Enhancing NASA's Procedure Representation Language to Support Planning Operations

Pete Bonasso*, Mark Boddy†, Dave Kortenkamp*

*TRACLabs, Inc. 1012 Hercules, Houston, TX 77058

†Adventium Enterprises, 111 Third Avenue South, Suite 100, Minneapolis, MN 55401

Abstract

Automation and autonomy are key elements in realizing the vision for space exploration. The NASA Exploration Technology Development Program (ETDP) has been developing several core autonomy capabilities, one of which is called a procedure representation language (PRL). PRL can be automatically translated into code that can be executed by NASA-developed autonomous executives. Another type of automation being developed by ETDP is automated planning aids. These will be needed to increase the number of missions that existing levels of flight personnel are able to handle. But PRL has few constructs to enable automated planners and schedulers to take advantage of the procedures resulting from PRL. In a continuing research effort, we have been developing extensions to PRL to add planning information – resource, constraints and sub-procedural information – so as to produce code useable by automated planning software. From a representative scenario for the PHALCON and EVA flight disciplines, we have derived requirements for planning, developed XML tags for the PRL changes, and translated the changes into the ANML planning language. This paper describes these results.

Motivation & Approach

Automation and autonomy are key elements in realizing the vision for space exploration. The NASA Exploration Technology Development Program (ETDP) has been developing a set of core autonomy capabilities that can adjust the level of human interaction from fully supervised to fully autonomous. One of those technologies is the development of a procedure representation language (PRL) that both captures the form of traditional procedures and allows for automatic translation into code that can be executed by NASA-developed autonomous executives (Bonasso et al 2003, Verma et all 2006). PRL provides for access to spacecraft and habitat telemetry, includes constructs for human centered displays, allows for the full range of human interaction, and allows for automatic methods of verification and validation. But most important, PRL is being developed with a graphical authoring system that enables non-computer specialists to write automated procedures (Kortenkamp et al 2007).

However, PRL is in a relative infancy with regard to supporting many of the autonomous software components being developed by NASA, specifically automated planners and schedulers. PRL has few constructs to enable automated planners and schedulers to take advantage of the procedures resulting from PRL. Such planners and schedulers generate new plans arising from new situations that often involve multiple concurrent tasks by multiple humans or human-robot teams. These plans will be composed of interleaved PRL-generated procedures, the execution building blocks of mission plans. As a result, automated planners and schedulers will require knowledge of the purpose of the PRL procedures, the resources used and any execution constraints that apply. Moreover, in order to make flexible plans, planners may only need to execute portions of existing procedures, so they need to know how such procedures can be usefully decomposed.

In support of ETDP, we have been investigating casting three key types of planning information into PRL, so as to support the construction of models that can be employed by modern AI planners. Our approach was to study a typical scenario involving two representative NASA-JSC flight disciplines, derive the resource, constraints and sub-procedural information, develop PRL extensions for that information and then translate that information (manually at first and then with translation software) into the newly-developed planning language ANML (Smith & Cushing 2008). Using the PRL extensions and their AANML translations, we have demonstrated the ability to generate plans (manually, for the moment) for the scenario equivalent to those used by the flight controllers.

This paper describes the selected scenario, the planning constructs that we derived, the subsequent changes to the PRL and examples of the resulting PRL and ANML.

