Realizing the potential of reference ontologies for the semantic web

Submitted to NIH 1/16/07
Jim Brinkley (PI)
Incorporating suggestions and comments from Dan Cook, John Gennari, Cornelius Rosse, Dan Suciu, Onard Mejino, Todd Detwiler, Jim Bassingthwaighte, Mark Musen, Daniel Rubin, Natasha Noy, and Peggy Storey

This proposal is motivated by the need to integrate vast amounts of complex biomedical information into a holistic understanding of biological function, and to apply that understanding to improved health care. The semantic web (SW), which is envisioned as a distributed network of knowledge layered on the web, is emerging as an essential part of the solution to this need since it will help to ensure that diverse sources of data mean the same thing. A critical component of the SW will be networks of interlinked ontologies, each of which is a collection of entities and relationships describing a domain of interest. At present most ontologies can be called application ontologies because they are designed for specific applications. However, these ontologies do not scale up, they often overlap with each other, and they are very difficult to link together because of incompatible knowledge models. Reference ontologies, such as our Foundational Model of Anatomy (FMA), are an emerging ontology type that have the potential to provide a conceptual foundation for linking together these diverse application ontologies into the SW. However, because their scope is so large reference ontologies are highly complex, and are very difficult for application developers to use. It is the purpose of this proposal to make such reference ontologies easy to use by developing methods for deriving application ontologies from them, so that they may fullfull their potential as a foundation for the semantic web. We will do this by adapting and extending research from the database community in order to 1) create application ontologies as views over reference ontologies, and 2) embed these views as queryable web services. In developing these methods we will pursue the following specific aims, in collaboration with the National Center for Bioontology (CBio) and the UW computer science department: 1) investigate and develop view-based approaches for mapping between reference and application ontologies; 2) design and develop software tools that implement these approaches; and 3) develop graphical interfaces that allow end-users to specify these mappings. The tools we develop will be integrated as one component of the CBio BioPortal framework for accessing interlinked ontologies and data. Their development and evaluation will primarily be driven by the need to integrate data and computational models describing cardiac function.

The National Center for Biomedical Ontology (CBio) Mark Musen, PI.

This proposal is in response to PA-05-063, Collaborations with National Centers for Biomedical Computing. The Center we propose to collaborate with is the National Center for Biomedical Ontology (CBio), http://bioontology.org. This Center is administered by Stanford Medical Informatics, with Mark Musen as PI. Text taken verbatim from the Center overview (http://bioontology.org/Center_Overview.pdf) is quoted. Areas of Center interest and expertise most relevant to this proposal are highlighted in *italics*.

CBio is a collaboration among "biologists, clinicians, informaticists and ontologists who are working together to record, manage and disseminate biomedical information and knowledge in machine-processable form. The goals of the Center are: (1) *to help unify the divergent and isolated efforts in ontology development by promoting open-source standards-based tools to create, manage and use ontologies*, (2) to create new software tools to help scientists to use ontologies to annotate and analyze biomedical data, and (3) to provide a national resource for the ongoing evaluation, *integration* and evolution of biomedical ontologies and *associated tools* and theories in the context of driving biomedical projects."

"In the field computer science, an ontology is a representation of the entities, and the relationships among those entities, within a defined application domain. Ontologies are explicit models that drive modern information technology and thereby support the development of systems designed for purposes such as decision support, data integration, data annotation, and natural-language processing."

"The Center is developing two major repositories of biomedical content: (1) Open Biomedical Ontologies (OBO), a comprehensive, online library of open-content ontologies and controlled terminologies, and (2) Open Biomedical Data (OBD), a database resource that will allow expert scientists to archive experimental data that is fully described (annotated) using the OBO ontologies and terminologies. The biomedical research community will access OBO and OBD via a system called *BioPortal* – a web site and a suite of Web services that will enable both human users and computer-based agents to access the rich content that the Center and its collaborators will curate."

"The Center's ongoing technological development is driven by the core requirements of the Center's driving biological projects, which currently focus on needs (1) *to create, access, and browse relevant ontologies*, (2) to use ontologies to annotate large experimental data sets – both biological and clinical, and (3) to access (for example, via visualization tools) and to draw inferences from the annotated data. The Center seeks to acquire, curate, and disseminate electronic knowledge resources that can enhance the design and execution of experiments, experiment execution, data analysis, information synthesis, and hypothesis generation. The Center's goals are not only technological in nature; there is a pervasive desire to stimulate a cultural shift in the way that biomedical scientists think about online knowledge resources and the application of codified knowledge in their research."

Various members of the UW team have collaborated in the past with Center personnel. In these cases the UW team members have mostly been anatomist content-experts who were developing our Foundational Model of Anatomy (FMA) [1], a large reference ontology that is the basis for much of the proposed work (section C.1). The one exception is John Gennari, an informaticist/computer scientist who was one of the primary developers of Protégé while at Stanford [2] and is now on the faculty at UW informatics. However, the PI on this proposal (who is an informaticist and the director of the UW Structural Informatics Group, the originating group for this proposal) has never collaborated with any Center investigators, although we have discussed the possibility many times over the years. Dan Suciu, our UW computer science database expert, has also never collaborated with Center investigators. We view this proposal as an ideal opportunity to begin such an informatics/computer science-based collaboration.

Center Organization

"Like each of the seven National Centers for Biomedical Computing within the framework of the NIH Roadmap, the activities of the National Center for Biomedical Ontology are divided into seven core components." These cores include personnel from Stanford, Lawrence Berkeley Lab, Mayo Clinic, University of Victoria, University of Buffalo, University of Cambridge, University of Oregon, and University of California, San Francisco. The cores, and their primary locations are:

1. Computer science (Stanford, with participation from Mayo Clinic, University of Victoria and University of Buffalo)
2. Bioinformatics (Lawrence Berkeley lab, with participation from University of Victoria)
3. Driving Biological Projects
   a. Linking human mutations in drosophila to human disease (University of Cambridge, Michael Ashburner)
   b. Relating zebrafish phenotypes to human disease genes (University of Oregon, Monte Westerfield)
   c. Analyzing evidence in HIV clinical trials (University of California, San Franciso, Ida Sim)
4. Computing Infrastructure (Stanford)
5. Training (Stanford)
6. Dissemination (Stanford)
7. Management (Stanford)

The Computer Science Core (Core 1)
This proposal is a collaboration with personnel in Core 1, who are primarily interested in ontologies, and tools to manage, integrate and provide access to ontologies. Core 2 is primarily devoted to the use of ontologies to annotate and integrate data, which is not the subject of this proposal. However, our proposed work is motivated by the need to integrate data, and we in fact are involved in data integration through other funded work (see Preliminary Results), so we will keep informed of the progress in Core 2.

As noted in the overview, "Because efforts in ontology development have thus far been uncoordinated and because individual ontologies are spread widely across the Internet in various degrees of formal coherence, a critical research focus of the Center is to *unify and integrate ontologies and to raise the level of ontology design*…" **The primary goal of this proposal is to help the Center in this research focus**.

"To build the new OBO (Open Biomedical Ontologies) resource, the Center (Core 1) is performing research and development in the following key areas:

- Methods for indexing and categorizing ontologies
- *Methods for relating different ontologies and terminologies to one another and for creating mappings among them*
- Metadata to store information about ontology provenance, usability, and peer review
- Capabilities for evaluation, versioning, *searching, querying, and viewing ontologies*
- Methods for visualizing ontology libraries and their collective contents
- *Semantic Web technology* that will provide both human users and intelligent software agents with comprehensive access to the virtual library of ontologies in the new version of OBO."

"The Center is building BioPortal to make its technologies and resources available over the Internet to the diverse community of biomedical researchers using simple browser technology. The goal is to enable integrated development of ontology-based tools and distribution of those tools on a global scale, keeping pace with the rapid changes in ontology content. BioPortal adopts a tiered software architecture. At the lowest level, it will provide the infrastructure to unify the component terminologies and ontologies (OBO), ontology metadata, and annotations on experimental data (OBD). A middle layer of *core ontology services* – for *ontology management and integration, ontology visualization*, and peer-review will provide specific functionality for end users. BioPortal's interface layer will give users (and computer-based agents) seamless access to heterogeneous biomedical terminologies, *ontologies*, metadata about ontologies, annotations on experimental data, as well as access to the Center's *open-source visualization* and analysis *tools*."

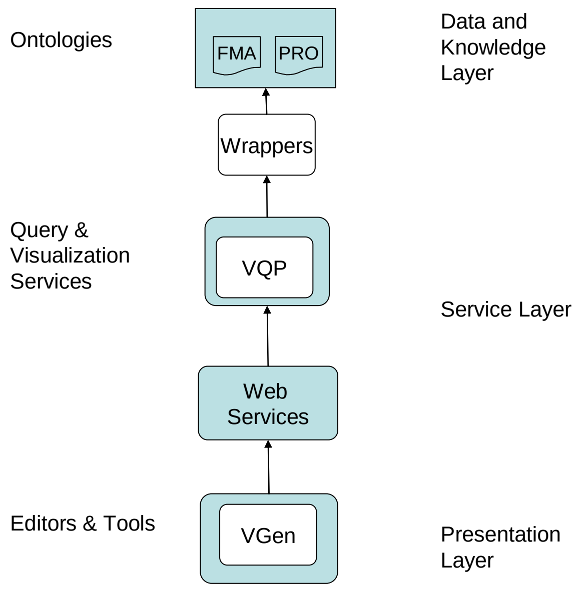

Figure 1 Integration of tools to be developed in BioPortal

Figure 1 is adapted from the Center's web site, and shows how the tools we develop will be integrated within the BioPortal architecture. As discussed in detail in the research design, we will develop a graphical tool called VGen (View Generator) for creating views of reference ontologies (specific aim 3). This tool will become part of the Editors and Tools Suite in the Presentation layer of BioPortal. The views we create will be queried (specific aim 2) by means of the VQP (View Query Processor), which will become part of the Query and Visualization Services web services in the Service layer. VQP will in turn issue queries (specific aim 2) to various wrappers that convert underlying reference ontologies like our own Foundational Model of Anatomy (FMA) and Physiological Reference Ontology (PRO) to a common representation language such as RDF. The task of performing this integration will be accomplished by our funding a small percentage of time for Center software developer Archana Vembakam, who will work with our own software developers.

Specific collaborations with Core 1 personnel
The primary core 1 personnel we will collaborate with are at Stanford and the University of Victoria. The Stanford investigators are Mark Musen, Daniel Rubin, and Natasha Noy, who have expertise in ontology development, maintenance and integration. Our University of Victoria collaborator is Margaret Storey, who has expertise in information visualization

Mark Musen is the PI of the Center, head of the Stanford Medical Informatics Program, and the primary author of the popular Protégé ontology development tool [2]. The FMA is implemented in Protégé and is the basis for much of our proposed work. In fact the FMA is so large and complex that it necessitated changes in Protégé itself [3].

Natasha Noy is a computer scientist and expert with primary interest in the more theoretical aspects of ontology alignment and management, whereas Daniel Rubin is an informaticist with interest in the practical aspects of ontology development and alignment. His practical bent will complement the more theoretical interests of Natasha Noy.

As per our discussions with Center investigators, the Center Core 1 research plan does not involve the development of methods for deriving application ontologies from reference ontologies, nor does it involve providing access to these derived ontologies as services, both of which will be addressed in this proposal. However, as noted in the following paragraphs these capabilities are envisioned in the Center long range vision, so the work we do will provide an important component of this vision that current Center resources do not permit. In addition, all three of these Core 1 investigators have been involved in several projects of direct relevance to this proposal. We therefore envision their primary role as working together directly with us to jointly develop these methods

In particular the original vision for BioPortal (before it was called BioPortal) [4] describes an architecture that is very compatible with the long term vision of our Structural Informatics Group (SIG), which is basically that of a service oriented architecture (section C.2). We will therefore be able to take advantage of ongoing BioPortal efforts, and the tools we develop, which will be based on our long experience with this kind of architecture, will fit in well with BioPortal. One aspect of the BioPortal vision is the notion of Ontology Web Services [5], which would provide various classes of web services for ontologies. These classes include query access, generation of ontology views, translation of ontologies from one language to another, management of multiple ontologies, and reasoning on ontologies. Our work in aim 2 will contribute to two of these components: query access and generation of ontology views.

Elements of the above overall visions have already been studied in the past by Center investigators. For example, Noy and Musen [6] use the FMA as a test case to describe a procedure for computing ontology views. As described in section A, a primary goal of this proposal is the computation of smaller and simpler application ontologies from reference ontologies. The basic approach is to compute the application ontology as a view over the reference ontology. Thus, the work of Noy and Musen is very relevant . In their case a view is computed as a collection of traversals along a defined set of link types, where each traversal can be of a specified length. Although this approach doesn't satisfy all our requirements it will be very useful as one of our starting points. In addition the discussion section of this paper describes several possible future directions that we will explore in our collaboration.

A second directly relevant project by Center Core 1 investigators is the recent conversion of the FMA to OWL-DL and OWL-Full and the proposal to access a non-materialized version of the FMA in OWL-Full through web services [7]. Since it is likely that our representation for ontology views will be some form of RDF/OWL (aim 1) and that we are proposing to implement web access to non-materialized ontology views, the experience behind this paper will be invaluable in our future collaborations.

These projects are primarily relevant to our aims 1 and 2. Relevant work for aim 3 (development of a graphical user interface or GUI for specifying views) is found in the Prompt Protégé plugin, also developed by Noy [8, 9]. Prompt includes a graphical user interface that, among other things, allows an end-user to specify the starting nodes and links that will be input to the traversal method for generating views. This experience and methodology, together with several visualization plugins developed by members of the Protégé community, will be valuable in aim 3 as we develop GUIs to specify the more complex views we expect to develop in aims 1 and 2.

Aim 3 will also (and primarily) take advantage of the expertise of Margaret Storey at the University of Victoria, an expert on Human Computer Interaction. One of Dr. Storey's primary interests is information visualization, and, in particular the custom visualization of large graph structures. As a result of that interest she and her students have developed the SHriMP graph visualization toolkit [10], and have applied it to the visualization of ontologies in the Jambalaya Protégé plugin [11]. In Jambalaya and Shrimp the main visualization metaphor is a "nested directed graph", in which subclasses or instances of a selected class are displayed as boxes within a larger box representing the parent class, and other relationships between classes or instances are represented as lines. Any of these structures may be zoomed to show more detail.

In related work Dr. Storey's group is developing tools for model driven visualization (MDV), which generalizes the nested directed graph as one of several "metamodels" that could describe a particular view of a graph [12].

Other metamodels could be a Tree view, or a Graph view (as a flat set of nodes and links). Although the current MDV work is in the area of source code visualization as a plugin to the Eclipse Integrated Development Environment [13], the concepts should be applicable to any graph structure. A possible direction to explore in aim 3 of the proposed work is to develop an ontology of visualization metamodels, and to compose selected components of this ontology with a reference ontology to automatically generate a visualization that is suitable as input to an application view specification GUI. These ideas will be discussed further in aim 3.

The fact that Center personnel have already been involved and will continue to be involved in view generation and graphical user interfaces means that we will be able to leverage their expertise beyond the relatively small percent time allocated in the budget. We envision the proposed work as a true collaboration, bringing together expertise in disparate areas to solve complex problems. Stanford brings expertise in ontologies, the UW computer science department brings expertise in databases, the University of Victoria brings expertise in graphical visualization, and the UW Structural Informatics Group brings expertise in developing, accessing and utilizing large reference ontologies. We believe that this combination of expertise will bring together the critical mass needed to move forward on these complex problems, the solution to which will greatly accelerate the progress towards a semantic web for the life sciences.

A. Specific Aims

The problem we address in this proposal is how to realize the potential of large, ontologically sound reference ontologies to become a foundation for the life science semantic web [14]. We will pursue this problem through the combined resources of CBio, the UW Structural Informatics Group and the UW computer science department.

