/eagle-i-onto.

The application-specific modules drive application functionality. The eagle-i application-specific modules contain all the properties and classes required to drive the UIs of the data collection tool and the search application. These are primarily annotation properties that tell the data and search tools how to display and interact with the ontology classes and properties to which they are attached.

The basic design principle is to define a set of annotation properties and possible instance values for these properties in a 'UI Annotation Definition file' (eagle-i-app-def.owl). For example, the '*inClassGroup*' and '*inPropertyGroup*' annotation properties are used to tag specific classes and properties, respectively, as exhibiting certain application-related features or behavior. Table 1 shows some of the possible instance values for the '*inClassGroup*'[1] and '*inPropertyGroup*' properties, Table 2 describes additional properties defined in the UI Annotation Definition file, and Fig. 4 illustrates how they control various aspects of the data collection tool UI. A second module, the 'UI Annotations file' (eagle-i-app.owl), holds the actual annotations made on core eagle-i classes and properties using these annotation values. These two application-specific modules have a different namespace than the core ontology, and class and property URIs are not numerical since they are not meant to be shared or reused.

[1] Throughout this text, *italics* are used to indicate a term denoting an ontology class, instance or property.

Instance Label	Description	Example
resource	Denotes classes for which instances are collected	'*instrument*', '*biospecimen*', '*protocol*'
referenced class	Denotes non resource classes that are used to populate drop down menus in the UI	'*technique*', '*disease*'
data model exclude	Denotes classes or properties that are not included in the model used for the data tool or the search tool UIs	BFO classes such '*continuant*' or '*occurrent*' or RO relations such '*precedes*' or '*is about*'
primary property	Denotes properties that will appear grouped in the first block of properties in the data tool and in the search result page	service restrictions and fees, resource description.
embedded class	Denotes a class for which instances can only be created in the context of an embedding class	'*antibody immunogen*' created within '*antibody*', '*construct insert*' created within '*plasmid*'
related lab	Denotes the properties that relate a resource to a laboratory	'*service provided by*', '*located in*'
admin data	Denotes classes or properties that are never displayed in the search results	personal email, facilities address, last name and first name of facility contact persons

Table 1. Sample values for *inPropertyGroup* and *inClassGroup* properties.

Property Label	Description	Example	Property Type
eagle-i preferred label	Defines the value of preferred label to display in the data collection tool and search UIs	Capitalized 'Organization' for OBI_0000245 ('*organization*')	Annotation Property
eagle-i preferred definition	Defines the value of preferred definition to be displayed in the data collection tool and search UIs	For OBI_0000245 ('*organization*'):"An entity that can play roles, has participants, and has a set of organizational rules. …."	Annotation Property
eagle-i domain constraint	Used to specify the domain of an imported property. Each annotation will contain the URI of one class.	Value set to "OBI_0000245" ('*organization*') for RO property '*location_of*'	Data Property
eagle-i range constraint	Used to specify the range of an imported property. Each annotation will contain the URI of one class.	Value set to "ERO_0000004" ('*instrument*') for RO property '*located_in*'	Data Property

Table 2. Additional properties defined in the UI Annotation Definition file.

The UI Annotations file has also been used to import external referenced classes that are used to populate drop-down menus in the data collection tool, such as MeSH terms for diseases. This file also contains shortcut relations between classes that in the core ontology are expressed using a more complex concatenation of properties to maintain full logical computability. For example, from an application standpoint we need to have a single property that relates a service to a core laboratory providing that service. OBI uses a composed relation built from two properties to make this association between an organization and a service it provides ('*organization*' '*bearer_of*' some '*service provider role*' and '*realized_by*' some '*service*'). The UI Annotations file replaces this complex statement with a single property linking a service to its provider ('*service provider*'

'*provides_service*' some '*service*') where '*service provider*' is defined as follows: [('*organization*' or '*Homo sapiens*') *and* ('*bearer_of*' some '*service provider role*')]. This need to simplify complex relation chains will be a common issue in using ontologies for data collection applications, and approaches like the ones suggested in [12] should be exploited.

4 Discussion

In this section we discuss implementation choices and lessons learned from our effort to build an ontology that fulfills standards for interoperability and reuse and also requirements imposed by the UI and logic for the applications. The application ontology developed enables the annotation of eagle-i resources at the level of granularity and semantic complexity required for answering the queries in our use cases.

