A Common Ground for Virtual Humans: Using an Ontology in a Natural Language Oriented Virtual Human Architecture

Arno Hartholt, Thomas Russ, David Traum, Eduard Hovy, Susan Robinson
University of Southern California's Institute for Creative Technologies & Information Sciences Institute
Marina del Rey, CA 90292
United States of America
{hartholt, traum, robinson}@ict.usc.edu & {russ, hovy}@isi.edu

Abstract

When dealing with large, distributed systems that use state-of-the-art components, individual components are usually developed in parallel. As development continues, the decoupling invariably leads to a mismatch between how these components internally represent concepts and how they communicate these representations to other components: representations can get out of synch, contain localized errors, or become manageable only by a small group of experts for each module. In this paper, we describe the use of an ontology as part of a complex distributed virtual human architecture in order to enable better communication between modules while improving the overall flexibility needed to change or extend the system. We focus on the natural language understanding capabilities of this architecture and the relationship between language and concepts within the entire system in general and the ontology in particular.

1. Introduction

Designers of large heterogeneous systems (such as task-oriented communicating agents) have an uncomfortable choice to make regarding their knowledge representations: should they choose a uniform representation for all modules that enforces common understanding and re-use, or should they allow each module to use its own representation, tailored specifically for that module? Either alternative includes a set of difficult and perhaps insoluble problems. In the former case, using a single common representation, it may be very difficult to decide which representation to use, given the different demands of such diverse processes as planning, perception in a real or virtual world, and natural language dialogue, and especially since the ways in which they will be developed are not fully understood at the start. Should one choose an impoverished language for which one can guarantee fast algorithmic complexity (but that suffers from representational inadequacy), or a very rich language that has expressive capacity closer to that of natural language (but that requires each component to perform complex deconstruction of the representations)? On the other hand, if each module is free to choose its own notation, how does one convert the necessary elements from one representation to another? How does one insure that the overlap in capacities is sufficient and faithful translation to the degree required is even possible?

In this paper, we suggest a middle ground is possible, in which a multi-phase project lifecycle can achieve the advantages of each approach while minimizing their disadvantages. In the early stages of the project, the best strategy is to allow each module designer to choose the representation language best suited for the state of the art in that area, while developing inter-process communication languages to bridge the gap, e.g. (Traum et al., 1996). As understanding of the relationships and requirements are better understood, one can bring the languages closer together. Finally, one needs appropriate tools both within each module and across modules to make modification and creation of new domains easier and possible without additional work by the designers of each module.

Of course, this approach has a cost: one has to develop additional integrating representation resources and notation conversion tools. Central among these is an ontology that provides the standardized terminology and inter-term relationship constraints, plus code to convert this terminology to the component notation. We describe the ontology, representation, and different uses of data in this paper and provide our experience with the efforts and tradeoffs involved.

We illustrate these points through our experiences with the Virtual Human Project at the University of Southern California (USC), which has built virtual agents for the Mission Rehearsal Exercise (MRE) (Rickel at al., 2001) and Stability And Support Operations – Simulation and Training (SASO-ST) (Swartout et al., 2006).

2. The Virtual Human Project

2.1 Project Overview

The Virtual Humans Project, at USC's Institute for Creative Technologies (ICT) and Information Sciences Institute (ISI), has the main goal of designing autonomous agents that support face-to-face interaction with people in many roles and in a variety of tasks. The agents must be embedded in the virtual world and perceive events in that world, as well as the actions of human participants. They must represent aspects of the dynamic situation in sufficient depth to plan contingencies, develop beliefs and intentions, and form appropriate emotional reactions. They must communicate with each other and with human participants using multi-modal natural language communication.

Our latest scenario, an extension of SASO-ST, includes two virtual humans: a Spanish doctor and an Iraqi village elder. Set in a small Iraqi town plagued by violence, the

human trainee takes on the role of an US Army captain with orders to move the doctor's clinic to a safer location (Figure 1).
In the course of the interaction, the human trainee must negotiate with the virtual characters, establishing trust and satisfying the objections of the doctor and elder to moving the clinic. The virtual humans evaluate the utterances made by the trainee and each other, update their models of the conversational states and models of each other, and plan how to react and what to do next.

