Embedding Temporal Constraints For Coordinated Execution in Habitat Automation

Paul Morris, Mark Schwabacher, Michael Dalal*, and Charles Fry**

NASA Ames Research Center
Moffett Field, CA 94035, U.S.A.

*Stinger Ghaffarian Technologies, Inc. **Dell Services Federal Government
Moffett Field, CA 94035, U.S.A.

Abstract

Future NASA plans call for long-duration deep space missions with human crews. Because of light-time delay and other considerations, increased autonomy will be needed. This will necessitate integration of tools in such areas as anomaly detection, diagnosis, planning, and execution. In this paper we investigate an approach that integrates planning and execution by embedding planner-derived temporal constraints in an execution procedure. To avoid the need for propagation, we convert the temporal constraints to dispatchable form. We handle some uncertainty in the durations without it affecting the execution; larger variations may cause activities to be skipped.

Introduction

Future NASA plans for the next decade call for long-duration deep space missions with human crews. Because of light-time delay and other considerations, it may not be feasible to micro-manage the mission from the ground, as was done for the Apollo missions to the Moon. Thus, increased autonomy is needed. Also, the crews will likely be small, perhaps with as few as four members. This poses a problem because such a small crew is unlikely to possess the expertise needed to deal with unexpected eventualities. The solution may lie in increased automation to help fill gaps in knowledge and relieve the crew of tedious housekeeping activities.

Habitat Automation Project Element

The Autonomous Systems (AS) Project is developing software, sensors, and other technology to automate the operation of systems that will be needed for future NASA missions, such as spacecraft, habitats, and propellant loading systems. One of the two major Project Elements within AS is called Habitat Automation (HA). HA is focused on automating the operation of habitats that could be used to house astronauts on the surface of a moon, planet, or asteroid, or on the way there. HA is testing its technology using the 2nd Generation Deep Space Habitat (Figure 1), a prototype habitat at NASA Johnson Space Center. Within HA, we are exploring the use of advanced automation tools to support such tasks as planning, plan execution, anomaly detection, diagnosis, and recovery. Several such tools have been developed in recent years. Our current focus is on integrating these tools so that they work together to provide a smooth operational capability.

Figure 1: Deep Space Habitat, 2nd Generation

The existing standalone tools include the Advanced Caution and Warning System (ACAWS), planning and scheduling tools called Scheduling and Planning Interface for Exploration (SPIFE) and Extendable Uniform Remote Operations Planning Architecture (EUROPA), and an execution system called Plan Execution Interchange Language (PLEXIL). The following are brief descriptions

of the tools.

ACAWS

ACAWS is a comprehensive system health management tool composed of components working in tandem to support spacecraft operators in determining the health state of a system, detecting anomalous health conditions, determining the root cause of detected failures, determining the effects of those failures on the system and the mission timeline, and recovering from the failure or mitigating its effects to accomplish as much of the planned mission as possible (Colombano *et al.* 2013). For diagnosis (determining the failed components and the failure modes), ACAWS uses a commercial product known as TEAMS-RDS (Testability, Engineering, and Maintenance System Remote Diagnostic Server) from Qualtech Systems, Inc (Qualtech 2013). For determining the effects of failures, ACAWS uses a tool developed at NASA Ames Research Center called the Failure Consequence Assessment System (FCAS). For anomaly detection (detecting sensor data that is significantly different from what has been seen in the past), ACAWS uses the Inductive Monitoring System, which was also developed at Ames (Iverson *et al.* 2012).

SPIFE-EUROPA

SPIFE-EUROPA is a combination of the SPIFE (Marquez *et al.* 2010) plan editing tool and the EUROPA (Frank and Jonsson 2003) automated planning tool (included with SPIFE as a plug-in). The SPIFE tool includes a sophisticated plan database and graphic user interface that detects violations that can be fixed by the user. EUROPA supports automatic planning with temporal, state, and numeric constraints. SPIFE and EUROPA, separately and in combination, have been used for ground planning in several NASA missions.

