PRONTOE
A Case Study for Developing Ontologies for Operations

Scott Bell[1], Pete Bonasso[1], Mark Boddy[2], David Kortenkamp[1] and Debra Schreckenghost[1]
[1]*TRACLabs Inc., Houston, TX, U.S.A.*
[2]*Adventium Labs, Minneapolis, MN, U.S.A.*

Keywords: Ontology, Operations, Space, Procedures, Planning, Robotics.

Abstract: In this paper, we describe a set of software tools called the PRIDE ONTOlogy Editor (PRONTOE) and a methodology that allows system operators and domain experts to build and maintain ontologies of their systems with no explicit understanding of the underlying ontology representation. We present two case studies: one using NASA flight controllers, and another using the DARPA Robotic Challenge.

1 MOTIVATION

Ontologies provide a structural framework for system knowledge that is useful for many applications. One particular application is to provide the knowledge necessary for software tools that assist operators in monitoring and controlling complex and dynamic systems. Using an ontology to model system information has advantages. The ontology models provide monitoring and control concepts as objects that can be used in multiple control domains. These models define properties that can be used to automatically populate object data fields and derive relations between objects to improve search of system information.

The representational power of ontologies, however, introduces a number of challenges. One such challenge is developing and using ontologies for operations that the operators and other domain experts do not have any experience in developing or maintaining ontologies of their systems. Using ontology experts to build and maintain these ontologies is prohibitively expensive, especially because the knowledge necessary to build the ontologies exists in a variety of documents and in the operator's or domain expert's heads. Another challenge is that the states and configurations of the specific objects in the domain are both voluminous and dynamic, making manual entry and maintenance prohibitive. A final challenge is that the data required, especially state updates, need to be extracted or imported from other disparate systems. In this paper, we describe a set of software tools called the PRIDE ONTOlogy Editor (PRONTOE) and a methodology that allows system operators and domain experts to build and maintain ontologies of their systems with no explicit understanding of the underlying ontology representation.

PRONTOE consists of a graphical editing tool that allows users to define and edit objects and their properties and relationships and to view those properties and relationships in a variety of ways and in the context of their particular domain. PRONTOE supports the integration of different ontology *kernels* that divide complex systems into interacting components. PRONTOE includes reasoners for assisting in object definition and consistency. PRONTOE has software tools for importing (and exporting) data to domain-specific databases. PRONTOE also allows for viewing real-time system data in the context of defined objects and their relationships. PRONTOE is being evaluated in two domains. The first is operation of the International Space Station (ISS) by NASA flight controllers. In this case study, the domain experts are the flight controllers who have engineering degrees and years of experience in operating ISS. The ontology is used by software tools such as task planners, procedure editing and execution systems, and diagnosis systems. The second domain is a humanoid robot being developed by DARPA for disaster relief operations. In this case study, the domain experts are robotics engineers. The ontology is used to develop operator interfaces for using robot capabilities as well as robot scripts for automating common activities.

Bell S., Bonasso P., Boddy M., Kortenkamp D. and Schreckenghost D..
PRONTOE - A Case Study for Developing Ontologies for Operations.
DOI: 10.5220/0004520500170025
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2013), pages 17-25
ISBN: 978-989-8565-81-5
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

2 APPROACH

Our approach consists of several interacting components. At the core of PRONTOE is a graphical user interface (GUI) that allows an operator to visually inspect and edit an OWL ontology. The ontology itself is divided into a set of kernels that correspond to the different operational aspects of the system. The kernel approach makes it easier for an operator to focus on the specifics of their system and not the requirements of the underlying ontology. OWL reasoners provide consistency and fill in required ontology information automatically.

2.1 Graphical User Interface

The PRONTOE Graphical User Interface (GUI) can be used to inspect a system ontology, search for specific system information in that ontology, modify information already in the ontology, and add new information to the ontology. The PRONTOE GUI is an Eclipse based Rich Client Platform (RCP) application. Developing PRONTOE as an RCP application allowed us to use a variety well developed libraries for both Java and Eclipse. For ontology data manipulation and reasoner interaction, we used the OWL-API Java library. To render the ontology graph, the Eclipse Zest Toolkit was used. For rendering system schematics, we used the Batik SVG library. Other libraries include integration with version control, workflows, and user authentication.

The RCP approach also allows us to easily add or remove features for different domains. For example, the space domains have different schematics, bundled ontologies, and editing widgets than the robot domain. By specifying these differences in a product definition, we can simultaneously release PRONTOE for different domains for multiple platforms. We also provide extension points into PRONTOE to help other developers create new domain specific features.