Figure 1: SASO-EN Scenario

2.2 Virtual Human Architecture

The Virtual Human Architecture includes a large set of modules, which reason about knowledge in different ways. Figure 2 shows a conceptual organization and information flow for these modules. The task reasoner, emotion module and Dialogue Manager are developed in SOAR and TCL (Newell, 1990). Other modules are developed in Java and C++. For a more in-depth discussion of the general architecture and some of its application, see (Kenny et al., 2007). Below we describe some of the modules and the ways they use knowledge:

- An Automated Speech Recognizer (ASR), converting vocalizations into words (Pellom, 2001). ASR needs the words (spelling and pronunciations) that appear in the domain, as well as their frequencies (Unigram, bigrams, and trigrams).
- A Natural Language Understanding module (NLU), converting unconstrained natural language expressions to internal representations (Bhagat et al, 2005). Our statistical approach to NLU requires a training corpus of paired utterance texts and semantic representations from the domain (that we call a *Framebank*).
- A task reasoner that can plan how to achieve goals and reason about alternatives and utilities of various actions (Traum et al, 2003b). The task reasoner focuses on states (that can have utilities for different agents) and tasks (that can have states as preconditions and effects), as well as plans that combine the two in causal networks.
- An emotion module that appraises the state of the world in relation to beliefs and goals, resulting in emotion and specific coping strategies (Gratch and Marsella, 2004). The emotion model makes direct use of the task model representations, as well as factors such as temporal status, likelihood, controllability, and changeability.
- A Dialogue Manager (DM), which relates the NLU output to the context of previous conversation and other internal state, including the task and emotion models, updates the internal state, and plans new communications (Traum et al 2003b, Traum 2003). The dialogue manager uses both the task model representations as well as more structured abstractions of actions related to natural language.
- A Natural Language Generation module (NLG), which converts internal communication goals to output text (Traum et al 2003a, DeVault et al, to appear). This uses detailed aspects of the dialogue model as well as either lexical and grammar rules or a framebank (or both).
- A text-to-speech synthesizer. We have used several synthesizers, including Festival and rVoice. These require domain words as input.
- A non-verbal behavior generator, which decides what body movements should be performed in order to convey the appropriate meaning of NLG output, emotions, perception and conversational regulation. (Lee & Marsella, 2006). This requires representations from the dialogue, task, and emotion models, as well as the NLG output and the body's current position, orientation, and behaviors. The generator outputs a Behavioral Markup Language (BML) (Kopp et al, 2006).
- A behavior blending system, SmartBody, which takes directives for motions and allocates resources (Thiebaux et al, 2008). This requires BML input and knowledge of the character's attributes in the virtual environment.
- The virtual environment, displaying the characters and their surroundings. We currently use the Unreal 2.5 Engine as our renderer. It must track the visual aspects of objects in the world and motion.
- The real environment, consisting of the trainee.

These modules communicate using a message passing protocol that any module can subscribe to. Some modules (e.g. ASR, NLU, NLG) are stateless, and transform their input into an appropriate output. Other modules track and update context and may send commands and requests to other modules.

2.3 Representation Languages and Knowledge Resources

As described above, there are many different types of knowledge resources in the SASO system, which have to be consistent in some ways, but also have different requirements for the different modules.