The semantic web is emerging as the most promising long-term solution to the problem of data and computational model integration at the level of meaning as opposed to the level of syntax or communication. The vision of the semantic web is that local *ontologies* describing entities and relationships relevant to specific application domains will gradually be linked together into world-wide knowledge networks. Recognition of the importance of such ontologies for data integration has resulted in an increasing number of *application ontologies* of relatively circumscribed scope that are designed for specific biomedical subdomains. However, it is becoming apparent that taking the next step of linking these application ontologies together into the semantic web is difficult because of incompatible and inconsistent knowledge models. *Reference ontologies*, on the other hand, are an emerging ontology type designed to capture basic medical knowledge in an ontologically sound manner that permits such knowledge to be reused in application ontologies, just as basic medical science is reused in research and clinical medicine. If application ontologies can be derived from or linked to one or more such reference ontologies then the problem of linking together and reasoning across application ontologies becomes much simpler since the applications will be derived from a common corpus of basic medical knowledge. However, because such reference ontologies are large and complex it is currently very difficult for application developers to derive relevant segments from different reference ontologies and combine them into an application ontology required for addressing a real world problem.

It is the purpose of this proposal to overcome the obstacles to the use of reference ontologies as a basis for the life sciences semantic web. We will approach this problem by exploring a generalized version of the database concept of "view" over ontologies rather than data. In so doing we will collaborate with local database experts on data integration, and Center investigators with expertise on ontology view generation and ontology visualization.

The exploration of these methods will be primarily driven by the needs of computational modeling experts to integrate and share experimental and model-derived data as part of efforts to develop large-scale simulations of the whole organism.

We will pursue the following specific aims in order to develop these methods, each of which will be evaluated primarily by its ability to generate the application ontologies needed for integration of computational model data:

1. **Investigate and develop view-based approaches** for deriving application ontologies from reference ontologies.
2. **Design and develop software tools** that implement view based approaches for deriving application ontologies from reference ontologies.
3. **Develop a graphical interface** that allows domain experts to specify target application ontologies from one or more source ontologies, without the need to understand the underlying mapping formalism.

The expected outcome of this research will be a set of view generation and access tools that will become part of the CBio BioPortal framework. These tools will permit domain experts to graphically create application ontologies that, because they are derived from reference ontologies, will be based on sound ontological principles, yet will be simple enough to be used directly in applications. Such tools should greatly speed up the dissemination of large reference ontologies into real world applications, thereby providing a solid foundation for the life sciences semantic web.

B. Background and Significance

B.1. The need for data and computational model integration

This proposal is motivated by the need to integrate vast amounts of complex biomedical information into a holistic understanding of biological function, and to apply that understanding to improved health care. Because of the complex nature of biological organisms such understanding will increasingly take the form of complex, multiscale and distributed computer simulations, and such simulations will in turn rely on the integration of heterogeneous and widely distributed experimental and clinical data.

This need is illustrated by our collaboration with Dr. James Bassingthwaighte, a preeminent expert in computational cardiovascular simulation. Among his many accomplishments, he is the originator of the Physiome project to combine and integrate simulation models of the body [15, 16], and is the current director of the National Simulation Resource (NSR) at the UW [17]. As part of his work at NSR he has directed the development of the JSim simulation system [18], which has been used for many applications in cardiovascular physiology, ranging from the molecular level [19] to high level circuit based models of the cardiovascular and pulmonary circulations [20, 21].

JSim, like most simulators, includes a Mathematical Modeling Language, MML, for defining a model. A researcher uses this language to declare simulation variables such as blood pressure or blood flow, as well as relationships among these variables, which are generally expressed as parameterized differential equations. The language requires assigning initial values for the variables, and values for the parameters of the equations. The resulting simulation model is input to the simulation engine, which runs the simulation and displays the predicted results on a graphical interface. Depending on the simulation model a simulation can run for less than a second to many days. Once the simulation is complete it must be validated by comparing the predicted variable values against existing experimental data, or against values obtained as a result of new experiments.

All these steps involve data: experimental data for validating the predicted values, predicted data since long simulation runs often preclude re-running the simulation each time a new analysis is done, and model parameters obtained from the literature or as aggregations of experimental data. Until recently, the simulation models were small enough and isolated enough that it was relatively easy to simply keep the data in flat files or spreadsheets, and to look up a few parameters in known literature references. But as projects like the Physiome [22] or Digital Human [23] attempt to create large-scale models from many smaller models that may be distributed over the Internet, it becomes increasingly essential to develop methods for sharing and integrating experimental data and model parameters.

An important step in both data and model integration was the advent of XML, and XML markup languages such as Systems Biology Markup Language SBML [24], and Cell Markup Language (CellML) [25]. As part of the Physiome project there are now several efforts to convert diverse simulation models into SBML or CellML. However, XML and XML markup languages alone are not enough to ensure data or model interoperability [26]. XML by itself is only a syntax; it does not enforce meaning. In order for data and models to be interoperable they must also describe the same things or at least be capable of being mapped to the same things. For example, a random CellML file we looked at from the Physiome site has, within a <component> tag with attribute “name=Fixed_parameters”> a <variable> tag with “name = F”. Since this variable is a child of the “Fixed_parameters” component, we know by inspection that it is one of the unchanging parameters that will be used in later equations. However, there is no indication in the CellML file itself what the variable “F” means. The only way to find out the meaning is to look at the accompanying publication, and to read that in this case “F” is Faraday’s constant. However, in some other CellML file describing cardiac blood flow the same <variable> tag “F” might refer to blood flow. In addition, there is no way in the CellML file to know where the value for this fixed parameter was obtained. Again, the only resort is to look at the literature, but even then the source is not always clear.

B.2. The semantic web

These examples, which are the norm rather than the exception in data and model integration, have given rise to the need for more flexible and extensible integration at the semantic level. This need has in turn given rise to the vision of the Semantic Web [27-29] as a layer of meaning on top of the Web that can be used to link data and models by annotating or mapping them to common semantics.

The key components of the Semantic Web are the Resource Description Framework (RDF) [30], Ontology Web Language (OWL) [31], and, arguably, web services for accessing information encoded in RDF and OWL [5]. Together RDF and OWL permit the construction of networks of *ontologies*, where an ontology is a representation of the entities, and the relationships among those entities, within a defined application domain [32]. RDF alone is a relatively simple yet elegant graph language that defines a distributed, semantic network (a kind of ontology) as a collection of triples, where each triple describes a source object, a relationship and a target object, whereas OWL is a layer on top of RDF that allows the expression of additional knowledge not captured in raw RDF. An important feature of RDF is that each object and relationship is described by a Uniform Resource Identifier (URI), which provides a globally unique identifier. This feature, plus the fact that RDF operates under an open world assumption, means that global networks of relationships can incrementally be built by linking to existing RDF networks.

The promise of the semantic web for data and model integration is that separate database schemas or model parameters can be built by reference to shared semantic web ontologies, and data can be annotated with terms from these shared ontologies. Languages like CellML are already moving in this direction. For example, in the aforementioned CellML file several of the tags refer to an ontology of common concepts like “publisher” and “creator” that are part of the Dublin core system used for identifying documents [33]. As long as documents include tags from this common resource, search engines will know that the meaning of “creator” is the same in all the documents.

What is needed therefore is to extend the use of these shared ontologies to all the tags in the CellML file, including individual model variables such as “F” above. In this way, when trying to integrate various models the meaning of the parameters will be clear. In addition, once model parameters and variables refer to a shared common understanding (ontology) then this ontology can be used to either relate or derive shared databases that contain values for these variables.

The problem is that there is no such widely shared ontology of variables like F. However, because of the growing popularity of ontologies, if not now then soon there will be not one but many independently developed ontologies that describe concepts like “Faraday’s constant”. In fact, an important clearing house for these efforts is the CBio-sponsored Open Biomedical Ontologies (OBO) project [34], which currently houses a growing number of ontologies, from fields such as Zebrafish biology, human developmental anatomy, and many others.

Virtually all these *application ontologies* have been or are being developed by domain experts for use in specific types of applications. As such they generally do not conform to principles that permit their scaling up to larger or more complex representations, nor which permit them to be easily linked to other ontologies in order to develop a common shared understanding. [35, 36]. As more ontologies are developed this problem of incompatible ontologies is becoming reminiscent of the very data integration problem that ontologies are designed to solve. Although the problem is not likely to grow as large as that of data integration because there will be fewer ontologies than databases, it nevertheless will need to be solved if the semantic web is ever to become a reality.

B.3. High-level and reference ontologies as a foundation for the semantic web

Our approach to this problem is to develop methods that allow application ontologies to be derived from *reference ontologies*. The concept of reference ontology arose largely as a result of our efforts to develop the Foundational of Anatomy (FMA, see preliminary results) [1] and is now recognized by the biomedical informatics community as an ontology type that is distinct from the application ontologies currently in use.

Unlike an application ontology, a reference ontology is not designed for any specific application domain, but is intended to be re-used in multiple domains. To-date there is only one biomedical reference ontology in existence, the FMA, but others are in development: a Physiology Reference Ontology (PRO) [37], and pathology and developmental ontologies through our collaborations with Barry Smith at SUNY-Buffalo [38, 39]. The notion is that each of these reference ontologies will encompass one of the fields of basic medical science, and just as basic science knowledge is re-used in multiple ways in research and clinical practice, so too will reference ontologies be re-used by including segments of one or more of them in different application ontologies. Since reference ontologies are a relatively new development they are being designed as extensions or specializations of high-level ontologies that take a global view of multiple domains of reality, and do so in accordance with principles of ontology science [35, 39]. Therefore, deriving application ontologies from reference ontologies reduces or eliminates the scale up problems currently presented by most application ontologies. In addition, since reference ontologies are designed to be non-overlapping in accordance with high level ontologies such as the Ontology of Biomedical Reality (OBR) [39], they may be easily combined and linked together.

B.4. The problem with reference ontologies

Because of these qualities reference ontologies are a natural candidate for providing the infrastructure needed to create a semantic web for the life sciences [14], which could in turn lead to a shared understanding of variables like “F” in the previous CellML example. However, the promise of reference ontologies will only be realized if ways can be found to utilize them in specific applications. Because they are meant to be reused, reference ontologies are broad and deep, whereas application ontologies, which are only designed for specific applications, are narrow and shallow. Reference ontologies are designed according to strict ontological principles [35], which allows them to scale up, whereas application ontologies are designed according to the viewpoint of an end-user in a particular domain. The result of these differences is that reference ontologies are too complex to be used “out-of-the box” in applications, even when developers are aware of them and would like to use them.

As developers of the FMA, we have experienced this problem first-hand. For example, several potential users, including the developers of RadLex and the NCI Thesaurus, have invented their own application anatomy ontologies because they lack the methods for extracting from the FMA the segments of anatomy relevant to the tasks their applications address. And of the many groups that have licensed the FMA the first request is almost always for a “slice” that contains just that portion that is needed in an application, and that simplifies the ontology by eliminating the ontologically necessary but confusing nodes that irrelevant to a particular application. In response to these requests we have therefore programmed ad hoc methods to create many such “slices”, each one of which is slightly different from all the others, and most of which must be redone when the FMA becomes larger or the application needs change.

In addition to the fact that these slices must be programmed in an ad hoc fashion by informatics experts, a major versioning problem occurs when the master FMA changes or the slices are independently changed and updates need to be propagated back and forth. This problem is compounded if the slices are in turn used, in combination with other ontologies, to derive yet additional application ontologies.

These issues lead to the following specific research problems that we will address in this proposal. The *first problem* is how to generate application ontologies from one or more reference ontologies. Rather than continuing to generate ad hoc application ontologies, we would like to develop formal methods for mapping between reference ontologies, such as the FMA, and application ontologies. The methods should be specified in a declarative way (rather than as ad hoc programs) so that they may easily be re-run as the source ontology changes, and so that the specification may be generated and manipulated by graphical interfaces.

The *second problem* is how to provide access to these application ontologies through query interfaces rather than as downloadable files. This problem must be solved in order to link large ontologies into the semantic web, but on a more immediate timescale it arises because of the issue of version control: the end-user may make changes to a generated slice and we may want to incorporate those changes in a version of the FMA that has itself changed. Thus, our long term goal is to never actually deliver the application ontologies to the end users, but instead to make them available as web services that can be queried by web-enabled applications. Such an approach should greatly reduce the versioning problem since the query interface always has access to the most up-to-date version of the reference ontology, and end-user programs, while they can link to the reference ontology, can never change it because they don't have the source ontology.

B.5. Approach

Our approach to these problems is based on the concept of *views* that is prevalent in the database world. In database terminology, a view is a query that computes a new table from old tables. In our proposed work we will extend this notion to ontologies, in which an application ontology becomes a view of a reference ontology, expressed not as a document, but rather as a query defined in some well defined query language (like SQL in the relational database world). The advantages of this approach are 1) the methods for defining the view (application ontology) are explicitly specified by the queries defining the view, and hence can be manipulated by another program such as a graphical interface, 2) the application ontology is always up-to-date since it is *non-materialized* (or virtual) – that is, it is only defined by the set of queries constituting the view. At any time the view can be *materialized* by running the queries and saving the file, which can then be made available in whatever format is most convenient for the application developer to download. However, as the computing infrastructure increasingly becomes a service oriented architecture (SOA) like that in CBio's BioPortal framework, many applications will not need the materialized view and will simply to be able to access it as a web service. In such cases the "virtual" application ontology never becomes out of date because the queries always directly access the reference ontology from which it is derived.

B.6. Research issues

The basic problem we therefore need to solve is how to derive a non-materialized application ontology from one or more reference ontologies. This problem gives rise to the primary research issues we will investigate in the proposed work: 1) how to specify the view needed to derive an ontology, 2) how to build a web service that accepts queries over the application ontology and reformulates them as queries over the underling reference ontolog(ies), 3) how to make the process of answering queries efficient enough to be useful, and 4) how to allow biology experts to specify the views without having to learn the complex view definition language that we will undoubtedly need to create.

Many of these issues have been studied in the database community, with considerable contributions by our collaborators in UW computer science. In addition, some efforts have been made to develop ontology views [6, 40-42], an area of expertise of our CBio collaborators Noy, Musen and Rubin. In researching these issues we will therefore be able to take advantage of the expertise of our collaborators in database and ontology views, as well as our own expertise in the FMA and methods for accessing it. More detail on the work that we will draw on is provided in the research design (section D).

B.7. Significance if our approach is successful

Successful completion of our aims will result in a set of tools that are of general use for extracting application ontologies from reference ontologies. In addition, such tools will allow us and CBio investigators to more easily satisfy requests for the creation of slices from the FMA and other evolving reference ontologies. More importantly, by developing methods by means of which ontologically sound reference ontologies of the basic medical sciences can be reused in application ontologies, we will provide an essential component for the life sciences semantic web, which in turn will be an essential component of efforts like the Physiome to integrate computational models and data into a holistic understanding of biological function in health and disease.

C. Preliminary Results

Our work will build on and combine efforts from the various collaborators involved in this proposal: 1) the University of Washington (UW) Structural Informatics Group (SIG), which developed the FMA and various servers and applications that access and use it; 2) the UW Computer Science department database group, which has expertise in databases and data integration, and has collaborated with SIG on several related projects; and 3) the National Center for Bioontology (CBio), which has many areas of ontology expertise, the most relevant for this proposal being the generation of views from ontologies, and the creation of ontology visualizations. Relevant work by the Center is described in the preamble that precedes section A. Relevant work by the UW database group is described in aim 1. Here we describe work in SIG, which is the originating group for this proposal.

SIG is an interdisciplinary research collaboration among biologists, informaticists, computer scientists and engineers [43]. The goals of SIG are to develop computational representations for the structure (anatomy) of the body, to use these representations as a basis for methods that organize both anatomical and other biomedical information, and to implement these methods in a distributed structural information framework for biomedical data and knowledge [44]. The rationale for the emphasis on structure is that anatomy is the primary basis for understanding in biomedicine. Therefore, if methods can be found to represent anatomy in a computable way, then these representations can form a rational basis for organizing much of biomedical information.