PLEXIL/PRL

PLEXIL (Verma *et al.* 2005) is a language for specifying plans for automation. The open-source PLEXIL distribution (http://plexil.sourceforge.net) provides tools for executing PLEXIL plans, interfacing with external systems to command, and monitoring plan execution graphically. It also provides tools for constructing simulators useful for standalone development. The Procedure Representation Language (PRL) is a language for specifying higher-level human procedures, in which information needed for automation may also be embedded (Kortenkamp *et al*. 2008). PRL is used with a graphical tool called Procedure Integrated Development Environment (PRIDE) to author procedures, which then can be automatically translated into PLEXIL (Figure 2, left-side) for automated execution under varying levels of automation (Dalal and Frank 2010). These technologies provided an integrated system for authoring and executing Deep Space Habitat procedures.

The complete integration of these tools is work in progress. In the past year, we investigated interactions between ACAWS and PLEXIL, and between SPIFE-EUROPA and PLEXIL. The latter combination that integrates planning and scheduling with execution is the primary topic of this paper.

One of the goals behind the integration of these tools is to enable the automated or semi-automated replanning of crew activities and habitat operations in response to system failures. For example, after a component of the habitat fails, ACAWS will detect the failure, determine which component has failed, and determine the consequences of the failure. ACAWS will communicate this information to SPIFE-EUROPA, which will then replan the remainder of the mission to either repair the failed component or replan activities to avoid using the failed component. Finally, PLEXIL will execute the revised plan. This final step requires the integration of SPIFE-EUROPA with PLEXIL, an early version of which is described in this paper.

Plan/Execution Experiment

In this paper we describe an experiment that involves a loose coupling (Figure 2, right-side) between EUROPA and PLEXIL. In this approach, the SPIFE-EUROPA tool is used offline to produce a plan that includes temporal constraints to coordinate high-level activities. A separate PRL tool called PRIDE is used to author lower-level PLEXIL procedures that produce commands to implement the high-level activities (Figure 2, left-side).

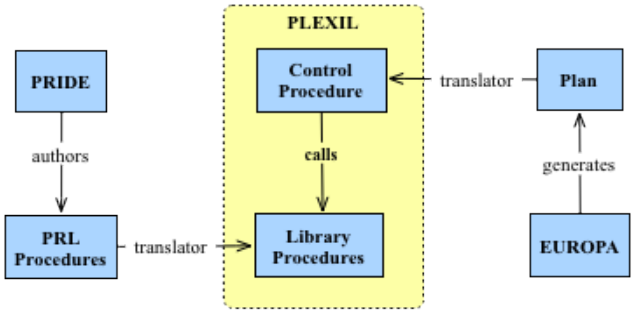

Figure 2: Plan Transfer Process

The experiment involves a routine survey task where the outside of the habitat is scanned for micro-meteorite impacts. The survey is repeated every day and can be skipped or truncated if need be. In the experiment, the survey is interrupted by a high-priority task involving a fluid transfer. Depending on how long the fluid transfer takes, the remainder of the survey may be resumed or skipped.

The PLEXIL tool executes (Figure 3) the plan obtained from EUROPA. It automatically sends commands to simulators of the relevant parts of the Deep Space Habitat (DSH), including the photo-survey and fluid transfer. Manual commands can also be sent along a separate route