4.1 Use of Best Practices for Biomedical Ontologies Development

Upper ontology. We used BFO in our implementation. Use of an upper ontology can facilitate the design and modeling process, and ease the reuse of existent ontologies based on the same upper ontology. Issues can arise, however, when driving application interfaces directly from BFO-constructed ontologies. For example, our end users don't want to see BFO classes such as '*continuant*', '*occurrent*' or '*processual entity*' in the UI. Accordingly, we used a '*Data Model Exclude*' annotation property to mask these BFO classes from appearing in the UI (See Table 1). While simple to implement, this solution requires the additional effort to create and maintain the annotations and to implement procedures and tools that can use them programmatically.

Another challenge was the fact that the domain and/or range of most RO properties are BFO classes. For instance, the RO property '*located_in*' has '*continuant*' as both its domain and range. We wanted to reuse RO properties, and at the same time present users with application-specific choices when filling values of properties in the eagle-i data collection tool, without forcing them to traverse all the sub-trees under '*continuant*'. To achieve this, we used annotation properties to "restrict" the domain and range of those properties for use within the data collection tool (see '*eagle-i domain restriction*' and '*eagle-i range restriction*' examples in Table 2). Finally, because we used properties to define the data fields that are shown for each resource type in the data collection tool UI, we had to exclude from the model those properties that were inherited from upper ontology classes. For example, '*transformation_of*' is inherited by all subclasses of '*continuant*', but it was not appropriate to display this field in our UI for most continuants, such as '*laboratory*' or '*instrument*'.

Reusing existing ontologies and vocabularies. As described in Section 3 we have extensively used the MIREOT principle to reuse terms from external ontologies. One of the drawbacks we have found in applying MIREOT is related to the overhead of implementation efforts (creating, maintaining and running scripts to synchronize terms) and the lack of well-defined standards for custom axioms imports. Tools like OntoFox [13] or OWL Module Extractor[2] are helpful, but an ideal solution would be to integrate these tools within an ontology editor such as Protégé[3].

Due to its wide usage for indexing related publications, we opted to import portions of MeSH to reference diseases. Development of best practices and better availability of standardized URIs for commonly used non-ontology based controlled vocabularies would enable better interoperability and reuse.

4.2 Layered eagle-i Ontology Modules

Our approach of decoupling application-specific from general purpose content through layered ontology modules has proved an effective means to drive an application UI while maintaining interoperability with external ontologies and data sources. Logically we wanted to separate our core ontology from the application-specific ontologies and therefore identify what was relevant to share with the community from what was specific to the needs of eagle-i. These layered modules also facilitated parallel development in a shared repository, as ontologists familiar with OWL constructs and functionality could manage

[2] http://owl.cs.manchester.ac.uk/modularity/
[3] protege.stanford.edu

eagle-i core development, while curators were able to concurrently add proper annotation values in the UI annotations file.

We have identified a set of requirements for designing modular ontologies that can bridge the gap between an application and domain-specific ontologies. These include: (a) application-specific labels and definitions; (b) exclusion of sets of classes and properties from the model used by the application; (c) restriction of domain and range for some imported properties; (d) definition of display order of object and data properties at class level. Figure 4 illustrates several of these features in the context of the eagle-i data annotation tool. Despite the effectiveness of this approach, it requires significant effort to keep the annotations current when the core module changes, and presents the risk of excessive proliferation of annotation properties and their instance values in attempts to simplify application coding complexity.