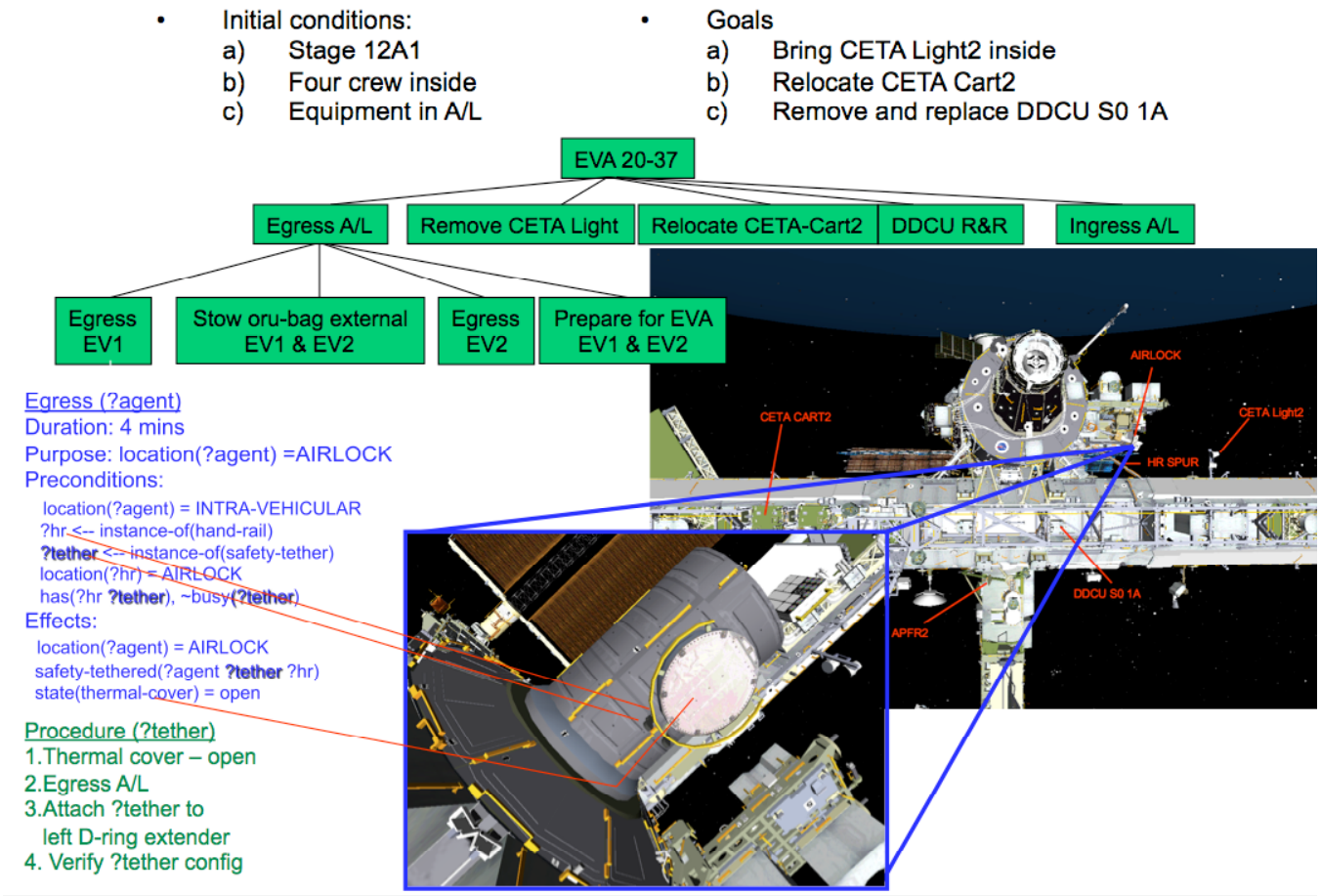

Figure 1 A partial expansion of a plan in our scenario. Starting with an initial configuration of the space station, a number of crew and a set of goals, the resulting plan consists broadly of the crew (EV1 and EV2) emerging from the airlock (A/L) with the appropriate tools and spares, carrying out each task and then returning to the airlock. The pictorials show the ISS in both plan-view, with all the locations of work as well as a zoom-view of the airlock, the location of the egress procedure. The green text to the left is the information one would find in the Word files of the SODFs for the one-person egress procedure. The Blue text shows the planning information associated with that procedure derived from our analysis of the flight control procedures. The variable ?tether is in bold to designate it as a resource used in the egress action.

Scenario

When we began this research, the PHALCON (Power, Heat And Light CONtrol) and EVA (Extra Vehicular Activity) flight controllers had recently conducted a mission wherein one of the EVA tasks was the removal and replacement of a DC-to-DC Converter Unit (DDCU). That task was actually a "get ahead" task that was added to the end of a sequence of previously planned EVA tasks. Due to space limitations, we will cover only the main EVA tasks for the six-hour scenario:

1) Crew egress the airlock
2) Retrieve CETA (Crew and Equipment Translation Aid) Light 2
3) Relocate CETA-cart 2 from the P1 truss to the P3 truss
4) Remove and Replace the DDCU 1A on the S0 truss
5) Crew ingress the airlock

Not only was this scenario sufficiently complex for our purposes, but also, since the PHALCON had to prepare and shutdown the old DDCU and power up the new DDCU, it involved two very different flight disciplines. The PHALCON represents the typical command center flight discipline whose activity consists in monitoring telemetry from a ground console, uplinking commands to various equipment on the space station, and coordinating with other flight disciplines, such as attitude control and life support. While there is an EVA flight controller, the locus of the EVA work is necessarily with the crew carrying out the EVA tasks. Thus with this one scenario we were able to investigate the planning requirements of a range of flight control activities.