PRONTOE's editing environment is divided into several different editing panes. Figure 1 shows a typical environment. The left window shows the current open ontology along with available ontologies to edit. This particular project has been checked out using the Subversion plugin, so the user can right click to add, commit, or update the ontology from version control within the PRONTOE application.

The central pane is a graph of a selection of the ontology. The graph shows class and subclass relationships. At the individual level, object properties are displayed with the property name labeled on the edge. Nodes can by right clicked on to expand or collapse their children. Double clicking on a node opens it for editing its name, object properties, and data properties.

The lower right hand pane is a schematic of the physical layout of the system represented by the ontology rendered as an SVG. The different colors in the schematic show the locations of the class and individuals shown in the center graph. Clicking on an area in the schematic will show all the classes and individuals associated with that location. Different SVGs may be used for different domains by creating a PRONTOE plugin. Figure 1 shows an SVG of the International Space Station (ISS).

The upper right hand pane is a tree showing the ontology's subclasses and individual. From here, classes and individuals may be added, edited, deleted, and reasoned upon. A search bar on the top provides incremental searching for classes and individual names.

All the editing panes can be rearranged or even detached according to user preference. The layout of the windows can be named and saved by the user using perspectives. These perspectives are available on the upper right of the PRONTOE window. The user can rapidly flip through different perspectives depending on their current work or role. By default, selecting an item in one pane automatically syncs the information in the remaining panes to reflect the selection. For example, clicking on a class in the ontology tree changes color the schematic with the locations of all the individuals under that class, and changes the graph to display the subclasses and individuals. This synching feature can be disabled per editing pane by deselecting the sync button in the upper right hand corner of each editing pane.

2.2 Kernels

For PRONTOE, we divide an overall system ontology into a *base* and several *kernels*. This division serves several purposes. The first and most fundamental one is that the base ontology forms an intermediate model that the users are not allowed to modify. Users are not ontology experts, and should not be left to figure out how to model the domain. The base provides a set of generic classes for things like resources, or locations, along with some deductive machinery for maintaining consistency in the description of the world state in PRONTOE's database.

The second reason for this separation between the base and multiple kernels in the ontology is bureaucratic, but no less important for that. For example, NASA's ISS flight controllers (FCs) have divided but interacting responsibilities. The FC responsible for orbital maintenance needs to coordinate with the FC

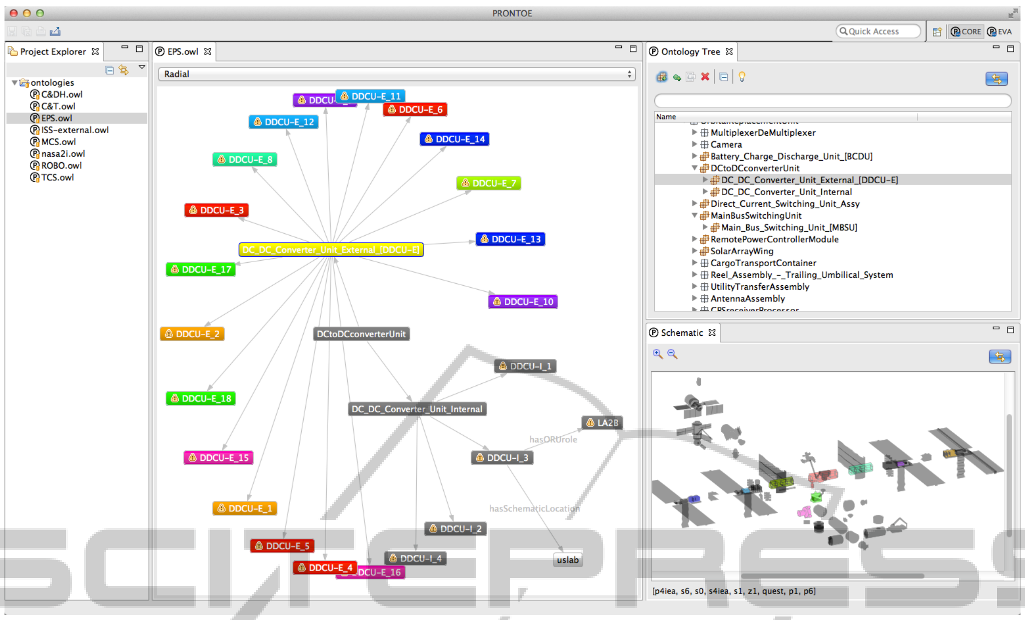


Figure 1: The PRONTOE editing environment.

responsible for power management. Astronaut Extravehicular activities (EVAs) are managed by several FCs at once. This organizational structure requires a supporting structure in PRONTOE. One flight control discipline must have complete control over their own part of the domain, with visibility into parts of the domain under the purview of other disciplines.