C.1. The Foundational Model of Anatomy (FMA) reference ontology and its derivatives

The Foundational Model of Anatomy (FMA) is our symbolic representation for the structure of the body, and our primary proposed framework for organizing other biomedical information. Developed over a period of ten years under the leadership of Cornelius Rosse [45-47] [1] from the simpler and less principled Digital Anatomist symbolic knowledge base used for anatomy education [48], the FMA is a representation of entities and relationships necessary for the symbolic modeling of the phenotypic structure of the human body, both in computable and human readable form. The FMA is an abstraction that explicitly represents a coherent body of declarative knowledge about anatomy as a domain ontology. The ontology is currently implemented in the Protégé frame-based system developed by CBio investigators [2] and is stored in a relational database.

We regard this model as *foundational* for two main reasons: (1) anatomy is fundamental to all biomedical domains; and (2) the anatomical entities and relationships encompassed by the FMA generalize to all these domains. The FMA currently contains 75, 000 classes or universals, representing structures ranging in size from biological macromolecules, to subcellular components, to cells [49] to the whole body itself, including neuroanatomy [50, 51]. These classes are associated with more than 120,000 terms, either as preferred terms, synonyms, eponyms or non-English equivalents, and related to one another by more than 2 million instantiations of over 170 kinds of relationships. The detail and scope of represented anatomical knowledge far exceed any currently available computable anatomical resources.

We developed and populated this large and complex model through an approach we call *disciplined modeling* [46], the elements of which guided the development of three major components of the FMA, namely: the Anatomy Taxonomy (AT) [52], the Anatomical Structural Abstraction (ASA) [53] [54] [55] [56] and the Anatomical Transformation Abstraction (ATA), which deals primarily with embryology and currently exists only as a high level ontology but is not yet instantiated to any extent [1]. A fourth component, Metaknowledge (Mk), specifies all the rules and principles that govern the model's other three components. The abstraction of the model is represented in a high level scheme: **FMA = (AT, ASA, ATA, Mk)**, which captures the information that is necessary for describing the anatomy not only of the whole body, but also of any structure (physical object) or space that constitutes the body.

The FMA is intended as a reusable and generalizable resource of deep anatomical knowledge, which can be filtered (using techniques to be developed in this proposal) to meet the needs of any knowledge-based application that requires structural information [57-59] [60]. As noted earlier, the FMA is distinct from application ontologies in that it is not in itself intended as an end-user application and therefore does not target the needs of any particular group. It is an ontology which serves as a domain reference, the goal of which is to accommodate, reconcile and incorporate all the different views of anatomy, hence a *reference ontology*.

C.1.1. Influence of the FMA on informatics research

Within SIG and the UW Biomedical Informatics program of which SIG is part, the FMA has been used in several informatics research projects. For example, Gennari et al [61] describe two experiments that use the

FMA as a hub for linking separate genomic knowledge sources (the Gene Ontology [62] and Mouse Genome Database [63]), the notion being that a comprehensive reference ontology like the FMA can serve to reconciliate separate sources. As another example, Travillian et al describe approaches for mapping human anatomy in the FMA to the anatomy of model organisms [64], particularly the mouse [65].

In addition to this internal use within UW informatics, over 70 US and international non-profit institutions have licensed the FMA (which is free for non-commercial use), and many have used it as the basis for their own informatics research [35, 36, 66-77]. To a large extent because of this external validation and use, the FMA has recently been adopted as the anatomy standard by the European working group (CEN) of ISO for health informatics, (personal communication Jean-Maries Rodrigue, Gunnar Klein), and as the ontology on which the next edition of Terminologia Anatomica (TA), published by the Federated International Committee on Anatomical Terminology (FICAT) (personal communication Pierre Sprumont) will be based. This latter development is significant because TA has been the standard anatomical terminology for over 150 years, and is the basis for most textbooks of anatomy. The adoption of the FMA by the TA committee (who are generally not informaticists) would require significant reorganization of TA, and would be a clear indication that anatomists (as opposed to informaticists) recognize the value that the FMA, and more broadly, computability, would bring to the discipline of anatomy.

C.1.2. Generalizing the FMA

CBio investigator Barry Smith at SUNY-Buffalo has asserted that the FMA is the only biomedical ontology that conforms to sound ontological principles, which has led to a collaboration between the Buffalo group, Cornelius Rosse, and other SIG FMA developers. The purpose of the collaboration is to combine the principles of the FMA with the ontological principles espoused by Smith [35]. This collaboration has led to formalizations of structural relationships [78, 79] [80], and plans for developing new reference ontologies in pathology [38] and other basic sciences such as developmental biology. In addition, a higher level ontology has been proposed, the “Ontology of Biomedical Reality” [39], that would provide an overarching framework for relating the next level basic science reference ontologies to each other. These in turn would, through methods to be developed in this proposal, be the basis for derived application ontologies. Through our consultations with Cornelius Rosse we will keep informed of progress in these areas, and will utilize new reference ontologies as they develop.

C.1.3. Physiology Reference Ontology

In addition to the FMA and any new reference ontologies to be developed through the Rosse collaboration with Barry Smith, we will also have access to a “Physiology Reference Ontology” (PRO) being developed under the leadership of Dan Cook, a physiologist and simulation expert in our group [37]. In keeping with the principles being developed for reference ontologies [38] [39], we are endeavoring to keep the concepts in the FMA distinct from those in PRO. Thus PRO is being developed according to functional principles independent of their location in the body, the idea being that application ontologies derived from both PRO and the FMA will localize physiological processes, just as application ontologies derived from the FMA and a pathology reference ontology will localize pathology. PRO has already been part of outside informatics research [81, 82].

C.2. Providing access to the FMA

The previous sections demonstrate the scope and influence of the FMA, not only as the largest and most principled biomedical ontology (as seen by several outside evaluators) but also as the inspiration for the creation of other basic medical science reference ontologies that together have the potential to provide a conceptual foundation for the life sciences semantic web. However, to-date there are very few actual applications built by others with the FMA, which is testimony to the challenge we face in making reference ontologies useful to anyone other than informatics researchers. In the remaining sections of these preliminary results we demonstrate that SIG has already made significant progress in making the FMA useful for practical applications that we ourselves have developed.

We first describe our approaches for making the FMA accessible as a remote service to other applications, since we have envisioned from the beginning that the FMA in its entirely would be too large to be simply loaded into a running application. In fact, since 1989 [83] we have consistently envisioned the FMA and its predecessor, the Digital Anatomist Symbolic Knowledge Base [48], as one component of a distributed client-server architecture we call the Digital Anatomist Anatomy Information System (AIS) [48] [84] [85] [86]. The design of the AIS more or less conforms to what is now called a Service Oriented Architecture (SOA), in which a set of resources are made available to various Internet applications via a set of middleware servers. In fact it is very similar to the CBio BioPortal architecture (Figure 1), which means that the tools we develop in the proposed work should be easily adaptable to BioPortal.

As one component of the AIS, the FMA has been made accessible to applications over the years by various incarnations of a Foundational Model Server (FMS)[48] [87, 88]. Our current implementation of the FMS is OQAFMA (OQAFMA Querying Agent for the Foundational Model of Anatomy) [88], which queries an optimized (for efficiency) version of the underlying FMA relational database. The query interface for OQAFMA implements a subset of the StruQL Query language, which was developed by a team at AT&T that included our primary UW computer science collaborator, Dan Suciu [89]. Unlike SQL or XQuery, the query languages for relational and XML databases respectively, StruQL is highly optimized for queries over graphs of the type used to represent semantic networks. In addition to 1) ***selection* queries**, which only retrieve a node and its immediate neighbors in the semantic network, StruQL also supports two additional query types, which we have called: 2) ***projection* queries**, which allow one to follow a single link type to an arbitrary depth, as for example, find all the parts of the parts the heart to any level of depth; and 3) ***path queries*,** which allow arbitrary combinations of selection and projection queries, as for example, find all parts of the heart to any level of depth that are subclasses of “cardiac chamber”.

An important feature of StruQL and OQAFMA is that paths may be specified as regular expressions over link types. For example, to find all entities contained in the thorax or any of its parts, a link path may be specified as a regular expression, “part*.contains”, which means to follow 0 or more part links from the “Thorax” node , followed by a contains link. This feature, together with the ability to specify alternations (more than one possible path from a starting node), permit the creation of “virtual paths” that can hide the complex details of the ontology from applications. For example, a natural query is to ask “What is contained in the thorax?” If this query is directly given to the Protégé API the answer will be null, because an entity can only be contained in an anatomical space, and the thorax is not an anatomical space, although some of its parts (thoracic cavity) are anatomical spaces. Yet, an application would like to be able to pose such natural queries without having to know the detailed ontological reasons why a direct query will produce the empty set. The ability to specify link paths as regular expressions raises the possibility of defining virtual paths as named regular expressions that can then be queried directly. Although not fully implemented in the current version of OQAFMA, such virtual paths could be an important component of the view definition language (VDL) that we will define in aim 1.

C.3. Applications that access the FMA

In this section we describe several applications we have developed that access the FMA as a service through the Foundational Model Server (FMS). These applications illustrate 1) the power of the FMA to support intelligent applications, and 2) the use of a service oriented architecture to support next-generation interlinked web applications of the type that we believe will predominate in the near future. All these applications can be accessed from our online demos page [90].

Viewing the FMA. We have developed several web-based interfaces to the FMA, all of which access some form of the Foundational Model Server (FMS). Among these are a web-based browser [91], a natural language interface [92] [93], and a graphical interface, called Emily [94]. Emily in particular permits users unfamiliar with OQAFMA to graphically formulate projection queries of the form <subject><relationship><object>, where any of these variables can be replaced by “unknown”. Values for the subject, relationship or object are selected from pull-down lists generated by searches over the FMA. These queries are translated into OQAFMA queries, and returned XML results are displayed in the interface. If all variables are selected the interface returns True if the relationship is satisfied, False if not. If one of the variables is unknown, the interface returns those entities that satisfy the constraint expressed in the triple. For example <”Heart”><”Part*”><”Unknown”> will return all parts of the heart to any depth. Results from individual projection queries can then be saved and combined with other projection queries using Boolean operators, thus building up complex queries from projections. The graphical query specification methods in Emily will be of use in designing the graphical interface for our view generation language in aim 3.

Intelligent image retrieval. An authoring tool we have developed, called AnnoteImage [95], allows a user to load an image, draw 2-D contours delineating regions of interest, then annotate (or label) these structures with names from the FMA by means of a process that contacts the Foundational Model Server (FMS) to display a list of possible names. A web-based application called the Digital Anatomist Image Manager [96] allows the user to upload the annotated images to a database that organizes the images into collections and subcollections. An image retrieval applet allows the user to graphically query the FMS to find the children along a particular link type (e.g., parts of the heart). All or selected parts can then be graphically added to an image retrieval query. The query engine contacts the image database to find all images that are annotated with any of these parts. The retrieved images can then be viewed as interactive atlases. The latter feature is based on our Digital Anatomist Interactive atlases [97], which are the top hit in a Google search for “Digital Anatomy”, have received numerous awards, have been linked to by over 95 countries, and receive on the order of 60,000 hits per day .

Intelligent scene generatation. The Digital Anatomist Dynamic Scene Generator (DSG) [98] and its successor, BioLucida [99], combine the FMA, 3-D model “primitives”, and a 3-D model database to dynamically generate graphical scenes corresponding to combinations of projection queries from the FMA. The 3-D primitives are triangular surface meshes created from cadavers with our Skandha software package for 3-D reconstruction from serial sections [100]. The primitives are small portions of these reconstructions designed to correspond to anatomical parts in the FMA. A web interface connects to a graphics server that in turn queries both the Foundational Model Server and a 3-D model database server to find 1) all entities along a single link type (parts of the heart), and 2) all those 3-D primitives which correspond to those parts. The retrieved parts are loaded into the server, and the color of each part is set according to the type of the object along the subclass hierarchy in the FMA (bones are white, arteries are red). The scene is rendered by the graphics server, saved as a 2-D snapshot, and returned to the web browser as a 2-D image. Web controls then permit the user to add to the scene via additional projection queries (“add the branches of the coronary arteries”), to rotate and zoom the scene, to remove or hide objects, and to save the result as a VRML (Virtual Reality Markup Language [101]) file for later interaction at a client workstation.

Data management. As part of our work on the national Human Brain Project (HBP) [102] and BISTI initiatives we have developed several web-based local lab management system building toolkits [103-106]. Of particular relevance to this proposal is an ongoing project called *Seedpod* [106, 107], a model-driven lab management building toolkit, in which a developer uses Protégé to create an ontology describing a domain of interest. The ontology also includes various classes and slots that describe the desired appearance and behavior of a database application. The ontology is fed to a Java application that transforms the frame-based ontology into a relational schema. A JSP/Java program interprets this schema to generate a web-based interface to the resulting database. Although still in development, this kind of declarative approach to the generation of lab management systems could potentially be driven by the application ontologies we will generate as part of this proposal, in particular the automatic generation of databases to manage simulation model data.

Wrapping data for sharing. Again as part of the HBP, we have developed service-based methods that dynamically convert relational data to XML, which has become the preferred data exchange language. The application, called XBrain [108, 109], has at its core the creation of a non-materialized XML *public view* of a relational database, which is expressed as an XQuery over a simple *canonical XML view* of the relational data. This public view can then be queried by XQuery as though it were a standard XML document. A web front end we are developing permits a user to graphically formulate an XQuery over the public view. This query is then fed to SilkRoute, a query engine developed by Dan Suciu and others in UW CS and AT&T [110], which composes the user XQuery with the public view XQuery to generate a series of SQL commands to the underlying relational database. The SQL results are then repackaged as XML for return to the XBrain application. The returned result may be displayed directly as XML, or, through XSLT, as HTML, CSV (comma separated values for input to Excel), or a custom image. In the latter case, if the query results involve data that are annotated with names from the FMA, they are aggregated and shown on a 2-D schematic image of a brain that is parcellated according to the FMA. We are currently working to extend this to a 3-D brain image that is displayed by our Java3D client-server MindSeer application [111].

The XBrain application is of particular relevance to this proposal because 1) the Silkroute public view over the relational database represents the kind of non-materialized view we would like to create from a reference ontology, 2) the composition of a user XQuery query against the public view XQuery suggests a possible approach to queries against non-materialized ontology views, and even views over views as a chain of composed queries; and 3) the methods we are experimenting with for graphical generation of XQueries suggest approaches for graphically generating view definitions in aim 3. Although XQuery is not the right choice for our ontology view definition language, the methods we have developed for XBrain should be applicable to whatever language we develop.

Data integration. Given the existence of several lab databases created with our lab management system building tools, as well as the ability to wrap relational databases as XML, we have experimented with several approaches to integration of data from multiple sources [112, 113]. Of particular relevance to this proposal is a prototype that allows the use of distributed XQueries over multiple remote XML data sources. The prototype, also developed in collaboration with Dan Suciu, is a generalization of XBrain called Distributed XBrain (DXBrain) [114]. In this system, distributed data sources are queried by an augmented form of XQuery, where XQuery functions are allowed to access external data sources accessible as web services. Most such data sources are either XML or dynamically wrapped by an application like SilkRoute to present as XML, but “foreign” query languages such as StruQL are allowed, as long as their results are returned as XML. The relevance of DXBrain for the proposed work is that it demonstrates the ability to distribute a query over multiple sources, then combine the individual results into a single overall result. This ability is a necessary part of our goal to generate an application ontology from more than one reference ontology, since the view query engine will need to distribute a single query over multiple sources.

D. Research Design

The specific aims represent our approach to solving the two overall problems we address in this work: generating application ontologies, and embedding them in query interfaces. In aim 1 we will investigate and develop the methods for creating and accessing application ontologies as views of reference ontologies. In aim 2 we will implement these methods in specific tools that will be integrated as plugins into BioPortal, and in aim 3 we will investigate and develop methods that hide the details of the formalisms in easy-to-use graphical user interfaces.