Deriving Planning Information

Knowledge engineering is key to constructing a planning model as well as to extracting the critical parts of a domain ontology. The PHALCON tasks were derived from systems operations data files (SODFs), word files that were part of the repository of flight controller procedures. For the EVA tasks, we needed to go over the written description of the full six-hour scenario because the EVA flight controllers develop each EVA activity essentially from scratch. While the activities of connecting and disconnecting equipment to and from the station are fairly routine (though new equipment requires astronauts to test the procedures in the neutral buoyancy facility at JSC), each EVA is unique both in the number and types of tasks and in the starting locations of the equipment and tools used in the tasks.

But these written documents have little or no planning information associated with them, such as activity duration, purpose, preconditions or constraints. To derive this planning information, we started with a given EVA or PHALCON procedure, discussed each step with the cognizant flight controller, and then rewrote the procedure to include the information required for planning. We then used this information to construct a planning model for the tasks involved. This procedure analysis and distillation resulted in a structured view of the tasks in our scenario.

A hierarchical task net of the EVA tasks with a partial decomposition is shown in Figure 1. Consider the egress airlock task. When the two astronauts exit the airlock, first one goes out and attaches to a safety tether. Then the second astronaut hands out certain equipment (in this case the bag to hold the CETA-light to be retrieved) to the first before exiting himself. The green text in Figure 1 shows the procedure one would find in any number of previous EVA procedures for the single person egress. Simply, it says to open the thermal cover, go out and tether up. The key choice here is which safety tether to use. The PRL from the procedure alone would look like the following (PRL is in XML so we use a pseudo-PRL for illustration):

Step 1
  Manual Instruction: Thermal cover to open
  Manual Instruction: Egress airlock
  Manual Instruction: Hook Safety-tether to D-ring on suit
  Verify Instruction: Verify Safety-tether configuration

Our derived planning information associated with that procedure shows that the point of the procedure is to get an inside human agent located outside the airlock, the duration is typically 4 minutes and it requires that there be an unoccupied safety tether outside the airlock. A bookkeeping side effect is that the thermal cover will be open when this procedure completes.

Derived Sub-procedures

From our analysis we distilled seven intermediate sub-procedures and ten leaf-level PHALCON procedures, and thirteen intermediate and 48 leaf-level EVA sub-procedures. An example of a partial breakout of sub-procedures for PHALCONs is shown in Figure 2. When viewed from a planning perspective, the DDCU shutdown consists of a group of tasks that can be done many hours

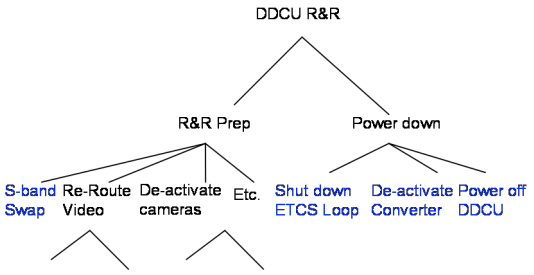

Figure 2 A breakdown of the PHALCON sub-procedures for a DDCU Shutdown. Light blue text indicate leaf level (non-decomposable) plan nodes.

prior to the EVA crew arriving at the DDCU site, and another group that is executed when the EVA crew is 30 minutes away from arriving at the site. The first group involves a mix of intermediate and leaf sub-procedures (shown in light blue) that are executed in conjunction with other flight disciplines that will be affected by the shutdown. The actual number of these sub-procedures used in a plan will depend on the state of the hardware of the sub-disciplines at the time of the DDCU R&R EVA. The second set of actions is leaf-level sub-procedures that include powering down and electrically isolating the DDCU.

In several procedures, for example, the S-BAND-swap,

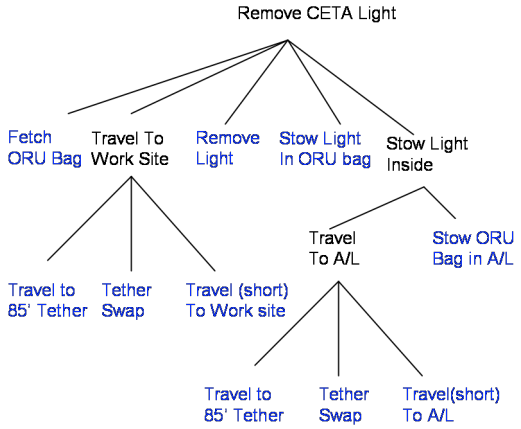

Figure 3 A breakout of EVA sub-procedures for removing a CETA light.

the PHALCON serves as a kind of supervising agent, and another flight controller, such as the CATO (Communications and Tracking Operations) flight controller, serves as the executing agent, who lets the PHALCON know when the procedure is complete.