As a result, PRONTOE's database has several features that complicate maintenance. First, it is subject to asynchronous access by multiple parties, who may be making interacting or conflicting changes. Second, it is *deductive*: for many changes to the database describing the world state, there are rules that will fire, making additional changes. All of these changes need to be included in the definition of a database *transaction* that allows us to keep PRONTOE's database in a consistent state.

Distributed authority adds more complication, because the separation between disciplines is only partial. For example, there is a defined relationship between two different components of the ISS, such as a pump and the power-channel that energizes it. Disconnecting the pump is a change that must be reflected in the relations for both the pump and the power-channel, which may be in different kernels in the ontology. This requires us to come up with some way to make the change in one kernel and keep the database consistent, which requires an unauthorized change in another kernel.

For now, the solution we have implemented is to use a version-tracking system. All changes, authorized or not, will be made as required to keep the database consistent. Changes made in other kernels will result in the "owners" of those kernels being informed so that they can either approve or reject those changes. Rejection by any flight discipline will then result in the entire set of changes being rolled back.

2.3 Reasoners

For editing-time classification, consistency checking, property inference, and Semantic Web Rule Language (SWRL) reasoning, we use Pellet. The inferred axioms are added back to the currently open ontology allowing the user to save them if they wish. As we are using OWL-API, it is easy to plug in different reasoners (e.g., HermiT) for evaluation. For example, in the case of the ISS if the operator entered the property that a certain computer controls a piece of equipment, then the reasoner will assert a property that the piece of equipment is controlled by that computer. This reduces the burden on the operator of having to specify completely the relationships in the ontology when many of them can be inferred.

3 CASE STUDIES

We have used PRONTOE to develop ontologies for two different complex systems. The first system is the International Space Station (ISS) being operated by NASA and the Johnson Space Center in Houston Texas. The second system is the Atlas humanoid robot being developed by the Defense Advanced Research Projects Agency (DARPA). In this section, we describe these two case studies and how we used PRONTOE to simplify ontology development and maintenance.

3.1 International Space Station

Our principle development activity for PRONTOE is an ontology for the International Space Station (ISS). We have worked for the past two years with NASA flight controllers to develop and design an ontology that partially models the ISS. The focus of the effort is on planning for Extravehicular Activities (EVA), basically space walks, so our ontological concentration is on ISS objects that are located on the outside (or external to) ISS, such as power module and antennas. PRONTOE, as mentioned in section 2.2, comes with a base ontology, a domain base we call ISS-base, and kernel extensions for EVA and for each flight discipline that supports a given EVA, such as electrical power and motion-control systems. The users can then use PRONTOE to extend these kernels, incrementally as new ISS activities arise. To prepare for an upcoming EVA, the flight controllers start with a current configuration of the ontology, and use PRONTOE to develop and save a snapshot of the configuration of equipment, power and control that is anticipated to be true at the time of the activity. In an extension to PRONTOE, we are developing a capability to generate change forms concerning location and configuration changes that resulted from the EVA to be distributed to other ISS parties such as mass properties analysis teams and ISS guidance and navigation teams.

We have developed interfaces that allow PRONTOE to automatically import from two large NASA databases of ISS equipment: the External Configuration Analysis and Tracking Tool (ExCATT) and the Inventory Management System (IMS). The ontologies created by these systems are large, with 4855 axioms containing 283 classes and 897 individuals. We have thus broken the ontology into kernels to ease editing. By connecting to existing databases, we reduce the upfront editing time necessary to build the ontology. We can also export to these databases, so any changes that operators make using PRONTOE can be pushed back into the official databases of record.

The end goal of PRONTOE is to have operators add equipment to the ontology. As an example of editing in PRONTOE, we will walk through a user adding a new type of gas tank assembly for the ISS. The existing gas tank assemblies are show in Figure 2 by selecting the GasTankAssemblies class in the ontology tree. The classes are all colored to mark their different locations on ISS. The user will create a new class by clicking on the “Add Class” button in the ontology tree toolbar. The resultant dialog is show in Figure 3. Inherited object and data properties that can be bound are show in cyan. New properties for the class can be added by clicking the appropriate plus button. In this case, the user wanted to create a new class called OxygenTankAssembly. Clicking OK on the dialog creates the appropriate axioms and marks the ontology as dirty. In Figure 4, the user is in the process of creating and specifying a new individual of OxygenTankAssembly. Object properties are specified in the top table and data properties specified in the bottom table. If object or data property hasn’t been specified, we mark the field yellow.