D.1. ***Aim 1. Investigate and develop view-based approaches*** *for deriving application ontologies from reference ontologies.*

View-based approaches are instances of the more general problem of mapping and query reformulation that has been extensively studied in the database community. Although some efforts along these lines have been explored in the ontology community, we believe that there is a rich source of database experience that has not yet been explored. At the same time database experts are not necessarily aware of the rich and complex research opportunities provided by the need to link together large ontologies in the semantic web. Thus, a key strength of our proposed collaboration is that it will bring together experts in databases (Dan Suciu, UW computer science), experts in ontologies (Natasha Noy, Daniel Rubin, Mark Musen, CBio) and experts in the creation and utilization of large reference ontologies (James Brinkley, John Gennari, Todd Detwiler, Jose Mejino, Dan Cook, UW Structural Informatics Group). Through numerous face-to-face meetings, teleconferences and email communication we will work together to investigate these problems, and will jointly develop solutions that will often be recorded in joint publications and a project web site. The primary point person for this project will be an informatics or computer science graduate student, who will investigate these areas as part of a thesis. We will undertake the tasks described in following sections in order to address these problems. The first task will help us refine the requirements for our view based methods. The remaining tasks represent our current understanding of these requirements.

*D.1.1. **Task 1**: Catalog and generate ad hoc application ontologies*

Our first task will be to catalog existing application ontologies we have created and generate new ones using our current ad hoc methods. Experience with real-world requests for FMA slices, as well as with the new ontologies we generate, will ensure that our work is "grounded" in real-world needs. These needs will influence our requirements specifications in task 2 (section D.1.2.1), and the generated application ontologies will serve as templates against we can evaluate our view-based application ontologies . In addition, the requests for application ontologies will often require that we change the FMA itself. We will therefore gain a better understanding of what knowledge should be in the reference ontology and what should be in derived application ontologies. This kind of comparison will be done by Jose Mejino, the primary content developer for the FMA.

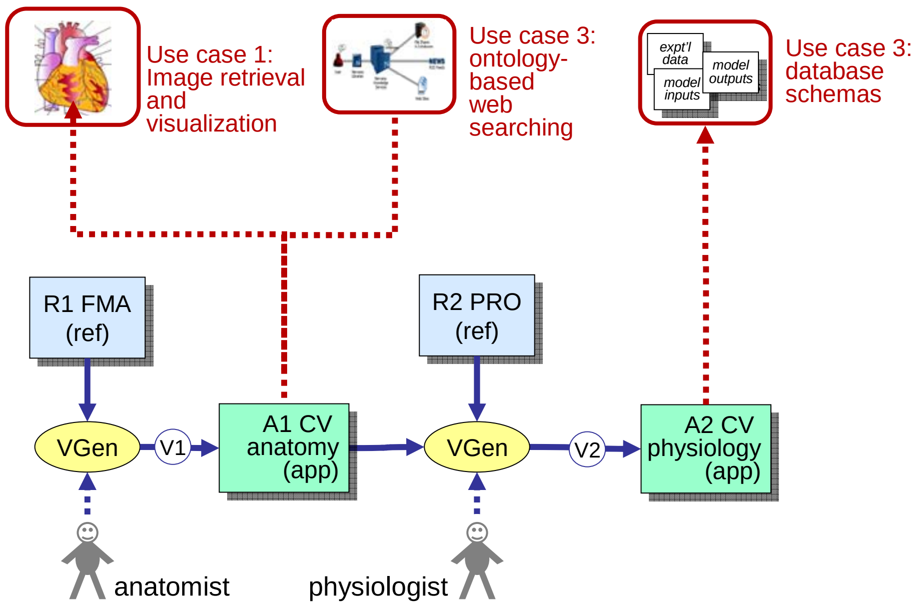

Figure 2 Use Cases

As part of this effort we will use our ad hoc methods to generate application ontologies corresponding to the use cases shown in Figure 2. The application ontologies are shown as boxes A1 and A2 in the figure. A1, an ontology of cardiovascular (CV) anatomy, will be derived from reference ontology R1 (the FMA). A2, a cardiovascular physiology applications ontology, will be derived from A1 and R2, our evolving Physiology Reference Ontology (PRO). That is, A2 will associate generic physiological parameters from PRO, with locations in the cardiovascular system from A1, which will in turn be derived from the FMA.

In the system we envision an anatomist or physiologist will use a graphical tool we will develop in Aim 3, called VGen (View Generator) to create a view (V1 or V2) which will specify the corresponding application ontologies A1 and A2. However, in aim 1 we will use ad hoc methods to derive the application ontologies, based on specifications by the anatomist (Mejino) and physiologist (Cook), and implemented by Todd Detwiler. The usefulness and validity of these ad-hoc generated ontologies will be assessed by our domain experts (Mejino, Cook, Bassingthwaighte). The ontologies will be evaluated for their potential utility in the applications implied by the use cases. The evaluation will be done by manual inspection using one of the many available ontology visualization tools (and later by the tools we will develop in aim 3). Our Stanford collaborators will also evaluate these ontologies for their own work.

These manually-generated and expert-validated application ontologies will then serve as templates for evaluating the tools we develop in aims 2 and 3. That is, in aim 1 we will evaluate whether the view specification language is expressive enough to generate the template application ontologies, in aim 2 we will evaluate how well the generated ontologies match the templates, and in aim 3 we will evaluate whether the GUI is rich enough to specify all the needed mappings.

Use case 3 in the figure corresponds to the problem described in the background: developing a shared understanding of the meaning of experimental data and simulation model parameters. However, this use case is the most complicated since it relies on integrating two separate ontologies. Therefore, we will initially develop our methods based on use cases 1 and 2, which only depend on single reference ontology, the FMA.

Use case 1: Cardiac image retrieval and visualization. The first use case is inspired by the need to develop methods to share and integrate the plethora of highly detailed 2-D, 3-D and 4-D images depicting cardiac anatomy. If a shared application ontology A1 can be developed that represents the structures that are visible in cardiac images, and if names from this ontology are used either as keywords, or, more precisely, as the labels for segmented structures on those images as we have done in our 2-D image retrieval and 3-D scene generator applications (section C.3), then search engines can be built that retrieve widely distributed images that are labeled with terms from the shared ontology. These images can then be visually compared based on common named structures. The search engine could be much more "intelligent" than an engine based solely on keywords since the search can be based on sets of structures and their relationships, all of which need to be visible on the retrieved images. Similar kinds of search can be also used to construct scenes showing only specific cardiac anatomy from the segmented regions of a patient image dataset. An application like our dynamic scene generator could then automatically generate a scene depicting just the anatomy needed to communicate a particular diagnosis to the patient. We will therefore develop the application ontology for use case 1 based on its potential applicability to 3-D scene generation and intelligent image retrieval.

Use case 2: Ontology-based search. The second use case is inspired by efforts to develop the next-generation Google: search methods that are based on the semantic web. As a specific example, one of our commercial FMA licensees, Nervana, Inc. of nearby Bellevue, WA (http://www.nervana.com) is developing and deploying semantic search engines for the biotechnology industry that are based on an expanding suite of biomedical ontologies (e.g., MEXH, NCI Theasaurus). The FMA as an anatomical reference ontology has been of particular interest to Nervana, but the sheer size and complexity of the FMA creates technical and computational challenges that limit access to its knowledge. As is the case with our other FMA users we have been working with Nervana to generate ad hoc slices, but recognize that the solution lies in researching and implementing the approaches proposed in this grant for creating and combining views of reference ontologies. As a test case, we will generate a Cardiovascular Anatomy Application Ontology as a subset of the FMA for the cardiovascular system. Even though this application ontology will be generated purely from the FMA it will not be the same as the application ontology generated for visualization because the expected use is different. See enclosed letter of support from the CEO of Nervana, Inc.

Use case 3: Generation of database schemata. In the third use case we will generate a cardiovascular physiology application ontology A2 that could be used to generate databases of simulation data, as described in the background (section B.1). In order to generate this application ontology we will first need to generate an application ontology of anatomical entities that are relevant for the simulation models.

An important distinction between this use case and the other two is that in this case the generated anatomy application ontology A1 must be combined with relevant terms from our Physiology Reference Ontology (PRO), which is the source of entities that represent the physical attributes (pressures, flow rates, etc.) that represent the experimental data and parameters of the simulation models. Thus, this use case gives rise to the need to derive an application ontology from two sources: 1) an application ontology that is itself derived from a reference ontology, and 2) another reference ontology.

*D.1.2. **Task 2**: Select query and view definition language (VDL)*

Our next task is to select an existing query and view definition language (VDL) as a starting point for our work. Because we would like to develop applications for the semantic web these languages should express queries over ontologies expressed in RDF/S or one of the various forms of OWL [31]. We will initially choose

RDF/S for the following reasons: 1) RDF/S is a simpler language than OWL; 2) every OWL document is a legal RDF document; 3) there are very few available query and view languages for OWL [115] [116], whereas there are many for RDF/S; and 4) we believe that, since our primary use cases do not involve substantial reasoning, RDF/S will be adequate for this research. However, we recognize that OWL is the evolving standard because it is more expressive than RDF/S [117]. Thus, in the long run we expect that as query and view definition languages over OWL are developed by others we will be able to take advantage of them. And since every OWL document is a legal RDF document we should be able to extend our tools to handle the additional semantics provided by OWL.

D.1.2.1. Requirements

In the next paragraphs we follow RDF conventions and call each pair of nodes connected by a labeled edge a *triple*. Using this terminology, the creation of a view over a source ontology involves two components: 1) a query component to specify which triples from the source should be included in the target, and 2) a view definition component that specifies how the selected triples should be transformed.

The Query component. Our experience with ad hoc views, OQAFMA, and Emily shows that with a large and complex ontology like the FMA simple queries of the type found in the Protégé API (nodes and immediate descendents) is far too inefficient. Based on our current understanding, the Query component of VDL will need 1) path queries which operate on regular expressions over a link chain 2) boolean combinations of the results of path queries as in Emily, and 3) constraints along link paths, of the type allowed in the XPATH query language for XML. Other requirements will be formulated as we gain additional experience with our ad hoc views.

View Definition. Given a selected set of triples from the source ontology, the second component of VDL must be rich enough to specify all the transformations that must be made on those triples before they are included in the target ontology. The simplest transformation is simply to copy all the selected triples to the target. This corresponds to Noy's View Traversal approach [6], and our own OQAFMA [88]. Other transformations will be more complex. For example, the FMA contains reified relations, that is, links that have attributes. An example might be that "The esophagus is adjacent to the trachea in the posterior direction". However, in some application ontologies a developer may only want to keep the fact that the esophagus is adjacent to the trachea, without regard to direction. As another example, a developer may want to remove an ontologically sound but confusing entity along a link chain, but in so doing will have to add a link between the two endpoints of the chain. In both of these examples, the view definition component of VDL will be a complex combination of changes, additions and deletions of triples.

D.1.2.2. Existing query and view definition languages

In contrast to OWL query languages, there are quite a few well developed query languages for RDF/S. Some of these include SparQL, RDQL, Triple, SeRQL, Versa, N3, RQL, RDFQL, and RxPath. A good survey of the features of some of these languages was compiled by Haase et al [118]. In our initial investigation we determined that RQL and SparQL are the most promising for our purposes.

SparQL is the W3C's proposed query language for RDF/S on the semantic web [119]. If SparQL were sufficiently expressive for our needs, it would be ideal to build our extensions onto this emerging standard. However, currently SparQL lacks sufficient support for path expressions (specifically, regular expressions over paths), a feature that we consider essential for a graph based query language. There has been some discussion amongst the SparQL contributors about adding such functionality to the language. If this happens it would greatly increase the value of SparQL to our project. However, since SparQL is an evolving standard, we intend to reconsider SparQL as a candidate for our VDL even if it requires us to incorporate such extensions ourselves.

Several groups have used RDF/S query languages to specify views over RDF ontologies [40-42]. Our initial survey suggest that among these RVL (RDF/S View Language) [40], appears to be the best starting point for our own VDL, but would need considerable extension to meet our needs.

D.1.2.3. Plan

Based on our initial survey the most likely query languages are RQL and SparQL . We will begin with RQL because it appears to come closest to meeting our initial needs, although we will monitor developments in the evolving SparQL web standard. Given the choice of RQL we will initially work with the RVL view definition language, again because it comes closest to meeting our requirements. Another advantage of RQL and RVL is that they were developed by a group that includes Dan Suciu's PhD thesis advisor, Val Tannen.

*D.1.3. **Task 3:** Extend the selected query and/or view language to handle additional view definitions*

Given our initial choice of view definition language the next task is to add needed features that are not present in the existing languages. An example of a needed extension in the query component of RQL/RVL is transitive closure over arbitrary properties. RQL does provide mechanisms for specifying path expressions. These types of expressions can be specified in RQL as:

```
select $X, $Y
from {$X}part.contains{$Y}
```

To do add transitive closure we would extend the RQL syntax to add a closure operator:

```
select $X, $Y
from {$X}part*.contains{$Y}
```

The above query binds to $X and $Y classes that are connected via any number of part relationships followed by a single contains relationship. Thus, extending the syntax to include transitive closure is relatively simple in RQL. The hard part will be adding these to the query engine. As another example of the need to extend the query component, we will also need to extend the language to query across more than one source, as we have done with XQuery for our DXBrain application.

In addition to our need to extend the query component, we will also need to extend the view definition component by adding syntax for expressing higher-level operations commonly used in the specification of ontology views. Example operations needed for converting the FMA including collapsing an attributed part relationship to just a part relationship, and removing intervening subclasses.

*D.1.4. **Task 4:** Develop methods for query processing over a single view*

In the remaining tasks we assume that VDL exists, that we have used it to specify views representing application ontologies like those in Figure 2, and we want to process queries against non-materialized views of those application ontologies. The query processor, which we call VQP in aim 2 (View Query Processor), will need to present the non-materialized view as though it were materialized (a "virtual view"). Thus, it should process all the same queries that the chosen query language (RQL) can process when applied to a document written in the chosen representation. It will then need to reformulate all such queries as queries over the source ontology. For example, in Figure 2, a query such as "Find all tissues that are parts of the heart", when applied to the derived application ontology A1 for the cardiovascular system, must be reformulated into a query against the underlying reference ontology R1 (the FMA) by composing it against the view V1 which generates A1.

If our initial representation is RDF and our query language is RQL (extended as in the previous task), then the query over A1 may take the form:

```
select Y
from {Y:Tissue}(PartOf)*{X}, {X:AnatomicalPart}Name{Z}
where Z like 'Heart'
```

This query is formulated over the application ontology A1. It specifies that Y is a class that "isA" Tissue, that X is a class that "isA" AnatomicalPart, and that Y and X are connected by a chain of PartOf relationships. It further specifies that X has a property "name" bound to Z, and that Z is a string that matches the string "Heart". All classes and relationships mentioned in this RQL query are part of the application ontology A1. However, A1 is not available directly, but is computed from a larger ontology, the Foundational Model of Anatomy, FMA.

There is a complex RVL query that defines the view V1 expressing how the FMA needs to be transformed into A1 The transformations performed by the RVL query may include: adding, removing, or renaming classes, adding, removing, or renaming relationships, and collapsing chains of relationships into a single relationships. Our system will take the RQL query above, and "compose" it with the RVL query defining V1, to obtain a new RQL query that is formulated over the vocabulary of the FMA ontology. In notation, it will take the queries Q and V1, and will compute a new query Q' = Q o V1. The rewriting of this query may involve renaming classses (e.g. AnatomicalPart may be replaced with Part, while tissue may be the union of several classes), and replacing single relationships with more complex ones (e.g. PartOf may become a longer chain of relationships).