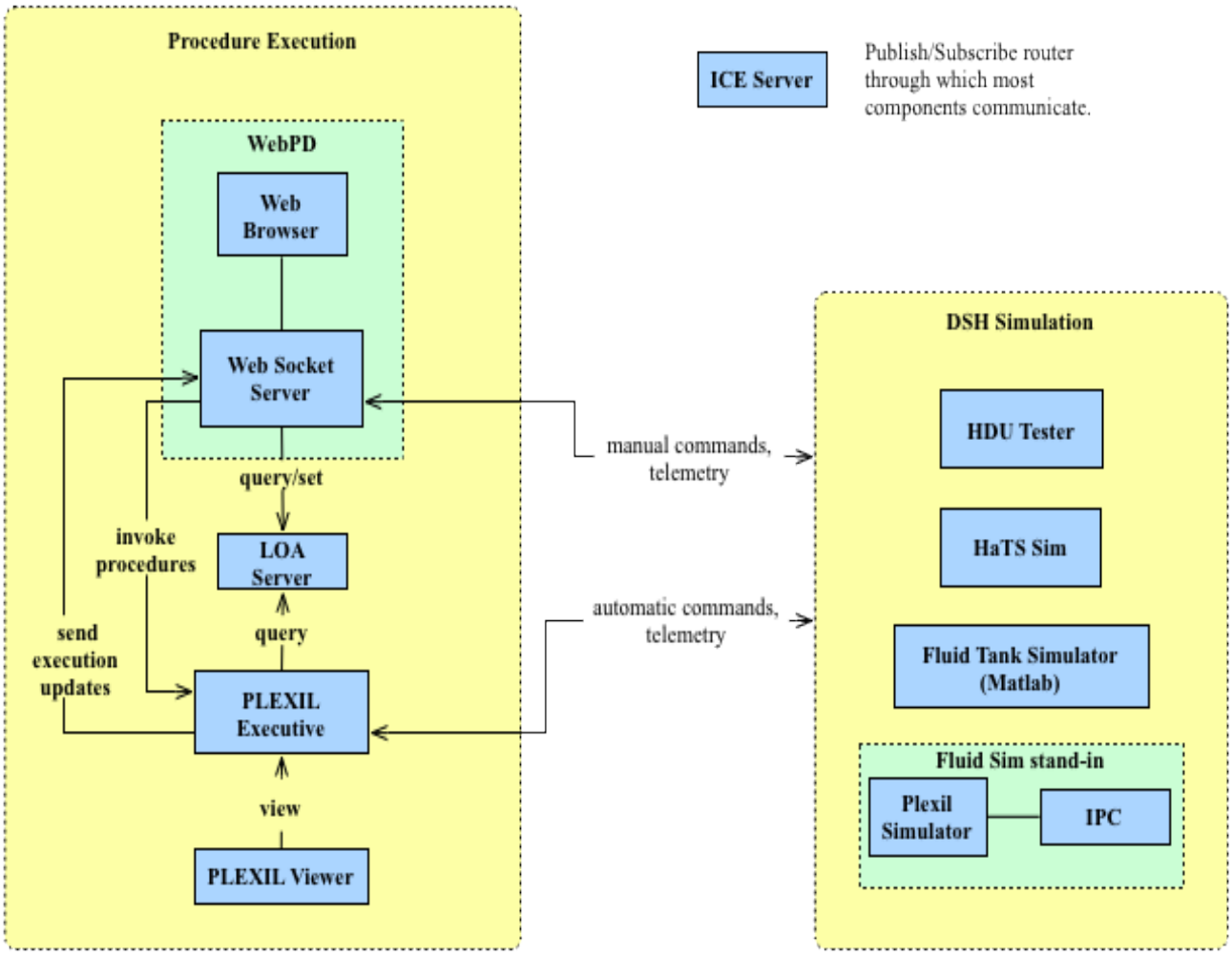

Figure 3: Plan Execution Process

from a web browser. Displays showing the execution are available on both routes.

EUROPA/PLEXIL Comparison

Before describing the integration of EUROPA and PLEXIL, we first compare and contrast their relevant features.

For this task, we use a specialized version of EUROPA, called DynamicEuropa, that is connected to a plan editor front end, called SPIFE. This combination provides for a Mixed-Initiative Planning framework (Bresina and Morris 2007) where a human operator collaborates with the EUROPA suite of automated tools to create a plan.

The EUROPA database provides for a plan consisting of a set of activities that are interrelated via a rich set of temporal, state, and resource properties. In particular, it includes a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991) to support planning and ensure consistency of the temporal constraints. The STN determines a plan that has *temporal flexibility*. That is, it corresponds to a set of related schedules rather than a single schedule. The temporal flexibility is intended to provide scope for adjusting to temporal deviations during execution. Instead of having a single time, the events in the plan have a lower and upper bound on when they can occur.