The first user trials of PRONTOE involved EVA flight controllers, and to a lesser extent the robotics flight controllers (known as ROBOs). The EVA flight controllers generally approved of our current development, but asked if we might build a tighter interface to the 3D graphics engine they use known as DOUG (Dynamic Onboard Ubiquitous Graphics). But they also indicated that knowledge of how EVA serviced equipment was related to the information in the others kernels, e.g., power and computer control, would be useful to them for setting up preconditions on their EVA tasks. Our ROBO flight controller was skeptical that the ROBO team would use PRONTOE for procedures, but she saw a number of potential uses for the tool, such as providing support for operational planning meetings, for collaboration among disciplines, for troubleshooting training, and for use in simulation scripting meetings.

Later in the project, we demonstrated our current version of PRONTOE to core systems flight controllers. They all were impressed with how we had pulled together data from disparate sources into one integrated view and suggested that we have a series of one-on-one knowledge engineering meetings with each of them to see if our kernels had enough key concepts modeled for the users to extend them without our help. We began those sessions with the vehicle motion control flight controller who spent an afternoon with us investigating the ontology and pointing out what was missing if he were to use it in his day-to-day operations. The resulting additions included being able to model internal items that connect to external items, allowing multiple remote power controller modules (RPCMs) in our power channel models, and adding computer control channels to augment our relations, controllerFor and controlledBy.

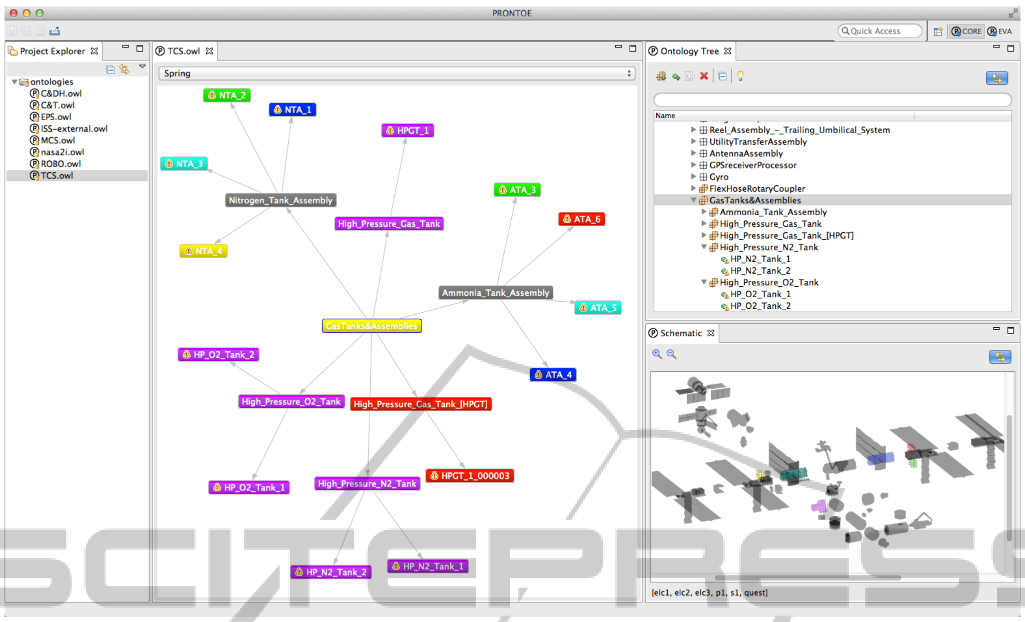
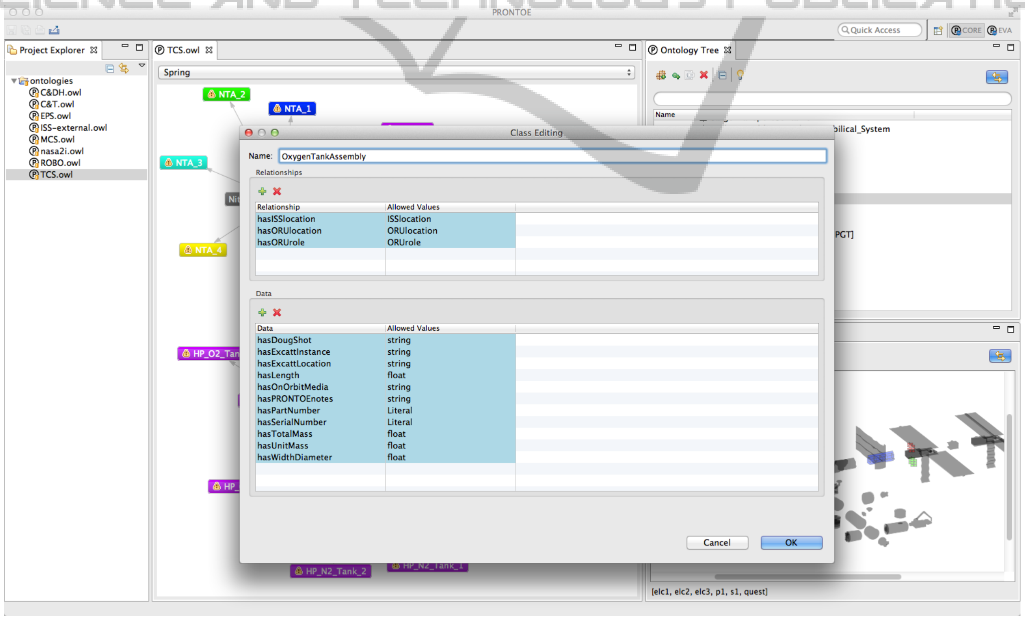