The general problem is illustrated by use case 1: Given an RVL query V1 mapping an ontology R1 into an ontology A1 and an RQL query Q over the ontology A1, compute the composed query Q o V. That is, the user formulates a query Q over the derived application ontology A1. The system needs to rewrite Q into a new query Q' over the reference ontology R1.

An important and difficult variation of this problem is the case when the ontology is stored in a database (for more efficient query processing) and the RVL view V expresses how to construct the ontology from the database tables. Such will be the case for the "wrappers" we will develop in aim 2. For example, consider the FMA ontology. Given its size, in our OQAFMA work we have decided to store the FMA in a relational database, by representing it as several relational tables. There is one binary table for every relationship (e.g. the relationship PartOf corresponds to a binary table PartOf(A,B) storing all pairs of class identifiers x, y that are in the PartOf relationship), and there are precomputed views for some transitive closure traversals (e.g. there is a binary table PartoOfStar(x,y) storing all pairs of classes x, y that are connected by a path satisfying the regular expression PartOf*). User queries formulated over the FMA ontology using the RQL query language must now be rewritten into SQL queries that can extract the data from the relational database. For example, consider the RQL query:

```
select X
from {X} PartOf {Y:Organ}, {Z} partOf {Y}, {X} name {U}
where U like 'left ventricle'
```

which returns all classes X that are part of the same organ as the 'left ventricle'. The system cannot execute the query in this form, because the data is stored in a relational database. Instead, in our approach this RQL query Q is composed with the query V mapping the relational database to the FMA ontology, which results in the following SQL query:

```
select
from PartOf p1, PartOf p2, Name n
where p1.B = p2.B and p2.A = n.A and n.B = 'left ventricle'
```

In general, the query V is now defined over a relational database and returns an ontology: V is usually a large expression, since it needs to return a richly structured ontology, like the FMA. The user query Q is expressed in a language like RQL. The system composes Q with V to obtain a new query Q o V, which is (or can easily be rewritten to) SQL. The SQL query is now run on the relational engine, which is MySQL or PostgreSQL in the case of FMA. The general problem of composing a query with a mapping extracting data from a relational database is a complex one, and it has been intensively studied in the database community, especially for the case when the output data is XML and the query language is XQuery. One of the most complete and efficient systems was developed by us [110] and deployed in the XBrain and DXBrain applications described in the preliminary results (section C.3) [108, 114].

Our job in this task will therefore be to leverage the extensive work we have already done in the XML world with SilkRoute and XBrain among others, and apply these techniques to the reformulation of queries over view-based application ontologies.

*D.1.5. **Task 5:** Develop methods for query processing over multiple views or reference ontologies*

A complication to the previous task occurs when a query has to be distributed among several source ontologies. If one or more of the source ontologies is itself a derived application ontology then reformulation will have to proceed along a chain of application ontologies. For example, in Figure 2 application ontology A2 is derived from reference ontology R2 (PRO) and application ontology A1, which itself is derived from reference ontology R1. In this case a query against application ontology A2, such as “Find the names of the attributes describing blood flow in the chambers of the heart”, will need to be reformulated into a query that finds blood flow attributes from PRO, and a query from A1 which finds the chambers of the heart. This second query will then need to be reformulated again as a query over the FMA.

This combination is, in general, complex: it involves creating new classes corresponding to classifications in both ontologies, such as AnatomicalPart-BloodFlow corresponding to the intersection of the concepts AnatomicalPart in the ontology A1 with the concept BloodFlow in R2. The resulting ontology R2 is defined by a view V2 over both ontologies A1 and R2. Now consider a user query Q expressed over the resulting ontology A2. This is rewritten by our system into a new query Q’ = Q o V2 over both ontologies A1 and R2. The latter presents a problem, since it involves matching classes and traversing links in two different ontologies. For example, it may start from a class X in A1, traverse some relationships from X to reach a class Y in A1, then search for a class Z in R2 that has the same name as Y, follow some links from Z in R2 to reach yet another class U, then return to navigate in A1, etc. Such complicated queries that navigate in multiple ontologies and correlate classes from multiple ontologies need to be decomposed into fragments that can formulated over each ontology separately, then combined. We plan to develop an extension of RVL that allows us to specify how to split the query into smaller fragments that are expressed over separate ontologies. In so doing we will build on our prior work (section C.3) developing DXQuery (Distributed XQuery), which allows the user to express that different query fragments need to be evaluated on different XML data sources, and then combined in a specific way [114].

*D.1.6. **Task 6:** Develop methods to optimize for efficiency*

Once we have developed the basic methods for view definition and query formulation we next need to ensure that the response time to a query over a non-materialized application ontology is fast enough to permit interactive use. Achieving reasonable response time could be a challenge if long chains of query reformulations are involved. As in the previous tasks, our database expert Dan Suciu has extensive experience in query optimization. Some of the approaches we plan to pursue are:

Schema optimization. Our experience with OQAFMA showed that redesigning the relational schema underlying the FMA, and creating large numbers of indices, can dramatically improve response time.

Query optimization. Since the queries we execute will result from automatically composing a user query (typically in RQL) with a complex view definition (in RVL), the resulting query will be large, redundant, and highly inefficient. The sources of redundancy include: (1) the reformulator may compute temporary answers that are not used in the final answer (2) it may compute temporary answers only to query them later (as opposed to querying directly the input ontology), (3) it may perform the same computation multiple times (common subexpressions), (4) it may fail to exploit certain powerful features of the host data source (for example aggregate queries lead to nested SQL queries, instead of using the GROUP-BY feature in SQL). As part of this proposal we will develop a suite of optimization techniques for RQL and RVL queries that will eliminate all these sources of inefficiencies. We have gained a lot of experience on this problem in our work in SilkRoute [110]. There, a user XQuery is composed with a large view-definition expressed in XQuery, which usually results in a very large SQL query exhibiting all inefficiencies described above: we have developed a suite of optimization techniques to remove those inefficiencies.

View composition. In some cases domain experts might define two views in sequence. For example, in Figure 2, if R2 (PRO) were not present, then application ontology A2 would first be derived from application ontology A1, which would in turn be derived from reference ontology R1 (the FMA). Having two views in sequence is a source of inefficiency in the system. For example, consider a user query Q formulated over A2.

The system must first compose it with V2, to obtain a query Q o V2 expressed over A1; then it must further compose it with V1, to obtain a query (Q o V2) o V1, which is now expressed over R1 and can be evaluated on the server storing the Foundational Model of Anatomy. However, this process is inefficient, and may make the optimizations described above harder to deploy. Under the “view composition” approach, the two views are first combined into a single RVL query V2 o V1, which maps the FMA ontology R1 directly to A2. This is done by the system administrator at a time when the mapping V2 is first defined. Then, a user query Q can be directly composed with this view to obtain Q o (V2 o V1), which is an RQL query expressed directly over the FMA ontology R1. Note that the result is the same as without view composition, since (Q o V2) o V1 = Q o (V2 o V1), but now query answering is much more efficient.

*D.1.7. **Task 7**: Evaluate*

We will use a text editor to manually create VDL specifications for the three use cases. We will evaluate these specifications as to expressiveness, conciseness, readability, and potential to be manipulated by a graphical interface. Ongoing analyses of this evaluation will be used to redefine VDL in the previous tasks. Evaluation of the ontologies that will be actually generated by these specifications will be done when VQP is implemented in aim 2.

D.1.8. Alternative and additional approaches

Many of the tasks described above involve difficult research issues. If not all our proposed methods work, either in the design stage of aim 1, or when we implement them in aim 2, we will fall back on successively less ambitious goals. Some of these fallbacks include: 1) **caching of materialized views to improve response time.** If all the optimization methods described in section D.1.6 do not result in adequate response time, we can always materialize a view anywhere along the chain. We can then replace the query reformulator at a given point in the chain of views with a wrapper that provides access to the materialized view; just as in aim 2 we will develop wrappers over the original reference ontologies; 2) **manual editing of materialized views.** If we allow materialized views in the chain then we can also allow the user to hand edit the materialized views if the VDL specification is not adequate.; 3) **methods that only generate materialized views.** If the view-based approaches are completely inadequate (unlikely) then we can pursue extensions to applications like Prompt [8], which only generate materialized views.

*D.2. **Aim 2. Design and develop software tools** that implement view based approaches for deriving application ontologies from reference ontologies.*

In this aim we will develop software tools that implement several of the view-based approaches we design in aim 1. Given the complexity of the problems, we will develop progressively more sophisticated versions of these tools, and will work with CBio to design them so that they can be plugins to the service layer of the BioPortal framework being developed at CBio. Tool implementation will be done by one of the students and by Todd Detwiler at the UW. Incorporation of the tools into the CBio BioPortal framework will be done by CBio software developer Archana Vembakam, working in collaboration with Todd and the UW student. We will undertake the following tasks in order to develop these tools:

*D.2.1. **Task 1:** Design system architecture*

We have designed an initial system architecture as shown in Figure 3, which depicts a configuration of views and processes to implement the use cases shown in Figure 2. This design is heavily influenced by our experience designing an XML-based system for data integration using distributed XQuery [114], as discussed in our preliminary results (Section C.3). It is also influenced by the UW CS Piazza Peer Data Management system [120], which is in turn inspired by peer to peer file sharing networks. The system we propose includes the following: 1) A set of **source ontologies** (R1 and R2 in the figure). These ontologies may be reference ontologies or materialized application ontologies; 2) A set of **wrappers**, (W1 and W2). In data integration work a wrapper is generally some service that dynamically converts an underlying information source from its internal representation into a representation expected by the data integration system. In this case the wrappers present the source ontologies in the language expected by the View Definition Language (initially RDF/S);

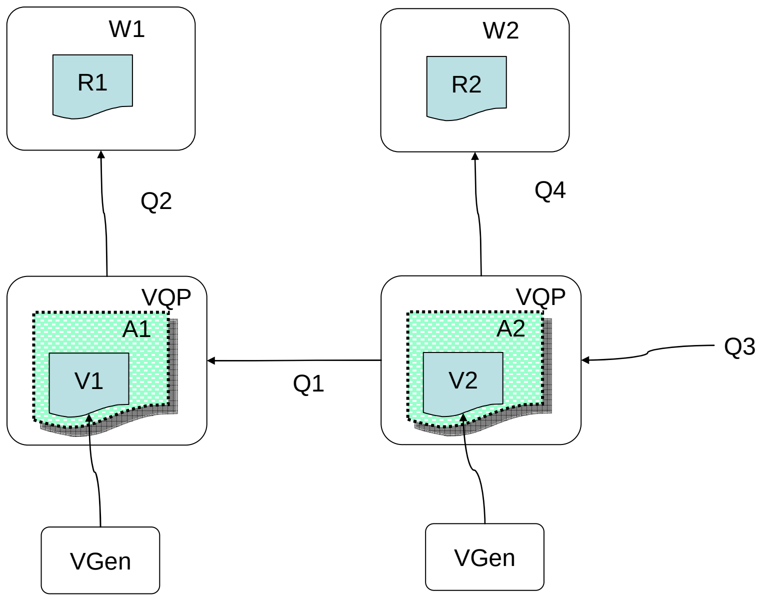

Figure 3 Visual Query Processor

3) A **view** V2 that specifies a virtual ontology A2, over one or more wrapped source ontologies R2, and possibly over another VDL specified view V1 that specifies a virtual ontology A1, both specified in a view definition language (VDL); 4) A **View Query Processor (VQP)**, which takes as input a VDL-specified view V, and a query Q over V, reformulates Q as queries over the source ontologies and views, and joins the separate results in a single output result. Each instance of VQP will run as a web service within BioPortal or in any other facility that wants to provide view-based access to its ontologies; and 5) A **View Generator (VGen)**, which is a graphical user interface that allows a user to specify V, and which will be developed in specific aim 3.

The basic concept is that the combination of a view definition V and an instance of the View Query Processor VQP constitutes a virtual application ontology A, that is, a non-materialized view of the underlying source ontologies. Furthermore, since the source ontologies may themselves be virtual, a web of non-materialized views may be formed, eventually terminating at "ground truth" ontologies presented by wrappers.

Figure 3 shows how this architecture could apply to the use cases illustrated in Figure 2. In this case the source ontologies R1 and R2 are the FMA and PRO respectively. Since the FMA is very large it is stored in a relational database. Therefore, the wrapper W1 must present the FMA as a virtual document in the representation language that is understood by VQP, which will initially be RDF/S. PRO might be small enough that it can be stored in a Protégé native file or converted offline to RDF, so in this case W2 can load R2 into memory. In either case, both W1 and W2 do not deliver the entire source ontology, they simply accept queries (initially RQL) over the wrapped ontology, reformulate them to the underlying format (SQL or memory accesses) and repackage the result in the desired output language (RDF/S).

Unlike Figure 2, where the cardiovascular anatomy application ontology A1 is shown as some materialized file, that same application ontology in Figure 3 is a non-materialized view over R1. The instance of VQP associated with A1 accepts RQL queries (say Q1, "Find all the chambers of the heart") over this non-materialized view, reformats them as queries Q2 over the wrapped FMA ("Find all the parts of the heart that are organ chambers") and returns the result. Similarly, cardiovascular application ontology A2 is shown as a non-materialized view over both A1 and reference ontology R2 (PRO). In this case some query Q3 over A2 might be something like, “Find all the flow variables for the chambers of the heart”, as issued by an application designed to generate a schema for a database of simulation model parameters. Q3 is reformulated by the instance of VQP associated with A2 into a query Q2 (“Find all the flow parameters”) over PRO, and the same query Q3 (“Find all the chambers of the heart”). The actual association between flow parameters from PRO and their location in the chambers of the heart is specified in view definition V2.

This design has some interesting properties that we will explore as we actually build these tools:

- The design is **fractal** in nature. Because an instance of VQP can access other instances of VQP, which themselves can access other instances, it is possible to build up very complex ontology webs from local sources – which is of course part of the vision of the semantic web. Although in the current proposed work we do not propose to try to interconnect independently developed ontology webs, this architecture would also allow such a capability if the mappings specified by VDL were bidirectional, a feature we plan to implement in future work.

- A virtual application ontology A can be materialized at any time, wrapped in some wrapper W, and made available as a wrapped “ground level” ontology to VQP. This capability could be very important when trying to make this system efficient enough for real applications.

This overall design will evolve as we implement the components. Given the view specification language and reformulation algorithms to be designed in aim 1, the two main components are the wrappers and VQP. These components constitute the next two tasks.

*D.2.2. **Task 2:** Design and implement wrappers over source ontologies*

The type of wrapper will depend on the format of the source ontology and the expected output format. Initially the output format will be RDF. The FMA is currently stored in a relational database. Thus, a wrapper for the FMA will need to convert from a (possibly optimized) relational representation to RDF. This is not a trivial operation for an ontology as large as the FMA. We will therefore take advantage of our work on SilkRoute, and work by Dameron at CBio [7] to convert the FMA to OWL and dynamically present it as OWL (although our initially query processor will just ignore the OWL constructs and treat the representation as RDF/S. This approach, however, may require optimization to improve efficiency.

Other ontologies may require less effort. For example, materialized application ontologies may already be stored as RDF, so a wrapper could simply be a query processor that handles RQL over RDF. Or, an application ontology could be stored in its original Protégé format, in which case a wrapper might use the Protégé API to access it.

Part of our work for this task will be to look at the existing FMA, PRO, and any ad-hoc generated application ontologies (section D.1.1) and decide which ones we simply want to materialize and which we want to access through a database. For our initial experiments it may suffice to simply materialize portions of the FMA so we can experiment with smaller versions of it. Our architecture should support this kind of plug-and-play.