Despite this, the human operator of SPIFE sees a single *nominal* schedule, which is easier to grasp. When adjusting the plan to satisfy constraints, EUROPA restricts the flexible plan to exclude constraint violations. This may also exclude the existing nominal schedule. In that case EUROPA computes a new nominal schedule that satisfies the constraints, while minimizing the changes from the previous one, and communicates it to SPIFE. This

approach serves to maintain general plan stability and allows the human operator to express simple timing preferences. The nominal schedule can also provide heuristic guidance to an automated planner (Morris *et al.* 2011) through the ordering of events.

EUROPA can eliminate certain resource and state violations in the nominal plan by inserting temporal constraints that prevent the violations. These make the plan "safe" from such violations as long as the constraints are satisfied.

In contrast to EUROPA, which is a constraint-based declarative language, PLEXIL is a procedural[1] language similar in many ways to an ordinary programming language. It can start and stop "activities" that are further decomposed by PLEXIL into specific commands to external systems. It does not include a specialized representation of temporal constraints or a temporal network that can propagate their effects. However, a PLEXIL plan can specify conditions for when and how activities execute. These include start-conditions, preconditions and skip-conditions. An activity is delayed until its start-condition becomes true. If a skip-condition tests true, an activity will be skipped. Failure of a precondition will cause an activity to fail and be terminated; thus, it resembles an error condition. The activity end events can have similar conditions. All the condition forms allow inequality tests of computed values and Boolean combinations of those tests.

The PLEXIL tool also has an ability to determine the current clock time (and date). In conjunction with the activity conditions, this provides a low-level ability to observe temporal requirements.

This paper focuses on temporal constraint issues in the offline integration of a declarative planner and a procedural execution system. There is extensive work on integration of planning and execution in other contexts (e.g., Rajan, Py, and Barreiro 2012; Cash and Young 2009).

EUROPA/PLEXIL Integration

As discussed earlier, the SPIFE-EUROPA tool allows the user to specify nominal times for activities. For our experiment, we wanted activities to execute at their nominal times unless they were compelled to do otherwise by temporal constraints. However, the durations of actions corresponding to external processes are generally uncertain to some degree. Thus, the start times of later activities can be pulled earlier or pushed later than the nominal time in order to satisfy temporal constraints. An example of this is where the later activity is constrained to immediately follow the completion of an external process.

As mentioned, PLEXIL does not have a capability to perform temporal propagation as in an STN. This raises the issue of how to do any pushing and pulling required by the temporal constraints. The solution we adopted is to convert the temporal plan to *minimum dispatchable* form (Tsamardinos, Muscettola, and Morris 1998). In a dispatchable plan, propagation can be restricted to immediate neighbors, and can be done in a lazy way when a neighboring activity is considered for execution. The minimum dispatchable plan has the minimum number of neighbors needed to accomplish this.

This means that the requirement to start at the nominal time, or earlier or later as needed, can be embedded in the PLEXIL start condition. The general form of the start condition is

$$time \geq \max[lb,\ lb1, lb2, ..., \min[nom, ub, ub1, ub2, ...]]$$

where *time* is the current clock time. Here lb and ub are the original lower and upper bounds in the flexible plan. The lb1, lb2, etc. values are additional lower bounds derived from local constraints with neighboring events in the dispatchable plan and the actual execution times of those events. Similarly, the ub1, ub2, etc. values are derived upper bounds, while *nom* is the nominal start-time of the activity. Note that the min operator permits the activity to start in order to satisfy an upper bound constraint even if the nominal start time has not yet been reached.