Figure 2: The gas tank assemblies on ISS.



Name: OxygenTankAssembly

Relationships

Relationship	Allowed Values
hasISSlocation	ISSlocation
hasORUlocation	ORUlocation
hasORUrole	ORUrole

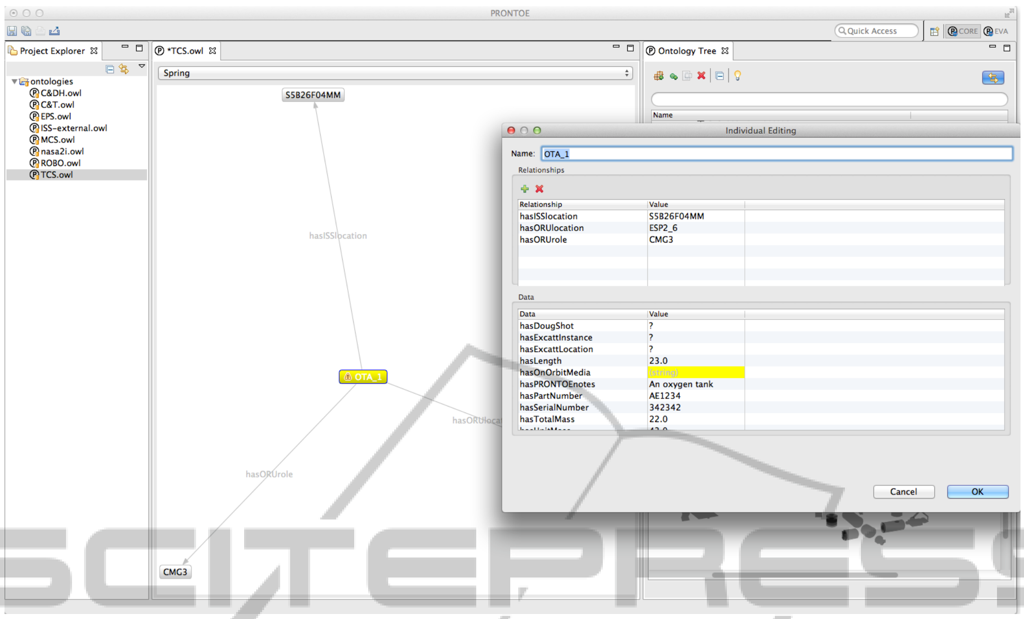
Data

Data	Allowed Values
hasDougShot	string
hasExcattInstance	string
hasExcattLocation	string
hasLength	float
hasOnOrbitMedia	string
hasPRONTOEnotes	string
hasPartNumber	Literal
hasSerialNumber	Literal
hasTotalMass	float
hasUnitMass	float
hasWidthDiameter	float

Figure 3: Adding a new class called OxygenTankAssembly to the ontology.