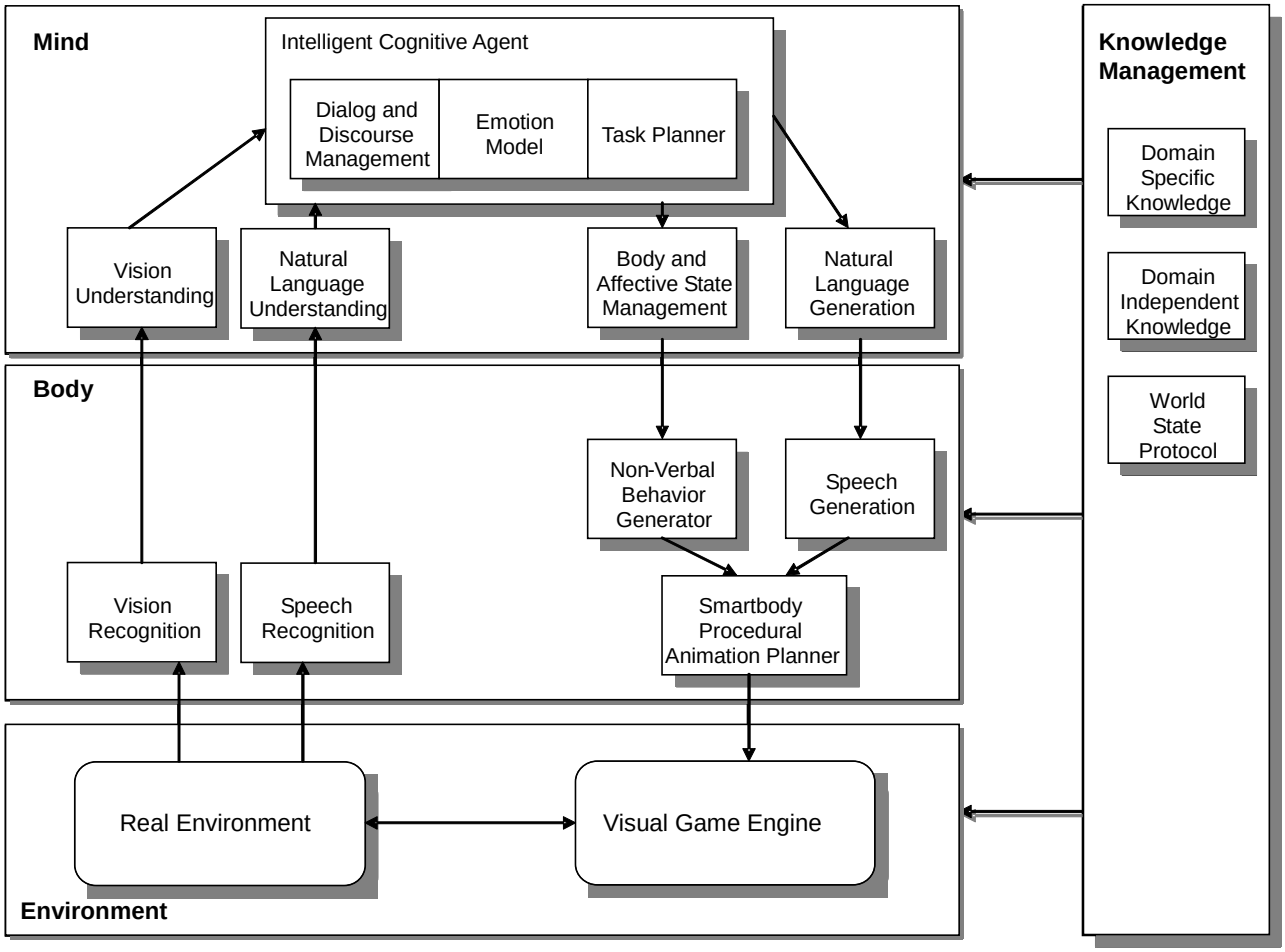

Figure 2: Virtual Human Architecture

Authoring these resources and maintaining them as the domain is changed and expanded can be a significant undertaking. Part of the problem is that there are at least three sources of content authoring:

1. General information on human cognition and interaction, from psychological and AI theories.
2. Story-based information, devised by an author.
3. Language-based information, trying to make sense of the things people say in these domains.

These sources can cause conflict, e.g., when the way people talk about a domain does not match up with the way the domain was formalized from the story. For example, from the point of view of the task model, only fully specified 'move' actions can take place, with a source and destination specified. On the other hand, it is easy to say in English, "move the clinic", without specifying these. It is thus a challenge to come up with a meaningful representation for that phrase. E.g., should one represent literally what was conveyed even if it doesn't make sense to the task reasoner? Should one augment the task reasoner to handle such abstractions? Should one "misrepresent" the utterance as the closest representation that is in the task model? This same suite of choices is presented many times in building the necessary domain modules.

Moreover, there are different means for providing knowledge to different components. Previously, they had to be constructed independently, which was a source of error and effort to maintain consistency. These included:

- SOAR productions that directly create objects and links as part of SOAR's working memory
- TCL macros, that take in arbitrary argument structures and create a set of SOAR productions
- NL *frames*, containing an action or state with added linguistic information for both the NLU and NLG

3. Ontology

3.1 General

In order to address the problem of terminology consistency, we developed a single terminology repository, the Ontology. We gathered from all modules' representations the terms they employ and merged them into one standardized list that forms the terms of the ontology. In doing so we faced a complex task, not yet completed, namely decomposing conceptually composite terms used by some modules into their component terms and relations, as required by others. In addition, to ensure that in the future terms are related only in ways that all the various modules can actually support, we defined inter-term relationships, such as an inheritance hierarchy and constraints on frames' slot values.

In themselves, these are not innovative ideas. But the range of tasks the ontology must support is rather larger than most NLP-related projects have to deal with.

Covering aspects as diverse as speech recognition and synthesis, natural language understanding, generation and dialogue management, body movement, task planning, and emotion, we were faced with the need to handle a wide spectrum of representational needs. Details of the various aspects of the ontology are described in the next sections.