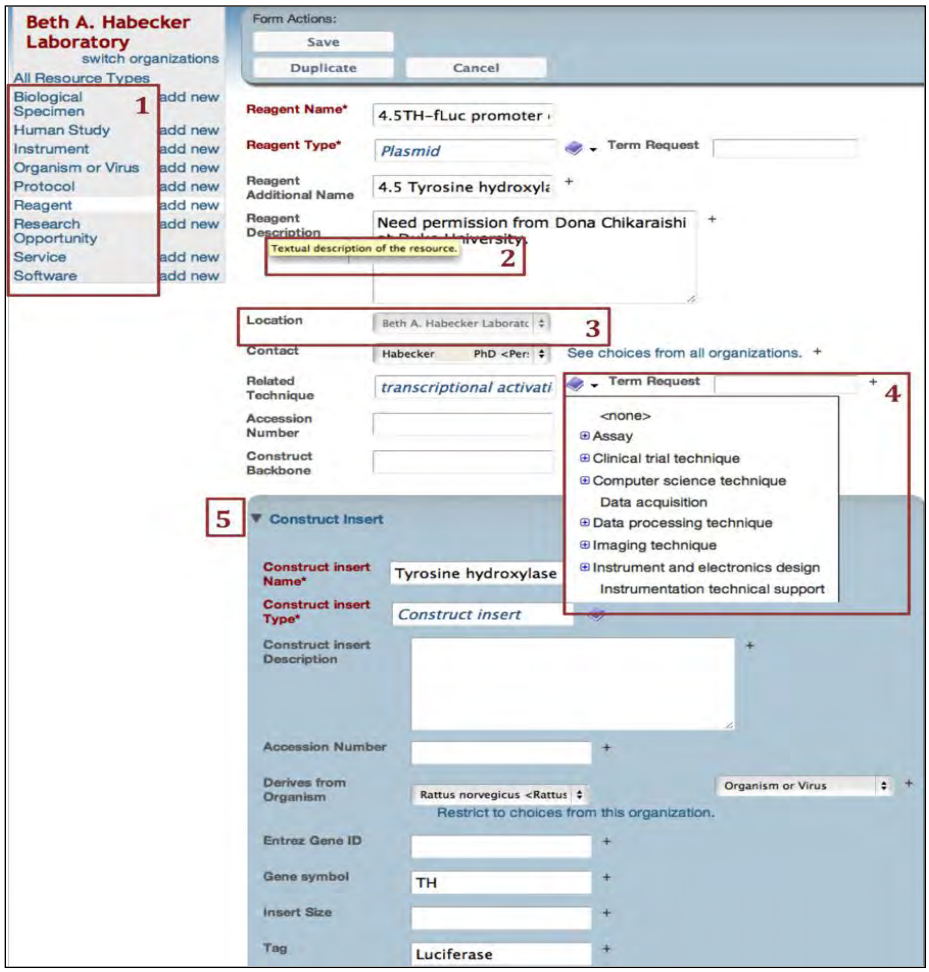

Figure 4. The eagle-i data collection tool user interface.

Shown is an example of a '*plasmid*' record annotated using the eagle-i ontology. (1) eagle-i classes annotated with the "resource root" value are displayed in the left bar menu. (2) The value of '*eagle-i preferred definition*' is used for tooltips that appear while hovering over the property labels. (3) The '*eagle-i preferred label*' is used for the display name of property. Here, the imported RO '*location_of*' has been renamed 'Location'. This property is also flagged as a primary property using the '*inPropertyGroup*' annotation property, as are '*Additional Name*', '*Description*' and '*Contact Person*' properties. This flag results in presentation at the top of the property list for a record. (4) Users can select a technique associated with the reagent. In the ontology, the '*technique*' class is annotated as a '*referenced class*' which tells the UI to allow reference to an ontology term but create no instances. (5) Construct insert is an example of a resource annotated as an '*embedded class*', which has to be created in the context of a construct or plasmid of which they are a part.

4.3 Coordination with the Biomedical Ontology Community

One of the most interesting aspects of the eagle-i "experiment" has been our commitment to collaboration with similar efforts aimed at resource modeling, data collection, and ontology development. We aligned our ontology with the NIF, VIVO, and RDS by reusing common terms and definitions, with the goal of migrating to usage of the same ontology classes and URIs at a later date. In doing so we have contributed to the design and enriched the content of existing biomedical ontologies – most notably OBI, where we contributed design patterns for services and added classes for devices, functions, and techniques. Finally, we developed ontological models for several important domains where the available ontologies are insufficient or non-existent, such as reagents, biospecimens, and genotype representation.

While the biomedical ontology community may benefit from these collaborations, it is clear that coordinated development practices require additional work for those involved in building or using ontologies to drive applications. It is not always straightforward to implement a consistent design within an application ontology that aligns and interoperates with external reference ontologies. For instance, eagle-i consulted with the OBI and NIF developer communities when modeling research-related services to ensure use of common principles and design patterns. However, the model that resulted, which classified services based on their input and output (e.g. data vs. a material entity), was not suitable for eagle-i users who preferred a hierarchy classified according to the process performed by the service (i.e. analysis, production, storage, etc). To accommodate both the broader biomedical ontology community and the eagle-i users, we implemented one service hierarchy in OBI, and then use MIREOT to reference individual service classes back into eagle-i and restructured them into a hierarchy that suited the needs of our application end-users.