An example of the breakout of sub-procedures for EVAs is shown in Figure 3. This breakout shows how to retrieve the CETA light with an agent at the AIRLOCK. The agent gets the ORU (orbital replacement unit) bag for the light, travels to the light's location, removes the light and stows it in the bag, then takes it back to the airlock. The interesting aspect of this procedure, and all EVA procedures that involve moving the agents from place to place is the limitation of the safety tether. If the target location is farther away than the 55' safety tether can reach, the agent has to travel to the location of another tether, usually of the 85' variety, and perform a tether-swap before moving on. In essence, this action can be applied at any level of the plan where a translation by free flying over handrails is required.

Unlike the multi-agent PHALCON procedures, where one agent is the coordinator and one is the executor, mulit-agent EVA procedures involve two agents working together to accomplish the task. In the case of the replacing the DDCU, for example, during the installation of the spare DDCU, agent1 has the old DDCU attached to his body restraint tether and agent2 holds the spare DDCU on a stanchion mount cover. The coordinated exchange is as follows:

1) Agent2 presents the spare DDCU to agent1.
2) Agent1 grabs the spare via a scoop on the spare and agent2 detaches the spare from the cover
3) Agent1 inserts the spare in position and then presents the old DDCU to agent2
4) Agent2 attaches the cover on the old DDCU and then stows it
5) Agent1 bolts the spare in place

This kind of coordinated effort as well as the fact that each agent can be doing a different task at the same time, motivated us to include specific temporal constraints in our PRL enhancements described in the next section.

PRL Enhancements

The PRL enhancements derived from the procedure analysis above include those for time, roles, resources and pre- and postconditions.

Time

Information about time and timing constraints is critical to planning operations. PRL has very little information about time encoded in it. Therefore we include these additional tags as part of our planning extensions:

Duration. PRL has no way of expressing the expected duration of a procedure. This is critical for planning purposes. The duration tag will not only take an expected duration in any number of time units, but will also allow a range (upper and lower bound) to be expressed. As well, the duration tag will allow the invocation of a computation with other local variables, such as when computing the travel time along a translation path.

Allen interval algebra. While timing constraints between procedures, steps, and instructions can be captured using convoluted constructs of pre- and postconditions, the ability to express intervals would be a useful syntactic sugar. At the procedure level these would call out any required timing constraints between procedures. When used at the step or instruction level these would guide execution within the procedure. We are adopting a subset of Allen's interval algebra (Allen 1983) as tags in the planning extension. This subset includes: before, meets, overlaps, starts with, during, finishes with and equal to.

Roles

Roles define the actors who will be performing the procedures. The actors filling a given role in the current PRL must be defined in the procedure (as a string). We have modified the role tag in PRL to be a formal parameter that can later be resolved with some constant defined as part of the planning problem statement (i.e., a particular astronaut). We also extended the role tag to allow for multiple roles in a procedur,e which was required in order to represent intermediate-level tasks, which may decompose into separate but synchronized procedures, each of which must be executed by a separate agent.

Resources

Resources are critical to planning operations. PRL says very little with respect to the resources that a procedure requires or uses. We propose these additional tags with respect to resources for our planning extension. PRL does not keep track internally of resource levels so that is left to an execution engine or resource management system. These tags therefore will need to be tied to an external representation of resources specific to each application.

Uses. This tag denotes that the procedure uses a certain number of a countable resource, such as tether-points. This tag includes a resource name and a numeric value expressing how many of that resource is used. The assumption is that there needs to be this number of the resource available before procedure execution. They are only freed up if a busy tag is employed as well (see next item). Thus, if a safety tether is used in a procedure, without a busy designation, it will continue to be "consumed" until another action, such as un-tethering, asserts they are not busy as a result of the action.

Busy. This tag denotes that a particular resource is busy for the duration of the procedure. This resource must be

free at the start of the procedure, but may or may not be freed up at the end of that procedure.

Consumes. This tag denotes that a procedure uses up a certain amount of this resource, such as oxygen, during the execution of the procedure. This tag includes a resource name and a numeric value expressing how much of that resource is used. The assumption is that there needs to be enough of the resource available before procedure execution and that the resource is decreased by the numeric value at the end of the procedure. Invoking a function can also derive the consumption value.

Produces. This tag denotes that a procedure restores a certain amount of this resource, such as power, during the execution of the procedure. This tag includes a resource name and a numeric value expressing how many of that resource is produced. Invoking a function can also derive the production value.

Planning preconditions

PRL has preconditions already defined, used as checks prior to procedure execution. Planning models need a different form of precondition, used to constrain what constitutes a legal plan, or to make inferences about the results of actions with conditional effects. Both forms of precondition must appear in the extended PRL dialect described here. The planning preconditions are used in the planning process in the expected way. The execution preconditions, also known as "go-conditions" serve multiple functions. First, they may (but need not) appear as planning preconditions. Second, they cause the generation of an explicit "check" action as part of the generated plan, indicating that at that point in execution the specified condition should be verified. Third, they are passed through to the PRL procedures tied to the primitive actions in the generated plan, so that the relevant conditions will in fact be checked.

