Infrastructure: Protege and OpenSimulator

Aaron Berkowitz

Progress: Earlier goals for this project included exploring the possibility of integrating Protege defined ontologies in OpenSimulator in a bidirectional way, that is, by mapping objects in a virtual world to various classes in an ontology such that external ontology tools could draw conclusions from information contained in the virtual world, and objects in a virtual world could draw on semantic functionality in scripting. There were two proposed ways of doing this. One was to implement a fairly lightweight melding of the database tables utilized by the two systems, such that database queries from either system could be used without modifying those queries in the originating system. However, since the Protege system does not have a canonical database layout, such a melding proves to be impossible. The other method was to develop scripts which held the data meaningful to the ontology, and place those scripts on the objects in the virtual world, which could the be accessed by the external ontology tool using standard software communication protocols. However, this method would also have to be customized for the individual ontology, such that there appears to be no good way of doing this in a generic fashion. A workaround to this problem would be to create a module or library that could be loaded into OpenSimulator which would expose the basic Protege API to scripts, and, possibly, allow the automation of creation of scripts meaningful to a given ontology that was loaded into the system. To this end I've explored how to create a module for OpenSimulator and a library of functions that can be accessed from scripts. Of course, given the deep similarity between C#, the language of OpenSimulator, and Java, the language of Protege, creating a library of C# functions that simply wrapped the corresponding Protege objects and method calls, and passed them via http or other standard communication protocol to the Protege system, would be fairly simple.

Plans: Since then, I've determined that more conceptual work needs to go into exploring exactly how an ontology could be productively incorporated into the current hospital project and into the larger Everything is Alive vision, rather than how the Protege system can be incorporated with
OpenSimulator, abstracted from any particular conception of use. To this end I plan on exploring how one might use already existing medical ontologies to improve the simulation aspects of the hospital project. One possibility involves using an existing medical ontology to aid in the creation of workflows. In particular, by building an ontology of standard hospital procedures and conjoining it with an existing medical ontology, it should be possible for a reasoner to derive the necessary steps in the workflow. Once we had a representation of the workflow in an ontology, it should be possible to have a program that could convert this into a series of scripts which could be loaded into OpenSimulator. Even more interesting, if the bots in the virtual hospital had access to the ontology, then that behavior could be generated dynamically, essentially attempting to mimic agent understanding of a domain within the virtual world. This sort of idea still needs to be generalized in a way that would be useful for the Everything is Alive project. Governing this is the question of whether actions can be cleanly modeled in an ontology along with ordinary sorts of objects. One of the concerns is that the standard OWL ontologies allow for an instance falling under multiple classes, hence, a naïve translation of those action instances into script behaviors might open us up to the sorts of semantic ambiguities that occur in programming languages which allow multiple inheritance. Additional avenues of exploration involve determining whether alternative ontology tools might better suit our needs. Expected Deliverable: A final paper, approximately twenty to thirty pages, outlining some of these potential uses of an ontology and assessing their feasibility.