*D.2.3. **Task 3**: Design and implement VQP*

Following our experience with the distributed XBrain application we will implement two progressively more complex versions of VQP, each of which will incorporate the methods developed in the corresponding tasks of Aim 1. The difference in each is the type of source ontologies and the corresponding increased complexity of VDL as well as the reformulator: 1) VQP1 – a single wrapped source ontology; and 2) VQP2 - multiple source ontologies, some of which will be wrapped and others of which will be presented by other instances of VQP. Since both the wrappers and VQP will present the same query interface (RQL initially), once VQP2 is implemented it should be possible to construct arbitrarily complex webs of interconnected application ontologies, all of which are eventually derived from reference ontologies representing the basic medical sciences. Like the wrappers, we expect to implement the various versions in Java and to embed them as web services.

*D.2.4. **Task 4**: Optimize for efficiency*

Based on our experience with SilkRoute (XML to SQL query reformulation) and with OQAFMA (StruQL to SQL) we fully expect that the response time for queries over our initial prototype implementations of the wrappers and VQP will be too slow for interactive use. Since efficiency of query processing is a major strength of the database community (and Dan Suciu in particular) we will take advantage of their work as much as possible. In particular, whenever possible we will reformulate queries as SQL in order to take advantage of the years of work that have gone into query optimization for relational databases.

We can already envision that a fractal semantic web that is entirely query based could be very slow, as queries must be propagated over multiple linked virtual views before they finally reach a materialized view. Thus, once we have implemented our architecture in prototype form we will implement several of the methods developed in aim 1 (section D.1), most likely in the order presented in aim 1: schema optimization, query optimization, and view composition. We will also experiment with caching of materialized views, but in this case will need to be sure that the cached materialized views are never changed except by re-running the query that materializes them against their associated VDL specifications.

*D.2.5. **Task 5**: Evaluate*

We will evaluate the various implementations of VQP by comparing VQP-generated materialized views (using the manually created VDL specifications in aim 1) with the validated ad hoc views created for our use cases in task 1 of aim 1. In each case the comparison will assess whether the VQP-generated view is as expressive as the ad hoc view, and whether it is as complete and concise. If the ad hoc views have been used in an application (such as scene generation in use case 1) we will substitute the generated view in the application as a practical test of whether the generated view is valid. Other parameters of evaluation will include the response time of the various query processors, the usefulness of the web service, the ease of incorporation into BioPortal, and other similar software engineering features.

D.2.6. Alternatives

As noted earlier, our overall goal will be to design the system so that a single view specification can be used to both generate materialized views and to process queries over non-materialized views. We also hope to implement only query-based interfaces in the long run. However, if further exploration shows that satisfying these goals is not completely feasible (as for example, we can't specify all the transformations we need in a single view specification, or if execution time is prohibitive for non-materialized views) we will pursue efforts to build systems for generating materialized views, as discussed in section D.1.8.

On the other hand it is possible that we will make better progress than anticipated. In this case we will explore bidirectional mappings so that we can experiment with alignment of two previously existing ontologies through a reference ontology. The development of such mappings is a particular area of expertise of the UW database group, as exemplified by the Piazza peer data management system [120].

*D.3. **Aim 3. Develop a graphical interface** that allows domain experts to specify target application ontologies from one or more source ontologies, without the need to understand the underlying mapping formalism.*

The problem we address in the final aim is how to develop methods that allow end-user domain experts to specify the view definition language (VDL) we will develop in aims 1 and 2. VDL will need to define all the transformations we describe in section D.1.2.1, and will need to do these over possibly more than one source ontology, and possibly over more than one instance of VQP. Thus, we expect that VDL will be at least as complex as RVL [40], the RDF view language that we will take as our starting point for the development of VDL. Based on our experience developing XQuery based views over relational databases [114] (described in section C.3) it seems highly unlikely that most domain experts will want to take the time to learn a complex query or view definition language. Thus, to make our view generation methods most useful we need to find ways to make it easy for users to specify a VDL view without having to learn the intricacies of the language. In order to do so we will develop a graphical user interface (GUI) that will allow the end-user to specify a view in VDL. We refer to this interface as VGen, (View Generator). VGen will be one of the "Editors and Tools" in the

Presentation layer of BioPortal, and it will access the various VQP web services in the Service layer. A prototype version of VGen will be designed and built by one of the UW students as a thesis project, under the supervision of Drs. Brinkley and Gennari, and in close consultation with Dr. Storey at University of Victoria. The prototype will be made more robust by Todd Detwiler, and integrated within BioPortal by Archana Vembakam at CBio.

D.3.1. Scenario

The following scenario, which will evolve during the course of the proposed work, will guide us as we develop VGen. On startup VGen will first present a list of available source ontologies, where a source ontology can be a reference ontology, a materialized application ontology, or a non-materialized view of one or more source ontologies (as possibly specified using a previous incarnation of VGen). All these types of source ontologies will be displayed in the same way in VGen. The user will select a set of ontologies to act as sources for the view. In the simplest example, corresponding to use case 1 in Figure 2, only one source ontology will be selected. In this case the source would be the FMA (R1 in the figure). The goal will then be to use VGen to define the VDL specification of view V1 needed to generate application ontology A1 (the target), which in this case would be cardiovascular anatomy. The basic idea will be to create a visual representation of the mappings between the source and the target. This representation could be a visual graph between nodes in the source and nodes in the target, where the edges represent the basic mapping types (such as extract, add, delete, rename). However, this basic notion is greatly complicated by the fact that a large source ontology like the FMA can never be visualized in its entirety. Therefore, ways must be found to zoom in on and select only those portions needed to specify the mappings. The mappings themselves will also be complex since many of them will involve sets of additions and deletions of triples.

Given a visual mapping of this sort, the system should then allow the user to observe the effect of the view definition by visualizing the resulting target ontology. Once the user is satisfied with the view he or she could then save the VDL specification under some meaningful name (say “Cardiovascular system anatomy” for A1 in Figure 2) and make it available as a web service in BioPortal. This view could then be made available as a source ontology within a new instance of VGen. An option would also be available to materialize the view and save it as a file for download.

The third use case in Figure 2, that of creating a view V2 from more than one source ontology, adds the complication of creating mappings from more than one source ontology (A1 and R2) to a single target ontology (A2). In this case the user would select both source ontologies from the menu of available sources, and would create mappings that involve edges from nodes in either or both the sources to the target. If, for example in use case 3, one of the target nodes should be “blood flow in the aorta”, a visual mapping might involve a line from the Aorta in source ontology A1 (cardiovascular anatomy) to “blood flow in the aorta” in the target ontology, as well as a line from “blood flow” in source ontology R2 (PRO) to the same node “blood flow in the aorta” in the target. Of course it would be tedious to specify all the flow variables for all blood vessels, so more complex mappings that involve inheritance would need to be represented.

The above scenario defines a set of research issues that must be addressed in order to implement VGen. These issues constitute the tasks for this aim.

*D.3.2. **Task 1**: Develop methods for visualizing and selecting subgraphs from large ontologies*

The problem in this task is how to visualize large and complex ontologies like the FMA, and how to select only the subgraph that needs to be included in the VDL view specification. This task is the visual equivalent of the query component of VDL (section D.1.2.1).

Although there are many techniques that have been developed to visualize ontologies, most of them only deal with small ontologies in which all or most of the nodes and edges can be visualized at once. However, these techniques do not scale up when dealing with an ontology the size of the FMA. Thus, ways must be found to display high level views of the ontology, and then allow the user to zoom in on those portions that are of interest, in the process selecting those portions that should be included in the view.

In addressing this problem we will be guided by Dr. Margaret Storey, who is an expert on human computer interaction and the developer of the Jambalaya program [121]. Jambalaya is a plug-in for Protégé that uses the concept of a nested graph to display a large ontology. The basic idea is that a hierarchy is displayed as boxes within boxes. Clicking on one of the boxes causes the system to zoom in on the selected box and display its children, and relationships among concepts are displayed as lines between boxes. This kind of display will be a starting point for our efforts to display large ontologies like the FMA.

However, as Dr. Storey has found, the nested box approach is not always the right visualization for the task. Thus, the current interest of her group is customization techniques -- how to visualize graphs to meet particular needs and how to use more knowledge about the user and task to filter what is shown. The emphasis is the use of visualizations that intelligently guide the user in performing specific tasks.

The other important Center effort that is relevant to this work is the Prompt tool by Noy [8], which, among other things, presents a graphical interface for extracting a subgraph of an ontology using view traversal. Although the method only permits traversal along a single link type (without regular expressions), it represents part of what we need to define complex mappings.

A related influence is our own Emily program described in the Preliminary Results [94]. This program presents a GUI that allows a user to specify individual view traversals in a similar manner to Prompt, and to combine the results of those traversals using Boolean operators.

We will also be guided by efforts in the database and ontology communities to graphically generate queries. One example is GRQL [122], which captures a user's browse history over an RDF ontology, then saves this history as a sort of semantic bookmark that can be played back.. There has also been considerable work in the database community to develop graphical interfaces, primarily for query formation over relational or XML databases. A good overview is in [122]. As part of our XBrain work we are investigating the use of QURSED [123] and XQBE [124] XQuery graphical generators for specifying queries over an XML view of a relational database.

D.3.2.1. Plan

We will begin with small materialized application ontologies that can be viewed in their entirety so that we can experiment with various existing ontology visualization tools.. As we scale up to larger ontologies we will use many of the ideas from Jambalaya and other applications in development by Dr. Storey's group, including dynamic techniques such as rapid expansion and contraction of subtrees, and other visual techniques, such as brushing and motion, which allow the amount of information displayed for exploration to be increased by an order of magnitude.

We will also explore these techniques with the understanding that purely visual approaches may not be enough to select the entire subgraph that needs to be included in the target ontology. For example, a feature that will be needed is the ability to essentially "lasso" a subgraph of the source ontology for inclusion in the target, perhaps along a single link type as can be done currently in Prompt, but later extending to virtual links expressed as regular expressions (using a popup visual regular expression editor). After several groups of such lassoed entities have been created an Emily-like interface could popup which would allow these lassoed groups to be combined using a Boolean expression

*D.3.3. **Task 2**: Develop methods for visually defining the mapping between large ontologies*

Given that visual (or hybrid visual/text-based) techniques are able to select the appropriate subgraph for inclusion in the application ontology, we next need to develop visual methods for specifying the view definition portion of VDL: those transformations that need to be made on the selected subgraph as it is included in the target. This task is again complicated not only by the different types of mappings that must be constructed, but also by the fact that not all of the source ontology can be visualized at once. As in the previous task we will be guided by Dr. Storey and her work with Jambalaya and intelligent visualization.

As in the previous task we will begin by exploring visual mapping techniques using smaller ontologies that can be visualized using existing ontology visualization tools, then gradually scaling up to larger ontologies. As one example, we will develop techniques that allow a copy of a lassoed portion of the source ontology (which itself might be a Boolean combination of other lassoed entities) to be dragged to the target ontology, leaving behind a set of lines from the source visualization to the target visualization. Zooming in on these lines might then permit some of them to be cut, thereby deleting portions of the copied source. Zooming in on the copy itself would allow relationships within the copied subgraph to be changed by adding, deleting and renaming links. All of these mapping operations would be recorded in an internal data structure for later compilation into VDL.

*D.3.4. **Task 3**: Develop methods for viewing the effect of mappings*

A related task to the problem of visualizing large ontologies is visualizing what the effect of various mappings would be on the generated application ontology. One approach would be to periodically materialize the application ontology, then visualize it using the same techniques developed for the first task. However, it should also be possible to visualize the generated application ontology “on-the-fly” as it is being created.

The latter capability is feasible if the visual mapping created in the previous task is compiled (perhaps interpreted) on the fly to a VDL specification, which is then coupled to an instance of the View Query Processor (VQP) to be developed in aim 2. Mouse clicks on visual representations of components of the target mapped ontology would then be reformulated by VGen into queries over the compiled view specification (V1 in Figure 2), which would in turn be reformulated by VQP into queries over the source (FMA in this case). The results would be passed back to the visual display where they would be converted to the most appropriate visual representation, depending on the context. It should be noted that this capability will only be possible if we can find ways in aim 2 to efficiently represent and process queries over non-materialized views, in aim 1 to concisely describe the mappings in VDL, and in aim 3 to reformulate the mouse clicks as queries to VQP. But the great advantage would be that the user would have near instant feedback as to whether the mappings were what he or she had in mind, and could quickly change the mappings if they were not.

*D.3.5. **Task 4**: Design and Implement VGen*

We will implement solutions to the previous tasks in successively more capable versions of VGen, our graphical view generator program. Figure 4 shows our current design for the architecture of VGen. VGen will be written in Java, and will be designed to run initially as a Webstart application, and later in whatever delivery mode is needed by the Center for integration into the presentation layer of BioPortal. The figure shows the use of VGen to generate application ontology A1 from reference ontology R1 (the FMA) in Figure 2. Once a source ontology R1 is chosen by the Controller it will be visualized by an instance of the Visualizer which, in response to user interaction, will call an instance of the Query Generator to formulate queries to the wrapper interface W1 over R1. Similarly, an instance of the Visualizer for the generated target ontology A1 will also issue queries via an instance of the Query Generator. These queries will be sent to an instance of VQP that takes as input the view V1 generated by VGen. VQP will in turn send reformulated queries to R1 via W1. The Mapper will be used to specify mappings between the visualized components of the source and target ontologies. These mappings will then be compiled by the View Compiler into the View V1, which will at each compile step represent the non-materialized view of A1. We will implement various versions of VGen that incorporate the individual modules at various stages in their development. The development of these modules will in turn depend on progress in each of the three previous tasks: the Visualizer will depend on task 1, the Mapper will depend on task 2, and the Query Generator will depend on task 3.

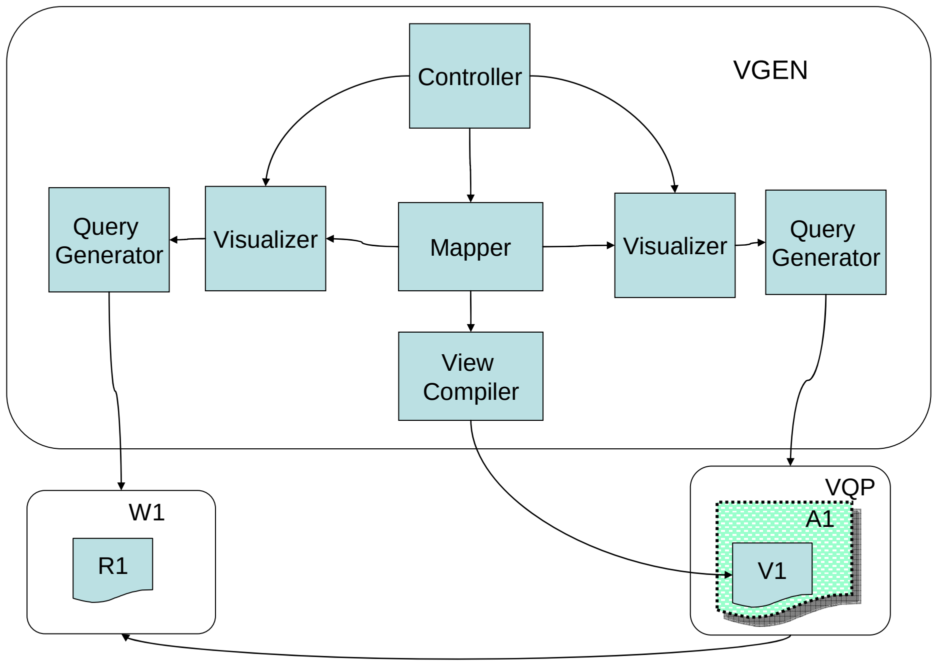

Figure 4 View Generator

*D.3.6. **Task 5**: Optimize for efficiency*

As is the case with VGen we will only optimize for rapid response time after our initial prototypes are shown to be valid. But as we see various bottlenecks we will endeavor to remove them. One possible module for optimization will be the Query Generator, which could be optimized in a similar manner to SQL query engines using the expertise of the UW database group. Optimizations of VQP and the wrappers in Aim 2 will also substantially influence the response time of VGen. If these cannot be optimized sufficiently we can always substitute materialized views.