Planning effects

The situation with effects is similar to that of preconditions. PRL has postconditions, which express the states in which the system needs to be for procedure success. Planners need a form of postcondition as well which because it is different from the PRL precondition will only appear in the planning data. A subset of the planning preconditions will be further tagged as the purpose (or objective) of the procedure. There may be many postconditions, only a subset of which constitutes the real objective of the procedure. Therefore, we've added the purpose tag so that the planner knows to add to the plan an explicit check for that condition and to signal a plan failure if it is not achieved.

Obtaining Consistent Semantics with ANML

Not all automated planning and scheduling systems use the same algorithms and representations for plans and not all of them interface with executives in the same way. For example, hierarchical task net (HTN) planners search for actions in a tree of actions that form a task hierarchy similar to military plans. Interval planners place actions in a timeline and adjust start and end conditions to meet deadlines. Some automated planners can send single tasks to an executive for execution; others post mostly complete plans. As the plan or task is executing, some

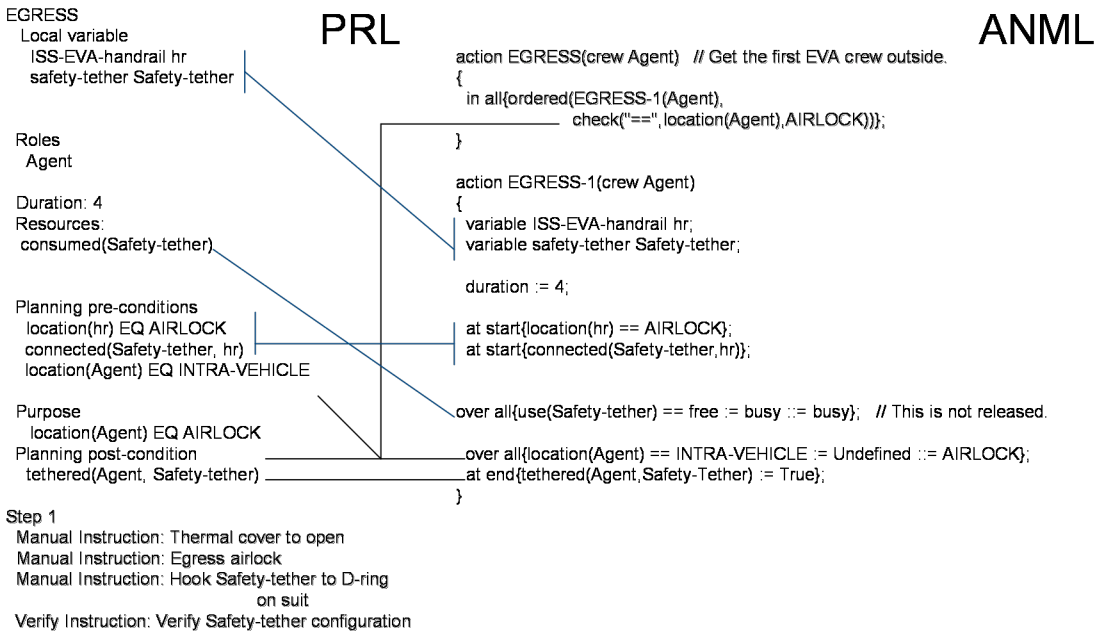

Figure 4 A sample translation of PRL to ANML for the Egress action

planners require a minimum amount of execution status, others may require more. Such differences in planning systems will have an impact on the number and types of additional information that the new PRL must provide.

One way to address the disparity among planning systems is to provide the PRL enhancements with a rigorous semantics by showing they can be translated to one or more of the planning domain modeling languages in current use by the artificial intelligence planning community. We have selected the ANML planning language (Smith & Cushing 2008), because it is based on strong notions of action and state (so planners like Aspen (Chien 2003) can use it), uses a variable/value model (so planners like Europa (Pell 1998) can use it), supports rich temporal constraints, and provides simple, convenient idioms for expressing the most common forms of action conditions, effects, and resource usage. The language supports both generative and HTN planning models (so HTN planners like AP (Applegate et al 1983) can use it) in a uniform framework and has a clear, well-defined semantics.

PRL to ANML Mapping

Figure 4 shows an example translation of the PRL from the EVA egress action described in above to an ANML representation. Each PRL item maps directly from a PRL tag to a given ANML construct as described in the previous section. Of particular note, however, is the handling of the purpose clause. Besides being one of the effects of the action and thus contributing to the change in location of the agent in an "over all" clause, an auxiliary intermediate action is generated that decomposes into this action and a runtime check for the stated purpose.