3.2 DARPA Atlas

In addition to using PRONTOE for space systems, we have also been developing ontologies in the robot domain. TRACLabs is developing automation and control software for the simulated Atlas robot used for the DARPA Virtual Robotic Challenge (VRC). We have defined an ontology of robot affordances of the Atlas robot to improve user understanding of the capabilities of a robot. These affordances define the perceived and actual capabilities of the robot based on Norman's definition of affordances (Norman, 1998). This robot-centric ontology can be used to ground human-robot interaction about what the robot can know about itself and its environment based on what it can sense, and what the robot can do based on encoded behaviors. The ontology of robot affordances models the capabilities of a robot as Behaviors to change the robot's Stance. A Stance is a meaningful configuration of a robot's components and/or sys-



Name: OTA_1

Relationships

Relationship	Value
hasISSlocation	S5B26F04MM
hasORUlocation	ESP2_6
hasORUrole	CMG3

Data

Data	Value
hasDougShot	?
hasExcattInstance	?
hasExcattLocation	?
hasLength	23.0
hasOnOrbitMedia	(string)
hasPRONTOEnotes	An oxygen tank
hasPartNumber	AE1234
hasSerialNumber	342342
hasTotalMass	22.0

Figure 4: Adding a new instance of OxygenTankAssembly.

tems. For example, sit is a Behavior of a bipedal robot that produces the Stance of sitting. A Behavior is accomplished by executing command sequences modeled as CommandLists. Each Command in the CommandList is associated with a BodyPart of the robot. The execution of a Command changes the State of the robot defined for components associated with that BodyPart. This ordered sequence of transient States is captured in a StateList. For example, a sequence of Move Commands in a Motion Behavior produces a corresponding sequence of Pose States that change the JointStates associated with the Command. The end state resulting from the execution of a CommandList is a Posture Stance. Figure 5, summarizes the key concepts in the ontology of robot affordances and the properties relating these concepts.

As shown in Figure 6, we use the PRONTOE ontology editor to visualize and inspect this ontology of robot affordances for Atlas. We automatically generate an ontology from the Unified Robot Description Format (URDF), which is an XML standard for representing a robot model. A schematic of the Atlas robot illustrates the robot components in the ontology, such as Joints and BodyParts. The user interacts with this schematic to identify concepts corresponding a particular component. For example, the user can search the ontology for all Behavior instances defined for the Atlas robot. We are currently investigating the use of the ontology of robot affordances to build operator interfaces to the robot.

4 APPLICATIONS

The ontology developed in PRONTOE is not an end in itself. It is designed to support a set of core autonomy capabilities. For example, we are able to translate the OWL ontology into a planning language called Planning Domain Description Language (PDDL) (Fox and Long, 2003). We then have a task planning tool, called AP (Elsaesser and Sanborn, 1990), that can read in PDDL and use those models to schedule tasks that need to be performed. For example, let's take the overall task of replacing one of the ISS power modules. This requires subtasks of shutting down certain systems that are on that power module. The ontology describes the connectivity between the power modules and various subsystems. However, that is not enough information for the planner. For example, the internal thermal control system (ITCS) of the US Lab on the ISS. The ITCS is controlled by ISS computers S01 for primary control and S11 for backup control for a power pump of Loop A (the low temperature loop) of an external thermal control system (ETCS). Basically, the pump moves heat from inside of the ISS to outside of the ISS. Similarly, Loop B performs the same function for the medium temperature loop. When the ISS power module is shut down, backup computers must be brought up to run it. Also, Loop A needs to be brought down completely. With the loss of Loop A, the low temperature loop in the lab won't function properly, so the lab ITCS needs to be reconfigured to single loop mode. In this mode, the

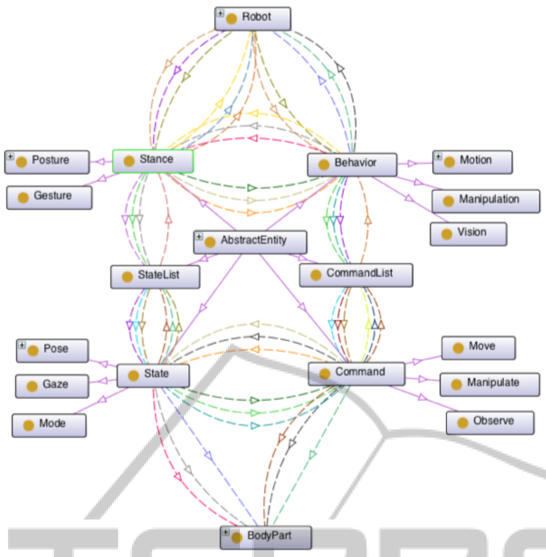


Figure 5: Key concepts and properties in the ontology of robot affordances.