It is preferable to maintain orthogonality of ontologies by having a single "home" for a particular kind of entity (like device in OBI or chemical reagent in CheBI) [4]. However, this goal requires substantial dedication because one first models in the application ontology where implementation can meet project requirements immediately. Next, term requests and/or modeling within an appropriate (reference) ontology are made, but this often takes a significant amount of time to coordinate agreement. Finally, the original model is obsoleted in the application ontology and classes from the reference ontology are referred using MIREOT. Additional effort is also required to keep terms synchronized.

The process of developing the eagle-i ontology modules has highlighted the need for effective automated mechanisms for extracting customized subsets of terms from reference ontologies. Such 'customizable community views' or 'slims' can be tailored to meet the needs of diverse communities or applications, and can minimize the effort involved in reusing existent ontologies [14]. Customized views can be extracted with different level of complexity (from referencing single terms through MIREOT, to importing a 'refactored' class hierarchy for a particular branch, or set of axioms to guarantee reasoning capabilities for a particular subset selected). This mechanism could also be used within an application ontology to provide a similar but more flexible implementation of our layered approach.

5 Conclusions

The process of developing an ontology-driven application has been an important benchmark for usage of biomedical ontologies and their driving design principles and tools. We have designed a layered set of modular ontologies, consisting of a broadly applicable core ontology and an application-specific ontology. This has allowed the identification of requirements and principles to inform a general design pattern for building applications that rely on ontologies for their logic and user interface.

We have identified a clear need to develop mechanisms, best practices and tools to bridge the gap between reference and application ontology development and usage. Future efforts will be aimed at refining and documenting these requirements, sharing our lessons learned, and engaging in efforts addressing the issues that the current biomedical ontology community is facing when dealing with real-world applications.

Acknowledgments

We acknowledge Ted Bashor, Rob Frost, Larry Stone and Daniela Bourges for their contributions to development of the eagle-i system.

References

1. Bug, W.J., et al.: The NIFSTD and BIRNLex vocabularies: building comprehensive ontologies for neuroscience. Neuroinformatics, 6 (3), 175-94 (2008)
2. Tenenbaum JD. et al. The Biomedical Resource Ontology (BRO) to enable resource discovery in clinical and translational research. J. Biomed. Inform. Feb;44(1):137-45 (2011)
3. Torniai, C., Bashor, T., Bourges-Waldegg, D., Corday, K., Frost, H.R., Johnson, T., Segerdell, E., Shaffer, C.J., Stone, L., Wilson, M.L., Haendel, M.A.: eagle-i: an ontology-driven framework for biomedical resource annotation and discovery. In: Bio-Ontologies 2010: Semantic Applications in Life Sciences, ISMB, Boston (2010)
4. Smith, B., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotechnol, 25 (11), 1251-5 (2007)
5. Peters, B. and The OBI Consortium: Ontology for Biomedical Investigations. In: International Conference on Biomedical Ontology, Buffalo, 2009
6. Krafft, D.B. et al.: VIVO: Enabling National Networking of Scientists. In Proceedings of the WebSci10, Raleigh, NC (2010)
7. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. In: Journal of Web Semantics 1, 7-26 (2003)
8. Grenon, P., Smith, B.: SNAP and SPAN: Towards Dynamic Spatial Ontology. In: Spatial Cognition & Computation: An Interdisciplinary Journal 4, 69-104 (2004)
9. Ruttenberg, A.: Information Artifact Ontology (IAO), http://code.google.com/p/information-artifact-ontology/
10. Smith, B. et al.: Relations in biomedical ontologies. In: Genome Biol 6, R46 (2005)
11. Courtot, M., Gibson, F., and Lister, A.: MIREOT: the Minimum Information to Reference an External Ontology Term. ICBO, Buffalo (2009)
12. Mungall CJ, Ruttenberg A, Osumi-Sutherland D.: Taking shortcuts with OWL using safe macros. Nature Precedings. 2010
13. Xiang Z, Courtot M, Brinkman RR, Ruttenberg A, He Y.: OntoFox: web-based support for ontology reuse. In: BMC Research Notes, 3:175 (2010)
14. Davis MJ, Sehgal MS, Ragan MA.: Automatic, context-specific generation of Gene Ontology slims. In: BMC Bioinformatics. 11:498. (2010)