While this example shows how we tie PRL planning information to procedural information, we also need PRL files that instead of a procedure show the decomposition of intermediate planning actions. Figure 5 shows such a file and its translation into ANML. In this DDCU R&R task, an intra-vehicular agent, IV1, will operate the SSRMS, while two EVA crew carry out the task. EV2 must prepare the SSRMS in the task called SSRMS_Setup. This task is referenced via the PRL Call procedure tag, which is a reference to another PRL file that has further task decomposition. In this case, the task will decompose into fetching and installing an Articulating Portable Foot Restraint (APFR) on the SSRMS, ingressing the SSRMS and then having IV1 move the SSRMS to the work site. At the same time, EV1 will be setting up the worksite by fetching the spare DDCU and moving it to the work site. Finally, the two

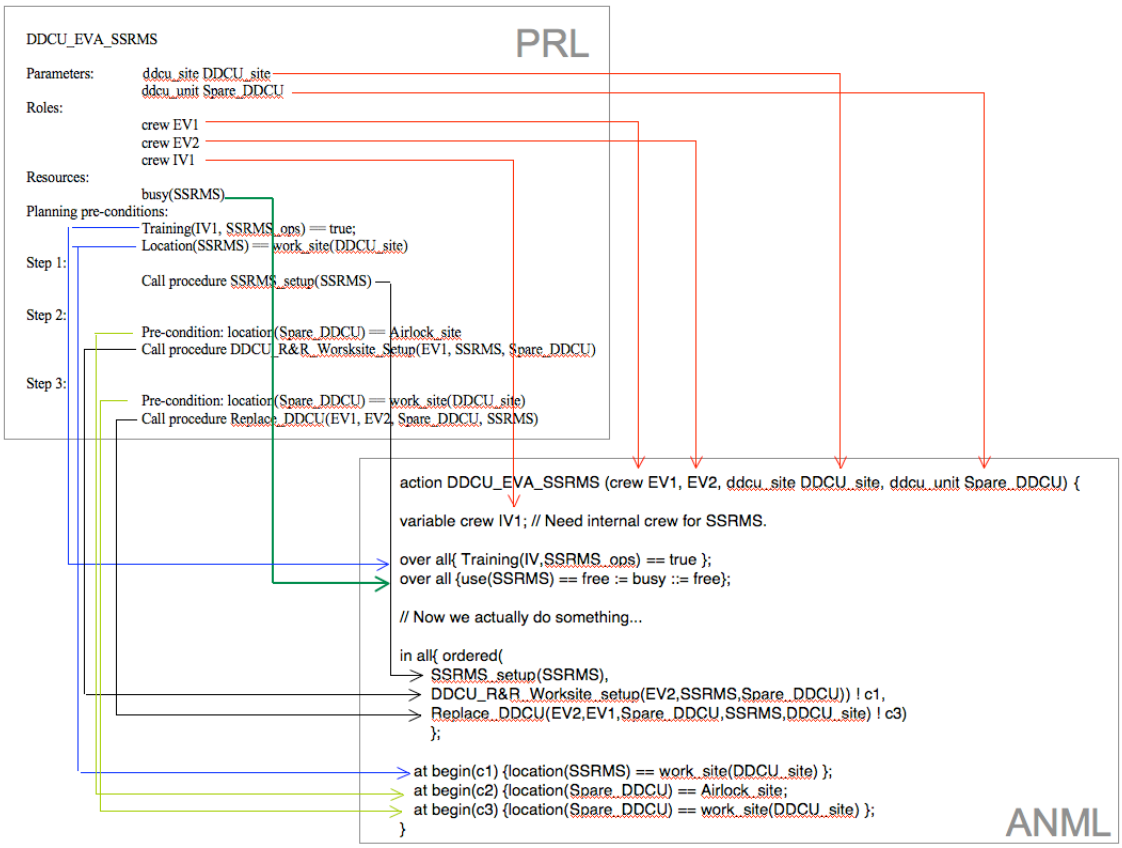

Figure 5 A PRL to ANML translation of an intermediate action

EVs will remove the old DDCU and install the spare. The temporal ordering in PRL assumes a sequence and we use the unordered block tag to allow for operations that just need to finish before the next step in the sequence. In ANML, a task tag, e.g., c1, allows us to attach preconditions to tasks.

ANML Modeling Choices

Based on the task analysis and distillations conducted in Phase1 we have a preliminary modeling approach to be used for representing EVA plan data in Phase2. We will extend these model choices in Phase 2 as we 1) determine the availability of interfaces to ISS configuration files, and 2) extend our domain to cover additional tasks, crew, tools and equipment.