This work is not fully complete. At present, the principal modules now directly using the ontology are the task model and the NLU (while some other modules build on these representations). We have been experimenting with the ontology itself in order to find the most supportive and flexible environment and notation.

At present, we have two iterations of our ontology and use Stanford's Protégé (Knublauch, 2004) to manage both. Protégé supports two types of representation languages: a frame-based representation (Protégé Frames) and the Web Ontology Language (OWL) (McGuiness & van Harmelen 2004, Bechhofer et al 2004). For our first iteration, we chose to use Protégé Frames, as this lay conceptually close to the existing data sources and did not have the overhead that OWL brings in. Our philosophy was to create an ontology that did not require many modifications to the existing system. This version gave us the benefit of integrated data sources and created the necessary experience needed to leverage all the benefits an ontology can give.

The goal for the second version of our ontology was the re-use of knowledge and the introduction of a more principled ontology design. Instituting a principled design of the ontology meant making changes to existing representations of the system.

We switched the representation language to OWL to automatically classify concepts and instances, and most crucially because it allowed us to institute a hierarchical structure of domain independent and domain dependent concepts. This resulted in a three-level organization, which will be discussed in the next section.

The OWL language allows a more flexible distribution of assertions. Drawing on its semantic web roots, OWL allows the addition of assertions to objects that are imported, as well as those created in a particular level. This is in contrast to Frames, where imported instances cannot have any information changed. The greater flexibility of OWL makes sharing of information easier, since one can inherit partially specified instances (the shared part) and then complete the customization at a more detailed level.

One further consideration was the wider availability of tools and ontology resources for OWL.

3.2 Structure

Using the import mechanism of the OWL language, we created a three-level organization of the knowledge. We have a common, general-purpose *world* ontology. Most classes, like 'Person' and 'Action', are defined here. Inheriting the world ontology and adding more specialized knowledge shared by multiple scenarios—locations, props, characters and basic task structures—is the *scenario family* level. This allows us easily to share certain information over a set of closely-related scenarios. Finally, at the lowest level, we have a *scenario* ontology that stores scenario-specific information.

The world ontology is structured to provide a widely applicable set of concepts that can be specialized and instantiated at the scenario family and scenario specific levels. The highest level of the ontology defines, for example, entities such as military officers; specific entities like our captain are then defined at the scenario family level. The world level is expected to be useful across many different scenario families.

In addition to the entities, instantiated actions and states exist at the scenario family level in a basic form. These instances are used by both the task model and the NLU frames, which add module-specific information to them, such as relations and linguistic information. This ensures consistency between modules and enables re-use of knowledge.

Consider a basic 'move' action, where our captain is moving the clinic from the market to the downtown area. We can define this as a set of slot / value pairs:

```
event move
agent captain-kirk
theme clinic
source market
destination downtown
```

Similarly, we can define the state 'the clinic is downtown' as follows:

```
object clinic
attribute location
value downtown
```

Currently, the world ontology contains 192 classes, 125 properties and 199 individuals. The scenario family level has an additional 6 classes and 548 individuals. The multi-party scenario level adds 5 classes and 88 individuals, along with additional relations between individuals inherited from the family level.

3.3 Task Model

The purpose of the Task Model is to represent the tasks (action plans), at both generic and specific (instantiated) levels, of the agents. This naturally encompasses the model each individual agent has of the world. The agent model contains entities, a representation of the world state with *object:attribute:value* triples, and task elements using a STRIPS-like representation (Fikes & Nilson, 1971). The task elements use states as their preconditions and effects.

In the OWL ontology, we introduced the notion of *generic actions* that include descriptions of their precondition and effects templates. This allows us to define basic, domain independent preconditions and effects only once and let the system instantiate that for each scenario.

For example, the generic 'move' action defines effects such as adding "the *theme* is at the *destination*" that are later instantiated for our scenario. This type of reasoning goes beyond the standard OWL inference capabilities and required the construction of our own template interpretation code.

The different levels of ontology structure, combined with the flexibility to choose where to assert knowledge can be

used to add some additional scenario effects of actions. As noted above, generic preconditions and effects of actions are defined at the world ontology level. Some additional effects of more specialized movement can be attached at the scenario family or scenario specific level. One example of that is the way that particular *instruments* of a movement action can affect the (perception of) the Spanish doctor's neutrality. If, for example, U.S. troops move the clinic, that has a negative effect on the doctor's neutrality—which he doesn't want to occur. If local workers perform the move, then his neutrality is maintained. This information is added at the scenario family level, since it depends both on specifics of the scenario family and on the existence of entities that are defined at that level (the local workers and U.S. troops).