Another issue is what to do if an activity start misses its deadline (upper-bound in the temporal network). In our experiment, the appropriate response for a delayed part of the survey was to simply skip it, since the survey was envisaged as being non-critical and repeated every day. Thus, the skip condition is used to enforce missed upper bounds. Similarly to the start conditions, embedded tests derived from the minimum dispatchable plan can be evaluated when the activity is considered for execution. The general form of the skip condition is

$$time \geq \min[ub, ub1, ub2, ...]$$

where *time* is the current clock time. In the actual implementation, we include a "slop factor" in the upper bound values to allow for execution latency.

We also deal with the issue of uncertainty in the execution times. Small or localized amounts of temporal uncertainty can be planned for in advance by using algorithms based on Dynamic Controllability (DC) (Morris 2006). (Large amounts of uncertainty may cause the plan to not be Dynamically Controllable, which necessitates execution-time repair of the plan in adverse cases.) The DC algorithm essentially trades planning-time flexibility

[1]In its pure form, called *Core PLEXIL*, the behavior of a plan, including its actual control flow, is completely determined by conditions defined on each node of the plan. In this sense PLEXIL has a strong underlying declarative property, though for practical purposes, and to the reader of plans authored in its standard syntax, PLEXIL is a procedural language.

for execution-time flexibility that covers the uncertainty. That is, it tightens earlier temporal constraints in order to provide slack for a later uncertain interval. For example, suppose an activity A has an uncertain duration of between 1 and 2 hours and another activity B is required to start at least 1 hour before A ends. If the uncertainty were treated as ordinary STN slack then B could start up to 1 hour after A starts because the end of A could be then delayed until 1 hour after that. However, if the plan is to robustly cope with the uncertainty, then B needs to start no later than the time A starts. From the point of view of the planning process, temporal uncertainty may be viewed as negative flexibility, and the DC algorithm may be regarded as transferring slack from one event to another to compensate for the negative flexibility. In the example, the range of start times for B is trimmed, which expands the range of end times for A to contain the interval of uncertainty.

The DC algorithm may add disjunctive *wait* conditions to some events; these can be folded into the PLEXIL start conditions. For example, suppose A is as above, but activity B is instead required to start *at most* 1 hour before A ends. Then to be safe, B must wait until either A has finished or 1 hour has elapsed since A started. This introduces an additional *wait* term into the start condition maximization

$$time \geq \max[..., wait(A,w), ...]$$

where $wait(A,w) = \min[A_E, A_S+w]$ and A_E, A_S, and w are the end time of A, start time of A, and wait time (1 hour in this example), respectively.

For the purposes of the experiment, we assumed 1% uncertainty for all the durations and applied the DC algorithm. Besides exercising the uncertainty methods, it is desirable in any case to introduce some slack in the durations in order to counteract an idiosyncratic feature of the dispatchability algorithm: When a duration is fixed, the algorithm may reformulate temporal constraints on the end-time as functionally equivalent constraints on the start-time. This reformulation tends to obscure the activity precedences in the plan and discards the potential for event-driven execution.

Discussion and Closing Remarks

The loose-integration approach of embedding temporal constraints in the PLEXIL start and skip conditions worked well for the purpose of the experiment. However, it pointed to certain issues that would need to be addressed in more general contexts.

Even where skipping is an acceptable means of dealing with a failing execution, there is the issue of where and how much to skip. If an activity is skipped, or if it is predicted that an activity will be skipped, then clearly any other activity that "depends on it" should also be skipped. However, in general the temporal information, especially in minimum dispatchable form, is not enough to determine that. For example, one activity may be constrained to precede another because it establishes a precondition for it. In that case, if the preceding activity is skipped, the following activity should also be skipped. (Conversely, if it is predicted that the following activity will be skipped, and if there is no other reason for the precondition, the establishing activity, now useless, may as well be skipped also.) However, another reason that one activity precedes another may be because they contend for some resource, and so are mutually exclusive. In this case, the precedence is merely a constraint imposed by the planner to prevent overlap, and there is no reason to skip the following activity just because the preceding one is skipped.

One approach to dealing with the issue of "skippable chunks" has been used in planning for Mars rovers. The activities are grouped into "observations" and the whole observation is skipped if any essential part of it is unfeasible. A more general approach would require the explicit modeling of "support" relationships that are distinct from the temporal ones.