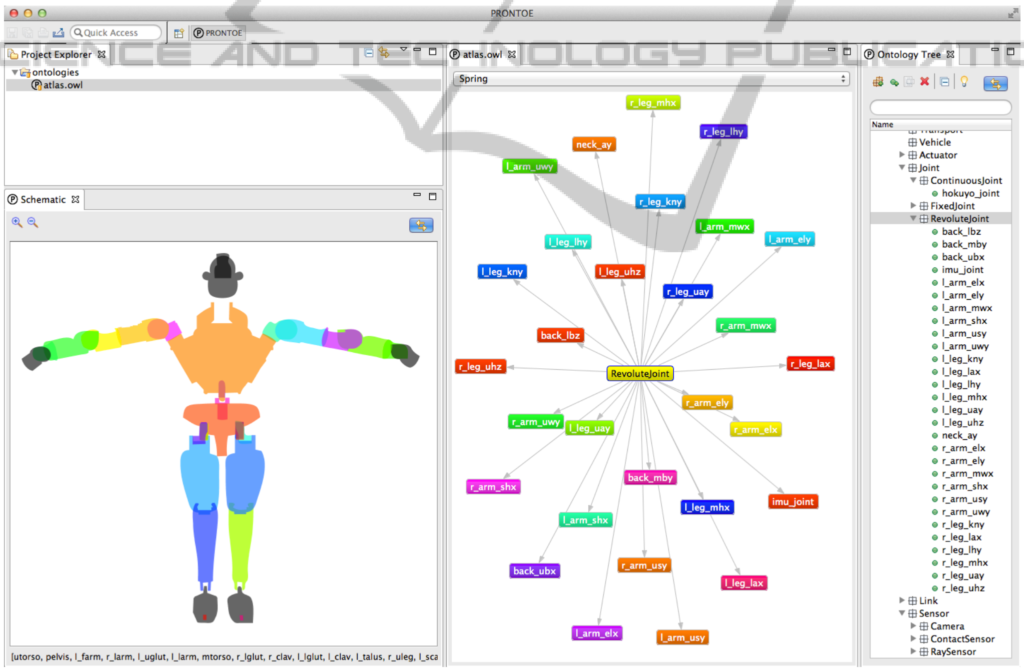


Figure 6: The Atlas ontology created from a URDF.

three-way valves are set so that all the water passes through the medium temperature heat exchanger that is serviced by Loop B.

In our ontology we model relationships of the various power and computer units to the thermal control objects. The ontology is translated into PDDL and read into the planner as the initial situation and a plan is generated. When AP generates a plan to remove and replace power module, if the lab ITCS is in single loop mode, no lab ITCS action must be taken. But when the ITCS is in dual loop, the necessary additional computer and power actions are added to the plan to have the water pass through the medium temperature heat exchanger. Without an up-to-date ontology containing all system connectivity and state, the automated planner would not be able to generate appropriate plans and manual intervention would be necessary.

We are also using PRONTOE for editing standard operating procedures for ISS. We have developed a procedure representation language (PRL) (Kortenkamp et al., 2008), that captures the form of traditional procedures, but allowing for automatic translation into code that can be executed by autonomous executives. In order to author PRL, we have developed the Procedure Integrated Development Environment (PRIDE) (Izygon et al., 2008). Procedures authored in PRIDE and output in PRL require knowledge about the available commands and state of the ISS subsystems (Bell and Kortenkamp, 2011). PRIDE can read in OWL files and create drag-and-drop interfaces for system commands and to verify system state. PRONTOE makes it easy to edit the OWL files to keep the data consistent with current ISS operational needs. For example, if a new piece of equipment is added to ISS, PRONTOE can assist a flight controller in adding just enough information to the ontology to quickly build procedures for the device. If the new device is faulty, a simple change to the ontology will render all procedures that dependent on it invalid. In this way, PRONTOE can act as a verification of procedures with the systems they interact with.

5 RELATED WORK

Ontological engineering (OE) has been a regular activity in the AI community for many years. In 1999 it was considered in its infancy for lack of use of widely accepted methodologies (Lopez, 1999), but as late as 2007, the majority of OE researchers still did not use any methodology (Cardoso, 2007). Yet, most OE research accepts as fundamental the need for an efficient, consistent paradigm for knowledge engineering ontologies (Soares and Fonseca, 2009).