Although it is possible to make these specific assertions manually, we have also been exploring ways to make these effects flow from a causal model.

One innovative use of the OWL language is some initial work on assigning additional properties to actions. For example, we have a general definition for "actions that reduce neutrality". This is defined as an action taken by a partisan party that benefits a neutral party. This allows us to automatically infer the effects on neutrality of certain actions in our domain, specifically, having the U.S. troops move the clinic. We plan to use a library of such meta-descriptions to include additional effects without the need to assert them specifically.

In addition to preconditions and effects, authors can also define *concerns* that agents might have for certain states to be true or false. These concerns allow the emotion module to calculate how an agent feels about the current state and / or possible future states of the world.

Below are the examples we used in section 3.2, augmented with the knowledge that is specific for the task model. For the event, these are the preconditions and effects:

```
event move
agent captain-kirk
theme clinic
source market
destination downtown
pre: clinic-location-market
del: clinic-location-market
add: clinic-location-downtown
```

For the state, these are the belief and concern:

```
object clinic
attribute location
value downtown
belief false
concern {doctor-perez 10}
```

At the moment, our ontology contains 14 types of actions and around 40 instantiated actions. These can use a total of 15 case roles (theme, source, etc.). States can be created using 20 objects, 15 attributes and 25 values. There are a total of around 40 non-generated states.

3.4 Natural Language

The NL modules communicate with the Soar agent by exchanging semantic information in a *semantic frame*, which stores information that is linked to the underlying actions, world state and entities. These frames are linked with natural language utterances to form an utterance / frame pair. These pairs are grouped per domain in separate framebanks, one for each character. The framebank for the trainee is used by the NLU; the framebanks for the virtual humans are used by the NLG. At the moment, only the NLU is fully integrated with the ontology. The NLG uses the concepts that are defined in the ontology, but NLG frames are produced by the dialogue manager. The latter manipulates the task model using an internal representation of concepts, rather than the ontology directly, so there is still a possibility of a conceptual mismatch if the representations in the ontology and dialogue manager are out of synch. In future work the dialogue manager will get all of its representations from the ontology.

Before the introduction of the ontology, all NLU semantic frames were created by hand. This allowed our linguists to create semantically rich frames. The drawback is that this richness is hard to support in the Dialogue Manager and task model. Typos and other mistakes can lead to other performance problems. It can also lead to lower performance in the NLU if the frames are not internally consistent.

In order to recreate the NLU framebank in the ontology, we needed three different types of information: the natural language utterance, formal information about the content of the utterance and linguistic information.

For example, an urgent request from the captain such as "I must move the clinic to the downtown area", can be represented in a semantic frame as follows:

```
mood declarative
sem.speechact.type statement
sem.modality.deontic must
sem.polarity positive
sem.type event
sem.event move
sem.agent captain-kirk
sem.theme clinic
sem.source market
sem.destination downtown
```

Note that the core semantics are derived from the basic 'move' action presented in section 3.2.

Our initial prototype for the multiparty domain has about 60 semantic NLU frames that are linked to around 250 utterances. We have yet to start formal subject testing, which will produce a substantial increase in the number of utterances (our previous two-party domain has about 1000 utterances in the framebank).

Ideally, all of the words in an utterance would be part of a lexicon in the ontology, tying the natural language directly to the concepts we support. However, the current implementation of the NLU is geared towards whole utterances rather than individual words or phrases, allowing us some short cuts in interpretation. All actions have a word family associated with them, which potentially allows for a variety of tenses. In addition, each object ID is treated as a lexical item. Current plans include more advanced NLU and NLG, which will make use of more lexical information.

3.5 Exporting Representations

Naturally, simply incorporating an ontology into a collection of disparate modules did not magically solve the standardization problem. Since it was infeasible either to rebuild the various modules from scratch or to recode them to employ the standardized representation formalism, we created a set of 'exporter functions' that converts each representation statement—from a single attribute-value pair all the way up to a scenario—into the internal notation of most modules, and a set of 'import functions' to perform the opposite conversion.

We have implemented these importer and exporter functions as Protégé editor tabs. The use of meta-concepts allows us to make changes to the ontology without the need of rewriting our plugins.