Skipping is a simple expedient for dealing with failed activities but plan repair in general requires more sophisticated approaches. For example, there may be other useful activities that we can substitute for the failed ones, or there may be alternate ways of achieving our objectives.

We can identify a spectrum of potential plan repair techniques that involve increasingly drastic modifications to the existing plan. These are:

- Reordering, postponing, abandoning
- Resource switching, alternate methods
- Novel combinations of primitives
- Creating new operators and models

The full capabilities of the planner will be needed for more complex plan repairs. In the future, we will investigate an "online" integration where EUROPA and PLEXIL are running at the same time and communicating.

The last category in the repair spectrum goes beyond current planner technology. As the repair solutions become more "creative," issues of predictability and trust arise. In the foreseeable future, it is unlikely that a computer system will have the comprehensive background and experience needed to fully evaluate the consequences of creative solutions. Thus, caution would dictate that such solutions should be presented as suggestions rather than being subject to execution by a fully-automated system.

References

Bresina, J., and Morris, P. 2007. Mixed-Initiative Planning in Space Mission Operations. *AI Magazine* 20:2.

Cash S. and Young, R. 2009. Bowyer: A Planning Tool for Bridging the gap between Declarative and Procedural Domains. In *Proceedings of the 5th Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE-2009).* Palo Alto, CA.

Colombano, S., Spirkovska, L., Aaseng, G., Schwabacher, M., Baskaran, V., Ossenfort, J., and Smith, I. 2013. A System for Fault Management, Including Fault Consequence. In *43rd International Conference on Environmental Systems (ICES)*. Reston, Virginia: American Institute of Aeronautics and Astronautics. To appear.

Dalal, K. M., Frank, J. 2010. Bridging the Gap between Human and Automated Procedure Execution. In *IEEE Aerospace Conference*, 2010.

Dechter, R., Meiri, I., and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49:1-3.

Frank, J. and Jonsson, A. 2003. Constraint-Based Interval and Attribute Planning. *Journal of Constraints* 8:4 Special Issue on Constraints and Planning.

Iverson, D.L., Martin, R., Schwabacher, M., Spirkovska, L., Taylor, W., Mackey, R., Castle, J.P., and Baskaran, V. 2012. General Purpose Data-Driven System Monitoring for Space Operations. *Journal of Aerospace Computing, Information, and Communication* 9:2.

Kortenkamp, D., Bonasso, R. P., Schreckenghost, D., Dalal, K. M., Verma, V., and Wang, L. 2008, A Procedure Representation Language for Human Spaceflight Operations. In *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-2008).*

Marquez, J. J., Ludowise, M., McCurdy, M., & Li, J. 2010. Evolving from Planning and Scheduling to Real-Time Operations Support: Design Challenges. In *Proceedings of 40th International Conference on Environmental Systems.* Barcelona, Spain.

Morris, P. 2006. A Structural Characterization of Temporal Dynamic Controllability. In *Proceedings of the 12th International Conference on Principles and Practices of Constraint Programming (CP-2006).*

Morris P., Bresina J., Barreiro J., Iatauro, M., and Smith T. 2011. State-Based Scheduling via Active Resource Solving. In *Proceedings of the 4th IEEE International Conference on Space Mission Challenges for Information Systems (SMC-IT 2011)*.

Qualtech Systems Inc. 2013. Web site, http://teamqsi.com

Rajan K., Py F., and Barreiro J. 2012. Towards Deliberative Control in Marine Robotics. In *Marine Robot Autonomy*, M. Seto Ed.: Springer Verlag.

Tsamardinos, I., Muscettola, N., and Morris, P. 1998. Fast Transformation of Temporal Networks for Efficient Execution. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98).*

Verma, V., Estlin, T., Jónsson, A., Pasareanu, C., Simmons, R., and Tso, K. 2005. Plan execution interchange language (PLEXIL) for executable plans and command sequences. In *Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS2005)*, Munich, Germany.