Use of Action Decomposition. We have chosen to employ action decompositions for a number of reasons, not the least of which is that the flight controllers tend to think of their planning tasks in a hierarchical way. Additionally, it allows us to construct part of the plan using compiled mission planning knowledge rather than explicitly modeling domain interactions. We also realize search efficiency, since it seems in the PHALCON and EVA domains, there are only a small number of legal ways to carry out tasks. Finally, the hierarchy allows us to scope for reserving crew/equipment. For example, in DDCU-EVA example above, several different resources are reserved at a level above where they are actually employed, so as to ensure that the operation does not start if they are not available. This can be done in a flat representation, but not as easily, especially not for execution, since one wants to ensure that the given item remains reserved.

Agent Assignment Restriction. ISS procedures to date only assign one crew member per procedure, including in situations where the overall plan requires coordinated action by multiple agents. We handle this restriction by using intermediate tasks with multiple, temporally constrained leaf actions, one for each agent needed.

Types. We use types to “dispatch" planning actions, in the sense that whether a given action can be used with a given set of variable bindings (unifications, co-designations, etc.) can be determined by type. We can do this without getting into the difficulties of type inference, by enforcing the closed-world assumption, i.e., all objects are known and can be enumerated, and that any type differences on which actions are tested for applicability partition the set of all objects. For example, the equipment type partitions all entities in the domain model into equipment and not-equipment. The INSTALL DDCU action partitions carriers into ceta cart, SSRMS, and not-a-carrier. Our subtypes are as follows:

Things.

- object – enumerated type.
- equipment – a subset of object.
- ceta-cart – a subset of equipment.
- scoop – a subset of equipment.
- pgt – a subset of equipment.
- brt – a subset of equipment.
- tfr – a subset of equipment.
- power-jumper – a subset of equipment.
- med-oru-bag – a subset of equipment.
- rover – a subset of equipment.
- pgt-socket – a subset of equipment.
- ratchet – a subset of equipment.
- ects-loop – a subset of equipment.
- pgt+tm – a subset of equipment.
- ORU-tether – a subset of equipment.
- stanchion-mount-cover – a subset of equipment.
- ISS-EVA-handrail – a subset of equipment.
- symbol – enumerated type.

People.

- people – subset of object
- crew – subset of people.
- fc – (flight-controller), subset of people.

Places.

- location – a vector of three symbols: < segment, bay, face >

What a location is may be context-dependent. For a rover it might be waypoints. Even on ISS, for crew moving around on EVA, rails should count at least as transitions between locations, if not locations themselves.

- path – an ordered sequence of locations.

Functions.

- label(equipment) → symbol
- installed-on(equipment) → equipment – domain is installed equipment, range is what it’s installed on. Making this a function rather than a predicate is cleaner in a lot of ways, but requires a special value for objects that aren’t installed on anything, which is easier to represent with a predicate.
- on(crew) → equipment – like installed-on, but let’s not use the same function for equipment and people.
- path-duration(crew location location symbol) → duration – how long will it take a given agent (currently crew-only), to move from one location to another, using a means denoted by a symbol? The model should eventually include the possibility of a non-human agent.
- plan-path(location location) → path
- O2-needed(crew path symbol) → number – how much (excess?) oxygen is consumed by crew in traversing path using the means denoted by a symbol?
- O2-level(crew) → number
- fuel-level(equipment) → number

- work-site(equipment-site) → location
- location(movable-object) → location.
- choose-ceta-cart() → ceta-cart
- get-current-operator(SSRMS) → person
- get-current-wif-adaptor(SSRMS) → wif-adaptor
- get-current-APFR(SSRMS) → apfr
- state(object) → symbol
- compute-movement-duration(SSRMS,location,location) → number
- get-current-adco() → person
- get-current-mcs() → person
- get-current-ctg() → person
- get-current-cato() → person
- get-current-odin() → person
- get-etcs-loop(ddcu) → etcs-loop
- power-string(equipment) → symbol
- comms-state(equipment) → symbol

Predicates.

- installed(equipment-site, equipment)
- use(equipment) – not quite a unary resource. More like a critical section (think "busy").
- use(location) – as above.
- use(crew) – as above.
- has(object, object) One object is connected to another (either could be a person)
- tethered(equipment, equipment) – one piece of equipment is fastened to another.
- tethered(equipment, crew) – a piece of equipment is fastened to a crew member. Also, tethered(E,C) implies has(C,E)
- tool-match(equipment, equipment) – match a (particular type of) tool to another piece of equipment. In the absence of a real object model, this is done by enumeration.
- training(person, symbol)
- connected(object, object) – generalization of tethered.
- empty(equipment)

Special Values and Constants.