Generating all of the output code from the central ontology gives us the assurance that all of the system modules are using consistent semantics for our application. Hence the importer and exporter functions also provide some quality control.

3.6 Reasoning

One of the benefits of using OWL is the availability of classifiers, which can automatically maintain hierarchy information based on the logical definitions of classes. This allows one to have a multi-hierarchy of more abstract and more specific classes maintained automatically. This is helpful in the organization of the action hierarchy, since one can have general move actions, and then specialize them, say, to move actions that have "the clinic" as the theme. Classifiers can maintain the class/sub-class relationships as well as properly assigning instances to their proper place in the hierarchy.

OWL defines several levels of expressive power, ranging from Lite, through Description Logic (DL) and Full. Certain reasoners, like concept classifiers only operate on the DL level of the language. Parts of the ontology that are expressed using the OWL Full language cannot be automatically classified, because the standard OWL reasoners require that one restrict the expressive power of the OWL language to the OWL-DL subset. But certain of our constructs are most naturally modeled using the OWL-Full language. In particular, OWL-DL does not allow one to specify properties as the values of other object properties. This causes problems in the definition of simple queries, since simple query is a semantic frame with one of the case roles unspecified and designated as the query. But representing that places the language into OWL-Full, and prevents the classification of queries.

4. Related Work

There is an interesting disconnect between ontology construction at the large scale and actual usage in complex computational systems. Large-scale term taxonomies such as WordNet (Fellbaum et al., 1998) simply do not provide the amount of information that our modules need. Even slightly smaller and more semantically oriented ontologies, such as Mikrokosmos (Onyshkevych and Nirenberg, 1995) and FrameNet (Ruppenhofer et al., 2006) base their semantics purely on linguistic principles. While very useful for generic NLP, and for us for NLU and NLG, they do not provide enough information to support the more detailed reasoning required, in our case, by task and action planning, body movement, etc. Our ontology, in contrast, has to contain more information about speech acts and intentional connotations of words, and hence is more focused on the particular domain, thereby being anchored to a semantic representation that the agents understand. Our ontology is also linked to a concrete model of objects in the simulated world, rather than being more generally connected to real-world items.

This fact has led us to develop our ontology through a process of organic growth, starting with more lexically-oriented term taxonomies such as WordNet then adding information as required by the various models. Thus, in many cases, the task and agent models drive ontology development, but their additions are not considered complete until the NL-related information required both to parse and to generate with those terms is also added. The result is a set of representation terms and interrelations that include a richly diverse set of information of quite different kinds, supporting reasoning in various spheres of human activity.

This model of organic growth has the disadvantage that it is never complete; we may at any time encounter a term that the system does not yet know. But it has the advantage that development of our system is more tractable and the precise semantics of the agent model is captured. It has the drawback that expansion of the domain also requires us to construct the semantic models rather than use existing sources. But even with existing sources, we would have to ground the semantics in our agent's world model, which is a considerable amount of work.

Our use of generic action templates is similar to the Parameterized Action Representation (Badler et al., 1998; Bindiganavale et al., 2000). The Parameterized Action Representation is used as a means of communication between users and the agents. Our underlying representation is tied to a different agent control system, and the contents of the templates are filled in by instantiation from the ontology rather than user input.

The Smartkom Project (Wahlster, 2006) is inspirational in its use of an ontology to solve a number of natural language processing issues for a system including a virtual character and several simple command tasks. Yet we do not know of any other multi-component model of human activity comparable to the Virtual Human Project with which to compare our experience.

5. Conclusions and Future Work

The current ontology gives us several benefits. First, it assures that the knowledge used by the task model and NLU are synchronized, because they share the basic representation. Second, it forces the author to strictly follow the rules of what constitutes a valid semantic frame, because we can constrain the model to follow our specification. Third, it allows users to reuse knowledge, by combining existing individuals. Fourth, it provides a safer mechanism for changing data, because knowledge is referenced, rather than copied. And finally, it provides a

common user interface for all author related tasks.

There are also drawbacks, though, which we hope to address in subsequent versions. Using OWL and Protégé introduces an extra learning curve for new developers in our project, which is especially an issue for non-computer scientists. In addition, although it allows for easier and safer change in some ways by changing certain assertions, we see that changes that include naming conventions require a lot more effort than a simple replace-all would in a text file. Lastly, even though Protégé offers a rich graphical user interface, this interface is not geared towards the authoring tasks our system requires. Especially for new users, it can be hard to find existing knowledge. The creation of new authoring tools on top of Protégé is something we have high on our priority list.