*D.3.7. **Task 6**: Evaluate*

We will employ several criteria for evaluating various versions of VGen: 1) ability to regenerate the same VDL code generated manually in aim 2; 2) ability of the generated VDL code to regenerate the same materialized application ontologies we will generate in an ad hoc fashion in aim 1 (section D.1.1); 3) response time; and 4) usability, a big area, in which users will be asked to comment on the look and feel of the interface, the intuitiveness of the visual metaphors for mapping, and many other issues. The feedback we receive from the users in these areas will be used to design new versions of VGen. However, given that we will only be able to design prototypes, we do not expect to do widespread usability testing on large numbers of users. However, when these tools are deployed by the Center additional resources may become available for more widespread testing.

D.3.8. Alternate approaches

We recognize that many of the features we plan to implement may in fact not be feasible in the time period of the proposed work. Some of these possible obstacles and their workarounds are:

- Inability to graphically specify all the mappings needed by VGen. Allow the user to call up a text editor for editing the generated VDL directly, then save the resulting VDL specification in the same way VGen would do it. In the worst case, if none of the visual mapping techniques work we can simply create a set of saved VDL views that serve as templates. A user could load these into an editor, slightly tweak them, and save back out under a new name. This is exactly the feature that we currently use the most in our XBrain application described in the Preliminary Results (section C.3).

- Inability to dynamically visualize the target ontology as it is created. Allow the user to materialize the target, and then visualize it in the same way the source is visualized.
- Inability to scale up to very large ontologies like the FMA. This could be a major problem since no one has yet solved it. In our initial work we will follow the lead of others and work with scaled down versions. The resulting tools will still be useful for smaller scale ontologies, and will lay the groundwork for scaling up in later work.

D.4. Timeline

Aim	Task	Year 1	Year 2	Year 3	Year 4
1	1 Catalog and create ad hoc slices	X			
	2 Select representation/query/view languages	X			
	3 Extend languages	X			
	4 Query processing over a single view		X		
	5 Query processing over multiple views			X	
	6 Optimizing for efficiency				X
	7 Evaluate	X X X	X	X	X
2	1 Design system architecture	X			
	2 Implement Wrappers		X		
	3 Implement View Query Processor (VQP)		X	X	X
	4 Implement efficiency optimizations				X
	5 Evaluate	X	X X	X	X
3	1 Methods for visualization and selection	X	X	X	X
	2 Methods for visual mapping	X	X	X	X
	3 Methods for viewing the effect of mapping	X	X	X	X
	4 Implement View Generator (VGen)	X	X	X	X
	5 Implement optimizations				X
	6 Evaluate	X	X	X	X

Milestones are shown by X's in the above timeline, at approximately six month intervals during the four year period. All aims will be pursued in parallel. Personnel involved in aim 1 will be faculty from Stanford and the UW, research scientists from both institutions, and one graduate student. The milestones for aim 1 indicate the expected time by which the methods will be developed to the point where they can be implemented in aim 2. About a six month lag time from method development to implementation in aim 2 is expected. Evaluation will occur for all milestones. Primary personnel for aim 2 are the PI, one graduate student (the same as in aim 1), and the two primary implementers (Todd Detwiler at the UW, Archana Vembakam at Stanford). Milestones for aim 2 occur about six months after the corresponding methods have been developed in aim 1, and reflect different versions of the View Query Processor. Aim 3, design of the graphical View Generator (VGen) will be carried out by personnel from the UW, in consultation with Margaret Storey from University of Victoria. This will be the thesis project of the second graduate student. Optimization of prototypes developed by the student will be done by Todd Detwiler at the UW, and installation at Stanford will be done by Archana Vembakam. Milestones reflect improved versions of VGen in response to improved development of the associated methods.

D.5. Plans for dissemination

The primary product of this research will be a set of software tools for use in generating and accessing ontology views. As in all University of Washington sponsored projects the copyright for these tools will be owned by the University of Washington. However, as per our own goals, the goals of the National Center for Bioontology and the program announcement to which this proposal is a response, the tools we develop under this proposal will be released under an open source license. As part of the research effort (aim 2) we will work with Stanford to integrate these tools as a set of "plugins" for the BioPortal framework being developed by the Center. Integrating them in this way will ensure their widest availability, and is directly in line with the goal of the Center to widely disseminate the results of its work. In addition to integrating the tools within BioPortal we will also make them available as separate downloads if needed.

Since the tools will primarily be implemented as Internet services the primary commercial avenue would be for a company to sell services based on the tools, in a similar fashion to most open source companies. However, depending on the needs of commercial users we will also retain the possibility of releasing the tools under a dual licensing scheme, one open source and one commercial.

References

1. Rosse C, Mejino JLV. A reference ontology for bioinformatics: the Foundational Model of Anatomy. Journal of Bioinformatics. 2003;36(6):478-500 http://sigpubs.biostr.washington.edu/archive/00000135/.

2. Gennari JH, Musen MA, Fergerson RW, Grosso WE, Crubezy M, Eriksson H, Noy NF, Tu SW. The evolution of Protege: an environment for knowledge-based systems development. Int. J. Human-Computer Studies 2003;58(1):89-123 http://www.smi.stanford.edu/pubs/SMI_Reports/SMI-2002-0943.pdf.

3. Noy NF, Musen MA, Mejino JLV, Rosse C. Pushing the envelope: challenges in a frame-based representation of human anatomy. Data and Knowledge Engineering 2004;48:335-359.

4. Noy NF, Rubin DL, Musen MA. Making biomedical ontologies and ontology repositories work. IEEE Intelligent Systems 2004;19(6):78-81 http://smi-web.stanford.edu/people/noy/papers/ontologyLibraries.pdf.

5. Dameron O, Noy NF, Knublauch H, Musen MA. Accessing and manipulating ontologies using web services. In: McIlraith SA, Plexousakis D, Van Harmelen F, editors. The Semantic Web - IWSC 2004: Third International Semantic Web Conference, Nov 7-11. Lecture Notes in Computer Science. Hiroshima, Japan: Springer-Verlag; 2004. http://smi-web.stanford.edu/people/noy/papers/dameron2004iswc.pdf.

6. Noy NF, Musen MA. Specifying ontology views by traversal. In: McIlraith SA, Plexousakis D, Van Harmelen F, editors. The Semantic Web - IWSC 2004: Third International Semantic Web Conference, Nov 7-11. Lecture Notes in Computer Science. Hiroshima, Japan: Springer-Verlag; 2004. p. 713. http://sig.biostr.washington.edu/share/sigweb/pubs/NoyViews2004.pdf.

7. Dameron O, Rubin DL, Musen MA. Challenges in converting frame-based ontology into OWL: the Foundational Model of Anatomy case-study. In: Proceeding, American Medical Informatics Association Fall Symposium. Washington, DC; 2005. p. 181-185. http://sig.biostr.washington.edu/share/sigweb/pubs/DameronAMIA2005.pdf.

8. Noy NF, Musen MA. The PROMPT suite: Interactive tools for ontology merging and mapping. Int. J. Human-Computer Studies 2003;59(6):983-1024 http://smi-web.stanford.edu/pubs/SMI_Reports/SMI-2003-0973.pdf.

9. Noy NF, Musen MA. PROMPT: Algorithm and tool for automated ontology merging and alignment. In: Seventeenth National Conference on Artificial Intelligence (AAAI-2000); 2000; Austin, TX; 2000. p. 450--455.

10. Storey MA, Muller H, Wong K. Manipulating and documenting software structures. In: Software Visualization: World Scientific Publishing Co.; 1996. p. 244-263.

11. Storey MA, Musen MA, Silva J, Best C, Ernst N, Fergerson RW, Noy NF. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. In: Workshop on

Interactive Tools for Knowledge Capture, K-CAP-2001. Victoria, Canada; 2001. http://www.cs.uvic.ca/~mstorey/papers/kcap2001.pdf.

12. Bull RI, Favre J-M. Visualization in the context of model driven engineering. In: MODELS 2005: Model Driven Development of Advanced User Interfaces; 2005. http://www.thechiselgroup.org/files/mddaui.pdf.

13. IBM. Eclipse Integrated Development Environment. http://www.eclipse.org/; 2005.

14. World Wide Web Consortium. W3C Semantic Web Health Care and Life Sciences Interest Group. http://www.w3.org/2001/sw/hcls/; 2005.

15. Bassingthwaighte JB, Qian H, Li Z. The Cariome Project: An integrated view of cardiac metabolism and regional mechanical function. Adv Exp Med Biol 1999;471:541-553.

16. Bassingthwaighte JB. Strategies for the Physiome Project. Ann Biomed Eng 2000;28:1043-1058.

17. University of Washington Department of Bioengineering. National Simulation Resource. http://nsr.bioeng.washington.edu/; 2005.

18. Raymond GG, Butterworth E, Bassingthwaighte JB. JSIM: Free software package for teaching physiological modeling and research. Exper Biol 2003;280.5:102.

19. Bassingthwaighte JB, Li Z, Qian H. Blood flows and metabolic components of the cardiome. Prog Biophys Mol Biol 1998;69:445-461.

20. White RJ, Bassingthwaighte JB, Charles JB, Kushmerick MJ, Newman DJ. Issues of exploration: human health and well being during a mission to Mars. Adv Space Res 2003;31:7-16.

21. Bassingthwaighte JB, Chizeck HJ, Atlas LE, Qian H. Strategies and tactics in multiscale modelling of cell-to-organ systems. IEEE Proc In Press. 2006.

22. Hunter PJ, Borg TK. Integration from proteins to organs: the Physiome project. Nature Reviews: Molecular Cell Biology 2003;4:237-243 http://sig.biostr.washington.edu/share/sigweb/pubs/Hunter2003a.pdf.

23. Federation of American Scientists. The Digital Human. http://www.fas.org/dh/; 2002.

24. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin, II, Hedley WJ, Hodgman TC, Hofmeyr JH, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novere N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 2003;19(4):524-31 http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=12611808.

25. CellML. CellML. http://www.cellml.org; 2005.

26. Wang X, Gorlitsky R, Almerida JS. From XML to RDF: how semantc web technologies will change the design of 'omic' standards. Nature Biotechnology 2005;23:1099-1103 http://www.nature.com/nbt/journal/v23/n9/pdf/nbt1139.pdf.

27. Berners-Lee T, Hendler J, Lassila O. The semantic web. Scientific American 2001;284(5):34-43.

28. Hendler J. Agents and the semantic web. IEEE Intelligent Systems 2001;16(2):30-37.

29. World Wide Web Consortium. The Semantic Web. http://www.w3.org/2001/sw/; 2002.

30. World Wide Web Consortium. Resource Description Framework (RDF). http://www.w3.org/RDF/; 2005.

31. World Wide Web Consortium. The OWL Web Ontology Language. http://www.w3.org/TR/owl-features/; 2005.

32. Musen MA. The National Center for Biomedical Ontology. http://bioontology.org/Center_Overview.pdf; 2005.

33. Dublin Core Metadata initiative. Home page. http://dublincore.org/; 2005.

34. OBO. Open Biomedical Ontologies. http://obo.sourceforge.net; 2005.

35. Smith B. New desiderata for biomedical terminologies. J. Biomedical Informatics. In Press. 2005 http://sig.biostr.washington.edu/share/sigweb/pubs/SmithDesiderata2005.pdf.

36. Zhang S, Bodenreider O. Law and order: Assessing and enforcing compliance with ontological modeling principles in the Foundational Model of Anatomy. Computers in Biology and Medicine. In Press 2005 http://sig.biostr.washington.edu/share/sigweb/pubs/ZhangOntoPrinciples2005.pdf.

37. Cook DL, Mejino JLV, Rosse C. Evolution of a foundational model of physiology: symbolic representation for functional bioinformatics. In: MEDINFO 2004. San Francisco: IOS Press; 2004. p. 336-340.

38. Smith B, Kumar A, Ceusters W, Rosse C. On carcinomas and other pathological entities. Comparative and functional genomics. In Press 2005 http://sig.biostr.washington.edu/share/sigweb/pubs/SmithPathology2005.pdf.

39. Rosse C, Kumar A, Mejino JLV, Cook DL, Detwiler LT, Smith B. A strategy for improving and integrating biomedical ontologies. In: Proceedings, AMIA Fall Symposium. Washington, D.C.; 2005. p. 639-643. http://sigpubs.biostr.washington.edu/archive/00000176/.

40. Magkanaraki A, Tannen V, Christophides V, Plexousakis D. Viewing the semantic web through RVL lenses. In: Second International Semantic Web Conference. Sanibel Island, Florida: Springer-Verlag; 2003. p. 96-112. http://sig.biostr.washington.edu/share/sigweb/pubs/MagkanarakiLense2004.pdf.

41. Volz R, Oberle D, Studer R. Implementing views for light-weight web ontologies. In: IEEE Database Engineering and Application Symposium (IDEAS). Hong Kong, China; 2003. http://sig.biostr.washington.edu/share/sigweb/pubs/VolzViews2003.pdf.

42. Miklos Z, Neumann G, Zdun U, Sintek M. Querying semantic web resources using TRIPLE views. In: Second International Semantic Web Conference. Sanibel Island, Florida; 2003.

43. Brinkley JF. The use of anatomical knowledge in medical imaging: an overview of the University of Washington Structural Informatics Group. In: Wong S, editor. Advances in Biomedical Image Databases. Boston: Kluwer Academic Press; 1998. p. 89-116.

44. Brinkley JF. Structural informatics and its applications in medicine and biology. Academic Medicine 1991;66(10):589-591 http://sig.biostr.washington.edu/overview/si.html.

45. Rosse C, Ben Said M, Eno KR, Brinkley JF. Enhancements of Anatomical Information in UMLS Knowledge Sources. In: Proceedings, 19th Annual Symposium on Computer Applications in Medical Care. New Orleans; 1995. p. 873-877.

46. Rosse C, Mejino JL, Modayur BR, Jakobovits RM, Hinshaw KP, Brinkley JF. Motivation and organizational principles for anatomical knowledge representation: the Digital Anatomist symbolic knowledge base. Journal of the American Medical Informatics Association 1998;5(1):17-40 http://sig.biostr.washington.edu/publications/online/KBpaper.pdf.

47. Rosse C, Shapiro LG, Brinkley JF. The Digital Anatomist foundational model: principles for defining and structuring its concept domain. In: Proceedings, American Medical Informatics Association Fall Symposium. Orlando, Florida; 1998. p. 820-824. http://sig.biostr.washington.edu/publications/online/D005094.pdf.

48. Brinkley JF, Eno K, Sundsten JW. Knowledge-based client-server approach to structural information retrieval: the Digital Anatomist Browser. Computer Methods and Programs in Biomedicine 1993;40:131-145.

49. Agoncillo AV, Mejino JLV, Rickard KL, Detwiler LT, Rosse C. Proposed classification of cells in the Foundational Model of Anatomy. In: Proceedings, AMIA Fall Symposium; 2003. p. 775.

50. Martin RF, Mejino JLV, Bowden DM, Brinkley JF, Rosse C. Foundational model of neuroanatomy: implications for the Human Brain Project. In: Proc AMIA Annu Fall Symp. Washington, DC; 2001. p. 438-442. http://sigpubs.biostr.washington.edu/archive/00000067/.

51. Martin RF, Rickard K, Mejino JLV, Agoncillo AV, Brinkley JF, Rosse C. The evolving neuroanatomical component of the foundational model of anatomy. In: Proceedings, AMIA Fall Symposium; 2003. p. 927. http://sigpubs.biostr.washington.edu/archive/00000139/.

52. Michael J, Mejino JL, Rosse C. The role of definitions in biomedical concept representation. In: JAMIA Fall Symposium Supplement; 2001. p. 463-467.

53. Mejino JL, Rosse C. Conceptualization of anatomical spatial entities in the Digital Anatomist Foundational Model. In: Proceedings, American Medical Informatics Association Fall Symposium. Washington, D.C.; 1999. p. 112-116.