Work on meta-theories, e.g., (Herzig and Varzincak, 2007), may be considered related in that it attempts to view an ontology from a perspective of common concepts and elements. Myers' work on planning domain meta-theories (Myers, 2000) falls in this vein, where she discusses such things as characterizing air/land/water as "transport media", and that movement concepts involve a source and a destination. Our work on a base ontology as distinct from kernel ontologies is similar and our interactive approach will use abstraction levels to make the authoring of models easier for the user.

6 FUTURE WORK

In the short term, we plan to add modeling of system commands, telemetry, and flight rules. The flight rules define operational constraints of the underlying system. For example, on ISS an outside light is physically paired with a heater. A flight rule states to prevent the light from freezing, the light and the heater cannot both be off. This simple example can easily be written using SWRL. By streaming live data in the ontology, we can check the both the consistency of the ontology and the underlying system health. For performance reasons, we are investigating using Tractable reasoning infrastructure for OWL 2 (TrOWL) to perform stream reasoning.

We are also working to allow PRONTOE to export to two advanced planning languages, ANML and PDDL. This will allow planners to directly use ontological data to create plans and schedules. This feature is important to EVAs due to their tight schedules and complexity of their execution.

We've recently added the ability for PRONTOE to drive the 3D visualizer DOUG. By clicking on any ontological entity within PRONTOE with a physical location on ISS, PRONTOE algorithmically directs DOUG's camera to an appropriate location to view the entity and flashes it if possible. We plan to integrate other ISS views and databases into PRONTOE, further enriching the ISS ontology.

7 CONCLUSIONS

By leveraging domain specific window widgets and an easy to use development environment, PRONTOE allows operators and other domain experts to develop and maintain ontologies. Importing tools and connections to a wide variety of data sources allows PRONTOE to easily capture system data from a wide variety of sources. Kernels, reasoners, and integration into version controls systems and workflows allows a user of PRONTOE to maintain large and dynamic ontologies. Useful ontologies were built for both NASA and robotic domains for automation and affordances respectively. In the future, we plan to add better modeling of system commands and telemetry. All PRONTOE domains will benefit from procedure executives and planners using the developed ontology. Based on our case study, the benefits for NASA operations by using PRONTOE are:

- Make available a consistent domain model that need not be reproduced for each automation application;
- Unify the often disparate sources of EVA and Core ISS System information;
- Provide for rapid update of ISS configuration information, thus allowing automated services to provide results based on the most recent data;

- Provide a consistent view of the domain so as to minimize error in operating ISS;
- Model a set of core concepts for dynamic system monitoring and control that have been proved out in disparate domains such as robotics.

ACKNOWLEDGEMENTS

This work is funded by a NASA Small Business Innovation Research (SBIR) grant. The authors grateful to Dr. Jeremy Frank of NASA Ames Research Center for his help with this project. The authors also wish to thank the numerous NASA flight controllers who have worked with us over the last several years to get the correct data into our ontology. Kevin Kusy was instrumental in creating import scripts for external databases. The development of an ontology of robot affordances for the simulated Atlas robot was funded under a DARPA Phase I Small Business Technology Transfer (STTR) contract.

REFERENCES

Bell, S. and Kortenkamp, D. (2011). Embedding procedure assistance into mission control tools. In *Proceedings of the IJCAI Workshop on AI in Space*.

Cardoso, J. (2007). The semantic web vision: Where are we? *IEEE Intelligent Systems*, 22(5):84–88.

Elsaesser, C. and Sanborn, J. (1990). An architecture for adversarial planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(1).

Fox, M. and Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124.

Herzig, A. and Varzincak, I. (2007). Metatheory of action: Beyond consistency. *Artificial Intelligence*, 171:951–984.

Izygon, M., Kortenkamp, D., and Molin, A. (2008). A procedure integrated development environment for future spacecraft and habitats. In *Proceedings of the Space Technology and Applications International Forum (STAIF)*. Available as American Institute of Physics Conference Proceedings Volume 969.

Kortenkamp, D., Bonasso, R. P., and Schreckenghost, D. (2008). A procedure representation language for human spaceflight operations. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

Lopez, F. M. (1999). Overview of methodologies for building ontologies. In *Proceedings of the IJCAI Workshop on Ontologies and Problem-Solving Methods*.

Myers, K. (2000). Domain metatheories: Enabling user-centric planning. In *Proceedings of the AAAI Workshop on Representational Issues for Real-World Planning Systems (AAAI Technical Report WS-00-07)*.

Norman, D. (1998). *The Psychology of Everyday Things*. Basic Books.

Soares, A. and Fonseca, F. (2009). Building ontologies for information systems: What we have, what we need. In *Proceedings of iConference*.