One attractive feature of using an ontology as a central repository is the potential ease of extension of the system. Whenever needed, we could draw additional terms and relations, as well as additional Upper Model terms, from the Omega ontology (Philpot et al., 2005), for example. By incorporating them into the appropriate points of the ontology, and making sure that ancillary data is provided (such as lexical items to support NLU and NLG or speech training to support ASR), the other modules of the system can employ the new terms almost directly.

Our decision to use OWL as the representation language has provided us with both benefits as well as limitations. The main benefits are the provision of classification services and the ability to attach assertions to inherited individuals. It has some limitations which have also proven to be problematic. The desire to include properties as values pushed us into OWL-Full and restricted the usefulness of the classification reasoners. There are some workarounds for this, but they make the representation more cumbersome. A second major feature that proves troublesome is the monotonic nature of the inheritance of imported information. The ability to import other ontologies is crucial to supporting reuse of information. But all such inheritance is monotonic—in other words, new information can be added, but none of the existing information can be retracted or overridden. This makes it imperative that one make sure that all assertions are made at the proper level in the inheritance structure. If assertions are added at too low a level, then no sharing takes place. If made too high up, then the information cannot be removed, which can limit the ability to share knowledge and settings. It is difficult, even for experienced users, to decide where new knowledge should be created.

The monotonic requirement also makes it impossible to have actual default values, since any such value could not be removed. We are able to avoid that problem through the use of meta-annotations that associate default values not with the individuals, but instead attach them to the properties themselves. These meta-annotations are then interpreted by our own reasoning code and the exporters to give us the effect of default values.

Finally, there is currently no standard query language for OWL, although progress is being made there as well (SQWRL 2008).

At present, not all Virtual Human modules have been integrated with the central ontology. We are busy integrating some, such as NLG, and plan to integrate more, most notably SmartBody. This involves extending the knowledge base with concepts from the virtual environment and the development of a rich lexicon. The ultimate goal is to tie together all the information that different modules use about a single concept.

Including an ontology and suitable knowledge entry and representation import/expert functions into an existing system can be seen as a step on our Virtual Human's maturation process from research pilot system to prototype to, eventually, a production-level system. In this task we face the challenge of determining the optimal tradeoff point between system simplification and complexity. In USC's Virtual Human Project, the ontology and associated framework provide a rich context for investigating this challenge.

6. Acknowledgements

We would like to thank the entire Virtual Humans Project group at USC, ICT and ISI, which allow us to work on these exciting issues.

This work was sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM), and the content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

7. References

Badler, N., R. Bindiganavale, J. Bourne, M. Palmer, J. Shi, and W. Schuler. 1998. A Parameterized Action Representation for Virtual Human Agents. *Proceedings of the Workshop on Embodied Conversational Characters*. Lake Tahoe, CA.

Bechhofer S., F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider and L. A. Stein. 2004. *OWL Web Ontology Language: Reference*, W3C Recommendation. http://www.w3.org/TR/owl-ref/

Bhagat, E., A. Leuski, and E.H. Hovy. 2005. Statistical shallow semantic parsing despite little training data. *Proceedings of the 9th International Workshop on Parsing Technologies (ACL/SIGPARSE'05).* Vancouver, B.C.

Bindiganavale, R., W. Schuler, J. Allbeck, N. Badler, A. Joshi, and M. Palmer. 2000. Dynamically Altering Agent Behaviors using Natural Language Instructions. *Proceedings of Autonomous Agents Conference 2000*, pp. 293–300.

Chercheur, J.L. 1994. *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufman Publishers.

Castor, A. and L.E. Pollux. 1992. The use of user modelling to guide inference and learning. *Applied Intelligence*, 2(1), pp. 37–53.

DeVault, D., D. Traum, and R. Artstien, "Making Grammar-Based Generation Easier to Deploy in Dialogue Systems", to appear.

Fellbaum, C. (ed.) 1998. *WordNet: An Electronic LexicalDatabase*. Cambridge, MA: MIT Press.

Fikes, R., and N. Nilsson. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189-208.

Grandchercheur, L.B. 1983. Vers une modélisation cognitive de l'être et du néant. In S.G. Paris, G.M. Olson, and H.W. Stevenson (eds.), *Fondement des Sciences Cognitives*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 6–38.

Gratch, J., J. Rickel, E. André, N. Badler, J. Cassell, and E. Petajan. 2002. Creating Interactive Virtual Humans: Some Assembly Required. *IEEE Intelligent Systems*, July/August, PP. 54–63.

