Ontology Design Choices

A number of specific questions about the VIVO core ontology revolve around 2 quite different philosophies of ontology design and implementation, both of which have merits.

There have been several discussions in the first months of the project indirectly about these issues - including questions of modeling time and provenance, as well as general discussion of granularity.

See also the discussion of Context Nodes from the Mesur project, an approach similar to Option B.

Option A - Model real things and make only direct relationships	**Option B - Model relationships and secondary information objects as individuals in their own right**
Classes: Make classes reflecting only very tangible things that everyone will intuitively understand: *Person, Event, Organization, Document	Classes: Create additional classes to hold information inherent to the relationship among the individuals of primary interest: *Position, to model the type, time window, and title of a Person's connection with an Organization. *Authorship, to connect a Publication with a Person and track author order. *Educational Background, to hold year, major field, institution, and type of degree for each degree a Person holds. *Activity Description, to hold the role, time window of a person's involvement in a collaboration, professional service activity such as being editor of a journal, optionally including relationships to another Person, a Grant, Team, Project, or Publication.
Object properties: Express everything about a relationship, with no additional information. *e.g., Person *facultyMemberIn* Department. *then, Person *formerFacultyMemberIn* Department if you want to continue to maintain that information - have to move people from one property to another if the relationship changes. *Person *memberOf* Organization with no start ore end date; either have to make an individual for the organization or just use a data property with the information as unstructured text. *Publication has Author Person (with no author rank - authors would be sorted by last name, although a separate data property on the publication could list the authors in rank order if available from the source citation)	Object properties: *connect a Person with a "context" individual representing the full context of the relationship and hence itself have data properties such as start date, end date, title, role, related organization name (if not the organization is not entered into the system as an individual) or an object property linking to an Organization. *rely on bridging this intermediate context "hop" when rendering pages or querying - Person has Position (1 hop) and Position is in Department (2nd hop); Person has Authorship with an author rank value (1 hop) and then the Authorship belongs to a Publication (2nd hop).
Data properties: carry much more information as unstructured text. *educational background listing same information but not able to be queried by degree or institution. *professional activities list organization name and years of involvement, but not in any way that enforces consistent names, time linkages, or even format. *possible to configure editor to encourage bulleted lists or limit entries to 1 value per statement.	Data properties: *remain much simpler, single-value text or numbers or dates or identifiers. *contain much less formatting - formatting is done through per-class rendering code that can give a more polished appearance.
Reasoning: *direct and simpler; hence more tractable and likely to scale better, but much of the information will be stored in data properties where it is inaccessible for either analysis or logical reasoning. *keeps the ontology as a logical representation of the world in a by-necessity simple application rather than asking it to double as the data model for a more complex application.	Reasoning: *can be used in combination with rules to insert and remove direct relationships, requiring no human intervention (e.g., insert a direct worksln property between a Person and a Department when a Position has no end date or when the end date is in the future; when an end date is specified and it's in the past, remove the direct relationship again. *models the data you want to manage to help you manage it - treats the ontology more like a data model than a logical representation of the world.
Linked data or SPARQL queries for sharing data: *easier to map to standard ontologies *harvested RDF simpler to interpret	Linked data or SPARQL queries for sharing data: *harvested RDF may be able to be made just as simple by inferred direct properties and/or mappings *SPARQL queries can be written to flatten the data for consumption externally.
code: *less need for custom forms and custom rendering.	code: *more programming to trigger rule-based review of temporally-dependent data *more "custom rendering" - per-class or per-property display changes
Advantages: *easier to understand and document. *better fit with simpler ontologies such as FOAF. *follows conventions and relies less on Vitro-specific annotations and code.	Advantages: *provides cleaner data and relationships to analyze and visualize. *allows for a temporal qualifiers so can maintain historical information where it's important enough to specifically model. *provides a place to put additional provenance information. *makes it more possible for VIVO as an application to provide custom displays.
Disadvantages: *less able to support requests for display changes (e.g., ordering authors or educational background information, options to show or not show publications on a person's profile) because there is no ability to store display-related information. *therefore more generic display	Disadvantages: *steeper learning curve *more Vitro-specific annotations.