- Undefined -- part of ANML. Permits no inference to be drawn, positive or negative. There is a type issue involving this and other constants. We plan to use a separate constant for each type, e.g., undefined-boolean, undefined-equipment, etc.
- *Nothing* -- Special value that matches no other member of the class, such as what equipment is at the site of a DDCU that has been removed.

By the end of our first year of research we had modeled all of the scenario tasks in ANML and PRL to ensure that our modeling approach was complete.

ANML as a Target Language

Our choice of ANML as a target language for this project was made after consideration of a wide range of alternatives, and presented both advantages and disadvantages. As discussed above, in this domain, having the ability to express both HTN and classical planning information in a uniform semantics is very useful. Additional constructs available in ANML that have proved beneficial include a very flexible model for specifying temporal constraints and parameter co-designation among subactions, and an explicit resource model.

As also previously mentioned, ANML is a relatively new language, and as such is still under development. The downside to this include the fact that there is currently no planner that accepts ANML as input, though that lack is currently being addressed on several fronts, including the ongoing development of ANML translators to PDDL and NDDL. Another, less significant implication of using a language under development is that the syntax has changed under us. The syntax used here is that given in (Smith & Cushing 2008). Readers of more recent papers describing ANML will have encountered a different syntax, for which we are currently updating our models.

However, there are also benefits to being early adopters. As the developers of one of the first significant planning models written in ANML, we have been in a position to push on what ANML can and cannot do, which has led to some small changes in both syntax and semantics, in ways that make the modelling task somewhat easier, for example, in defining the semantics of execution-time choices versus parameters that are determined when the plan is constructed.

Summary

Our key findings with regard to the potential of capturing and storing planning information online are as follows:

- Through analysis of complex PHALCON and EVA procedures we determined that in fact, flight controllers plan with actions that are actually sub-procedures of the formal procedures maintained online.
- After identifying the specific sub-procedures we were able to distill from them the key information with regard to time, resources and pre- and postconditions that could potentially be used by automated planning software
- Further, we determined that such planning information could be given consistent semantics by developing a mapping from the PRL to a formal planning language, namely ANML.

Based on the above we hypothesize that planning software being developed under NASA funding should be able to use the new PRL files via an ANML translation to provide automated aids for flight controller mission planning. Some planners use the New Domain Description Language (NDDL) (http://react.cs.uni-

sb.de/mbt2007/slides/raimondi.pdf). A partial ANML to NDDL translator for the Automation for Operations (A4O) program has already been implemented. Many planners use the Planning Domain Description Language (PDDL) (Fox and Long 2003). An ANML to PDDL translator is now under development. These efforts will greatly extend the number of planners that can be applied to domain models captured in ANML.

Our current efforts are focused on developing an interactive planning tool that uses the PRL planning information to automatically generate plans as good as the ones previously developed for the scenario described in this paper.

Acknowledgements

We are indebted to NASA-JSC flight controllers Wayne Wedlake and Dave Crook for providing extensive domain expertise for this work. This effort is funded through SBIR grant NNX09CA17C, sponsored by Dr Jeremy Frank of NASA's Ames Research Center.

References

Allen, J.F., Maintaining Knowledge About Temporal Intervals. Communications of the ACM, 1983. 26(11): p. 832-843.

Applegate, C., C. Elsaesser, and J. Sanborn, An Architecture for Adversarial Planning. IEEE Transactions on Systems, Man, and Cybernetics, 1990. 20(1): p. 186-194.

Bonasso, R.P., D. Kortenkamp, and C. Thronesbery, Intelligent Control of a Water Recovery System: Three Years in the Trenches, in AI Magazine. 2003. p. 19-44.

Chien, S., et al. Autonomous Science on the Earth Observer One Mission. In i-SAIRAS 2003. 2003. Nara, Japan.

Kortenkamp, D., R.P. Bonasso, and D. Schreckenghost. Developing and Executing Goal-Based, Adjustably Autonomous Procedures, in AIAA InfoTech@Aerospace Conference. 2007.

Fox, M. and D. Long, PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research, 2003. 20: p. 61-124.

Pell, B., et al., An Autonomous Spacecraft Agent Prototype. Autonomous Robotics, 1998. Vol. 5

V. Verma, A. Jónsson, C. Pasareanu, and M. Iatauro, Universal Executive and PLEXIL: Engine and Language for Robust Spacecraft Control and Operations, American Institute of Aeronautics and Astronautics Space 2006 Conference.

Smith, D.E. and W. Cushing, The ANML Language, in iSAIRAS. 2008: Los Angeles, CA.