Gratch, J. and S. Marsella. 2004. A domain independent framework for modeling emotion. *Journal of Cognitive Systems Research* 5(4): 269–306.

Kenny, P., A. Hartholt, J. Gratch, W.R. Swartout, D. Traum, S. Mareslla, and D. Piepol. 2007. Building Interactive Virtual Humans for Training Environments. *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC).*

Knublauch, H., R.W. Fergerson, N.F. Noy, and M.A. Musen. 2004. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. *Proceedings of the Third International Semantic Web Conference.* Hiroshima, Japan.

Kopp, S., Krenn, B., Marsella, S., Marshall, A., Pelachaud, C., Pirker, H., Thorisson, K., Vilhjalmsson, H. "Towards a Common Framework for Multimodal Generation: The Behavior Markup Language". 6th International Conference on Intelligent Virtual Agents (Marina del Rey, CA, August 21-23 2006).

Lee, J. and S. Marsella. 2006. Nonverbal Behavior Generator for Embodied Conversational Agents. *Proceedings of the 6th International Conference on Intelligent Virtual Agents*, pp 243–255, Marina del Rey, CA.

McGuinness, D., F. van Harmelen, eds. 2004. *OWL Web Ontology Language Overview*, W3C Recommendation, http://www.w3.org/TR/owl-features/

Martin, L.E. 1990. Knowledge Extraction. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 252–262.

Newell, A. (1990) *Unified Theories of Cognition.* Cambridge, MA: Harvard.

Onyshkevych, B. and S. Nirenburg. 1995. A Lexicon for Knowledge-Based MT. *Machine Translation* 10(1–2): 5–57.

Bryan Pellom, "SONIC: The University of Colorado Continuous Speech Recognizer", University of Colorado, tech report #TR-CSLR-2001-01, Boulder, Colorado, March, 2001

Philpot, A., E.H. Hovy, and P. Pantel. 2005. The Omega Ontology. *Proceedings of the ONTOLEX Workshop at the International Conference on Natural Language Processing (IJCNLP).* Jeju Island, Korea. October 2005.

Rickel, J., J. Gratch, R. Hill, S. Marsella, and W.R. Swartout. 2001. Steve Goes to Bosnia: Towards a New Generation of Virtual Humans for Interactive Experiences. *AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment.* Stanford University, CA.

Ruppenhofer, J., M. Ellsworth, M.R.L. Petruck, C.R. Johnson, and J. Scheffszyk. 2006. FrameNet II: Extended Theory and Practice, version 1.3. Berkeley FrameNet Project, University of California, Berkeley, CA.

SQWRL: Semantic Query-enhanced Web Rule Language. 2008. Website http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL.

Swartout, W.R., J. Gratch, R. Hill, E.H. Hovy, S. Marsella, J. Rickel, and D. Traum. 2006. Toward Virtual Humans. *AI Magazine* 27(1).

Swartout, W.R. 2006. Virtual Humans, *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)* (senior paper). Boston, MA.

Thiebaux, M., A. Marshall, S. Marsella, M. Kallmann. 2008. SmartBody: Behavior Realization for Embodied Conversational Agents. *Proc. 7th International Conference on Autonomous Agents and Multiagent Systems.* (to appear).

Traum, D. , L. Schubert, M. Poesio, N. Martin, M. Light, C. Hwang, P. Heeman, G. Ferguson and J. Allen. 1996. *Knowledge Representation in the TRAINS-93 Conversation System, in International Journal of Expert Systems* 9(1):173-223.

Traum, D. 2003. Semantics and Pragmatics of Questions and Answers for Dialogue Agents in *proceedings of the International Workshop on Computational Semantics*, pp 380-394.

Traum, D. , M. Fleischman, and E. Hovy. 2003. NL Generation for Virtual Humans in a Complex Social Environment in Papers from the *AAAI spring symposium on Natural Language Generation in Spoken and Written Dialogue*, pp. 151-158.

Traum, D., J. Rickel, J. Gratch, and S. Marsella. 2003. Negotiation over Tasks in Hybrid Human-Agent Teams for Simulation-Based Training, in *proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, pp. 441-448.

Wahlsterm W (Ed.), SmartKom: Foundations of Multimodal Dialogue Systems", Springer, 2006.

Zavatta, A. 1992. Un Générateur d'Insultes s'intégrant dans un Système de Dialogue Humain-Machine. *Doctoral Thesis, Informatics*. Université Paris-sud, Centre d'Orsay.