54. Mejino JLVJ, Rosse C. Symbolic modeling of structural relationships in the Foundational Model of Anatomy. In: Proceedings, First International Workshop on Formal Biomedical Knowledge Representation (KR-MED 2004). Whistler Mountain, Canada; 2004.

55. Neal PJ, Shapiro LG, Rosse C. The Digital Anatomist spatial abstraction: a scheme for the spatial description of anatomical features. In: Proceedings, American Medical Informatics Association Fall Symposium. Orlando, Florida; 1998. p. 423-427.

56. Mejino JLV, Agoncillo AV, Rickard KL, Rosse C. Representing complexity in part-whole relationships within the Foundational Model of Anatomy. In: Proceedings, Fall AMIA Symposium; 2003. p. 450-454.

57. Rosse C. Terminologia Anatomica; considered from the perspective of next-generation knowledge sources. Clinical Anatomy 2000;14:120-133 http://sig.biostr.washington.edu/publications/online/CRTAnat.pdf.

58. Rickard KL, Mejino JLV, Martin RF, Agoncillo AV, Rosse C. Problems and solutions with integrating terminologies into evolving knowledge bases. In: Proceedings, MEDINFO 2004. San Francisco; 2004. p. 420-424. http://sigpubs.biostr.washington.edu/archive/00000160/.

59. Agoncillo AV, Mejino JLV, Rosse C. Influence of the Digital Anatomist Foundational model on traditional representations of anatomical concepts. In: Proceedings, American Medical Informatics Association Fall Symposium. Washington, D.C.; 1999. p. 2-6.

60. Mejino JLV, Rosse C. The potential of the Digital Anatomist foundational model for assuring consistency in UMLS sources. In: Proceedings, American Medical Informatics Association Fall Symposium. Orlando, Florida; 1998. p. 825-829.

61. Gennari JH, Silberfein A, Wiley JC. Integrating genomic knowledge sources through an anatomy ontology. In: Proceedings, Pacific Symposium on Biocomputing; 2005. p. 115-126. http://faculty.washington.edu/gennari/papers/gennari-PSB-05.pdf.

62. GeneOntologyConsortium. Creating the gene ontology resource: design and implementation. Genome Res 2001;11(8):1425-33. http://www.ncbi.nlm.nih.gov/cgi-bin/Entrez/referer?http://www.genome.org/cgi/content/full/11/8/1425.

63. Blake J, Richardson J, Bult C, Kadin J, Eppig J. MGD: The Mouse Genome Database. Nucleic Acids Research 2003;31:193-195.

64. Travillian RS, Rosse C, Shapiro LG. An approach to the anatomical correlation of species through the Foundational Model of Anatomy. In: Proceedings, Fall AMIA Symposium; 2003. p. 669-673.

65. Travillian RS, Gennari JH, Shapiro LG. Of mice and men: Design of a comparative anatomy information system. In: Proceedings, Fall AMIA Synmposium; 2005. p. 734-738.

66. Smith B, Kohler J, Kumar A. On the application of formal principles to life sciences data: a case study in the Gene Ontology. In: DILS 2004: Data Integration in the Life Sciences; 2004. p. 124-139.

67. Zhang S, Bodenreider O. Alignment of multiple ontologies of anatomy: deriving indirect mappings from direct mappings to a reference. In: Proceedings, AMIA Fall Symposium; 2005. p. 864-868.

68. Mork P, Bernstein PA. Adapting a generic match algorithm to align ontologies of human anatomy. In: ICDE 2004; 2004. p. 787-790. http://www.informatik.uni-trier.de/%7Eley/db/conf/icde/icde2004.html#MorkB04.

69. Bean CA, Rindflesch TC, Sneiderman CA. Automatic semantic interpretation of anatomical spatial relationships in clinical text. In: Proceedings, AMIA Fall Symposium; 1998. p. 897-901.

70. Bodenreider O, Burgun A. Characterizing the definitions of anatomical concepts in WorldNet and specialized sources. In: Proc First Global WorldNet Conf 2002. p. 223-230.

71. Zhang L, Perl Y, Halper M, Geller J. Designing metaschemas for the UMLS enriched semantic network. J. Biomedical Informatics 2003;36:433-449 http://portal.acm.org/citation.cfm?id=1008562.

72. Beck R, Schulz S. Logic-based remodeling of the Digital Anatomist Foundational Model. In: Proceedings, AMIA Fall Symposium; 2004. p. 71-75.

73. Rindflesch TC, Bean CA, Sneiderman CA. Argument identification for arterial branching predications asserted in cardiac catheterization reports. In: Proceedings, AMIA Fall Symposium; 2000. p. 704-708. http://lhncbc.nlm.nih.gov/lhc/docs/published/2000/pub2000018.pdf.

74. Sneiderman CA, Rindflesch TC, Bean CA. Identification of anatomical terminology in medical text. In: Proceedings, AMIA Fall Symposium; 1998. p. 428-432.

75. Tringali M, Hole WT, Srinivasan S. Integration of a standard gastrointestinal endoscopy terminology in the UMLS Metathesaurus. In: Proceedings, AMIA Fall Symposium; 2002. p. 801-805.

76. Dameron O, Gibaud B, Morandi X. Numeric and Symbolic knowledge representation of cerebral cortex anatomy: methods and preliminary results. Surg Radiol Anat 2004;26:191-197.

77. Kumar A, Yip YL, Smith B, Marwede D, Novotny D. An ontology for carcinoma classification for clinical bioinformatics. Stud Health Technol Inform. 2005;116:635-640.

78. Smith B, Mejino JLV, Schultz S, Kumar A, Rosse C. Anatomical information science. In: Cohn AG, Mark DM, editors. Spatial Information Theory: Proceedings, International Conference, COSIT 2005. Ellicottville, NY, Sept 14-18; 2005. p. 149.

79. Smith B, Ceusters W, Klagges B, Kohler J, Kumar A, Lomax J, Mungall C, Neuhaus F, Rector AL, Rosse C. Relations in biomedical ontologies. Genome Biology 2005(6):R46.

80. Donnelly M, Bittner T, Rosse C. A formal theory for spatial representation and reasoning in biomedical ontologies. Artificial Intelligence in Medicine. In Press 2005.

81. Rubin DL, Bashir Y, Grossman D, Dev P, Musen MA. Using an ontology of human anatomy to inform reasoning with geometric models. Stud Health Technol Inform 2005;111:429-435.

82. Dickson S, Pouchard L, Ward R, Atkins G, Cole M, Lorensen B, Ade A. Linking Human Anatomy to knowledge bases: A visual front end for electronic medical records. Stud Health Technol Inform. 2005;111:586-591.

83. Brinkley JF, Prothero JS, Prothero JW, Rosse C. A framework for the design of knowledge-based systems in structural biology. In: Proceedings, 15th Annual Symposium on Computer Applications in Medical Care. Baltimore, Maryland; 1989. p. 61-65.

84. Brinkley JF, Rosse C. The Digital Anatomist distributed framework and its applications to knowledge based medical imaging. Journal of the American Medical Informatics Association 1997;4(3):165-183.

85. Brinkley JF, Wong BA, Hinshaw KP, Rosse C. Design of an anatomy information system. Computer Graphics and Applications 1999;19(3):38-48 http://sigpubs.biostr.washington.edu/archive/00000024/.

86. Brinkley JF, Rosse C. Requirements for an on-line knowledge-based anatomy information system. In: Proceedings, American Medical Informatics Association Fall Symposium. Orlando, Florida; 1998. p. 892-896.

87. Stalder DS, Brinkley JF. The Digital Anatomist Foundational Model Server. In: Perl Conference 3.0. Monterey, California; 1999. http://sig.biostr.washington.edu/share/pubs/Stalder1999.pdf.

88. Mork P, Brinkley JF, Rosse C. OQAFMA Querying Agent for the Foundational Model of Anatomy: a prototype for providing flexible and efficient access to large semantic networks. J. Biomedical Informatics 2003;36(6):501-517 http://sigpubs.biostr.washington.edu/archive/00000136/.

89. Fernandez M, Florescu D, Levy H, Suciu D. A query language for a web site management system. SIGMOD Record 1997;26(3):4-11.

90. Structural Informatics Group. Online demos. http://sig.biostr.washington.edu/products/demos; 2005.

91. Detwiler L, Mejino J, Rosse C, Brinkley J. Efficient Web-Based Navigation of the Foundational Model of Anatomy. Proc AMIA Symp 2003:829.

92. Distelhorst G, Srivastava V, Rosse C, Brinkley JF. A prototype natural language interface to a large complex knowledhge base, the Foundational Model of Anatomy. In: AMIA Fall Symposium.; 2003. p. 200-204. http://sigpubs.biostr.washington.edu/archive/00000138/.

93. Distelhorst GM, Hinshaw KP, Rosse C, Brinkley JF. An improved natural language interface to a complex anatomical knowledge base. In: Proceedings, MEDINFO. San Francisco; 2004. p. 1575. http://sigpubs.biostr.washington.edu/archive/00000143.

94. Shapiro LG, Chung E, Detwiler LT, Mejino JLV, Agoncillo AV, Brinkley JF, Rosse C. Processes and problems in the formative evaluation of an interface to the foundational model of anatomy. JAMIA 2005;12:35-46 http://sigpubs.biostr.washington.edu/archive/00000168/.

95. Lober WB, Brinkley JF. A portable image annotation tool for web-based anatomy atlases. In: Proceedings, American Medical Informatics Association Fall Symposium. Washington, D.C.; 1999. p. 1108.

96. Brinkley JF, Jakobovits R, Rosse C. An online image management system for anatomy teaching. In: Proc. AMIA Fall Symposium; 2002. p. 983. http://sigpubs.biostr.washington.edu/archive/00000120/.

97. Brinkley JF, Bradley SW, Sundsten JW, Rosse C. The Digital Anatomist information system and its use in the generation and delivery of Web-based anatomy atlases. Computers and Biomedical Research 1997;30:472-503 http://www9.biostr.washington.edu/da.html.

98. Wong BA, Rosse C, Brinkley JF. Semi-automatic scene generation using the Digital Anatomist Foundational Model. In: Proceedings, American Medical Informatics Association Fall Symposium. Washington, D.C.; 1999. p. 637-641.

99. Warren W, Brinkley JF. Knowledge-based, interactive, custom anatomical scene creation for medical education: the Biolucida system. In: Proceedings, AMIA Fall Symposium. Washington, D.C.; 2005. p. 789-793.

100. Prothero JS, Prothero JW. A software package in C for interactive 3D reconstruction and display of anatomical objects from serial section data. In: NCGA Proceedings; 1989. p. 187-192.

101. Nadeau DR. Building virtual worlds with VRML. IEEE Computer Graphics and Applications 1999;19(2):18-29.

102. Brinkley JF, Myers LM, Prothero JS, Heil GH, Tsuruda JS, Maravilla KR, Ojemann GA, Rosse C. A structural information framework for brain mapping. In: Koslow SH, Huerta MF, editors. Neuroinformatics: An Overview of the Human Brain Project. Mahwah, New Jersey: Lawrence Erlbaum; 1997. p. 309-334.

103. Jakobovits RM, Rosse C, Brinkley JF. An open source toolkit for building biomedical web applications. J Am Med Ass. 2002;9(6):557-590 http://sigpubs.biostr.washington.edu/archive/00000134/.

104. Fong C, Rosse C, Clark JI, Shapiro L, Brinkley JF. An ontology-based image repository for a biomedical research lab. In: MEDINFO 2004; 2004. p. 1598. http://sigpubs.biostr.washington.edu/archive/00000144/.

105. Fong C, Brinkley JF. Customizable Electronic Laboratory Online (CELO): A web-based data management system builder for biomedical research laboratories. http://sig.biostr.washington.edu/projects/celo/; 2005.

106. Li H, Brinkley JF, Gennari J. Semi-automatic database design for neuroscience experiment management systems. In: Proceedings, MEDINFO. San Francisco, CA.; 2004. p. 1639. http://sigpubs.biostr.washington.edu/archive/00000145/.

107. Gennari JH, Mork P, Li H. Knowledge transformations between frame systems and RDB systems. In: 3rd International Conference on Knowledge Capture (K-CAP '05); 2005; Banff, Canada; 2005. p. 197-198.

108. Tang Z, Kadiyska Y, Li H, Suciu D, Brinkley JF. Dynamic XML-based exchange of relational data: application to the Human Brain Project. In: Proceedings, Annual Fall Symposium of the American Medical Informatics Association. Washington, D.C.; 2003. p. 649-653. http://sigpubs.biostr.washington.edu/archive/00000141/.

109. Tang Z, Kadiyska Y, Suciu D, Brinkley JF. Results visualization in the XBrain XML interface to a relational database. In: Proceedings, MEDINFO. San Francisco, CA; 2004. p. 1878. http://sigpubs.biostr.washington.edu/archive/00000146/.

110. Fernandez M, Kadiyska Y, Morishima A, Suciu D, Tan W. Silkroute: a framework for publishing relational data in XML. ACM Transactions on Database Technology 2002;27(4).

111. Moore E, Poliakov A, Brinkley JF. Brain visualization in Java3D. In: Proceedings, MEDINFO. San Francisco, CA; 2004. p. 1761. http://sigpubs.biostr.washington.edu/archive/00000151/.

112. Tarczy-Hornoch P, Brinkley J, Shaker R, Mork P, Donelson L, Barrier M, Gennari J, Rosse C, Rossini A, Suciu D, Halevy A. Bio-PDMS: A structural ontology driven peer data management system (PDMS) for biology. In: Biomedical Information Science and Technology Initiative (BISTI) Symposium. Washington, DC; 2003.

113. Wang K, Tarczy-Hornoch P, Shaker R, Mork P, Brinkley JF. BioMediator Data Integration: Beyond Genomics to Neuroscience Data. In: Proceedings, AMIA Fall Symposium; 2005. p. 779-783.

114. Bales N, Brinkley J, Lee ES, Mathur S, Re C, Suciu D. A framework for XML-based integration of data, visualization and analysis in a biomedical domain. In: Proceedings, Third International XML Database Symposium (XSym 2005). Trondheim, Norway; 2005. p. 207-221. http://sigpubs.biostr.washington.edu/archive/00000178/.

115. Fikes R, Hayes P, Horrocks I. OWL-QL - A language for deductive query answering on the semantic web. http://ksl.stanford.edu/KSL_Abstracts/KSL-03-14.html; 2003.

116. Uceda-Sosa R, Chen CX, Claypool KT. CLOVE: A framework to design ontology views. In: ER 2004; 2004. p. 844-849.

117. Staab S, Studer R, editors. Handbook on Ontologies: Springer; 2004.

118. Haase P, Broekstra J, Eberhart A, Volz R. A comparison of RDF query languages. http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/rdfquery.pdf; 2004.

119. World Wide Web Consortium. SparQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/; 2005.

120. Halevy AY, Ives ZG, Madhavan J, Mork P, Suciu D, Tatarinov I. The Piazza peer data management system. IEEE Transactions on Knowledge and Data Engineering 2004;16:787-798 http://sig.biostr.washington.edu/share/sigweb/pubs/HalevyPiazza2004.pdf.

121. Storey M, Musen M, Silva J, Best C, Ernst N, Fergerson R, Noy N. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. In: Workshop on Interactive Tools for Knowledge Capture; 2001.

122. Athanasis N, Christophides V, Kotzinos D. Generating on the fly queries for the semantic web: the ICS-FORTH graphical RQL interface (GRQL). In: IWSC 2004: Third International Semantic Web Conference. Hiroshima, Japan; 2004. p. 486.

123. Petropoulos M, Papakonstantinou Y, Vassalos V. Graphical query interfaces for semistructured data: the QURSED system. ACM Transactions on Internet Technology 2005;5(2):390-438.

124. Braga D, Campi A, Ceri S. XQBE (XQuery By Example): A visual interface to the standard XML query language. ACM Transactions on Database Systems 2005;30(2):398-443.