Research Trends in Autonomous Space-Based Planning and Scheduling

Gina Moylan and Ella Atkins

University of Maryland Space Systems Laboratory
382 Technology Drive
College Park, MD 20742
gmoylan | ella @ssl.umd.edu

Abstract

There are many research currents in the application of planning and scheduling to space-based systems. In reality, however, there have been few working examples. We believe it is a valuable exercise to review some of the work in this domain with a focus on some of the problems that are responsible for the limited proliferation thus far of automation in this domain. We then examine these limitations in the context of an important future space application that inherently depends on automation—self-assembly. We conclude with an objective synthesis of the strengths of these approaches and the common research recommendations we have found based on this review.

Introduction

The pursuit of automation strategies in space began shortly after the very first spacecraft were launched (Chory et al. 1986). While different experiments in autonomous navigation and guidance were conducted in the decades that followed, it would be well over three decades until the first application of autonomous spacecraft scheduling operations was successfully applied. This happened in 1994 during a lunar surveying mission called Clementine (Sorenson et al. 1995). During one of its routine orbits, a rules-based script[1] triggered the execution of command sequences based upon receipt of a ground-supplied state vector. Though this was a successful scheduling exercise, it would not be until 1999 that an integrated planning and scheduling (P&S) system would be flown on board a spacecraft. The Remote Agent Experiment (RAX)—a three tiered (Gat 1997) P&S architecture—flew on the Deep Space One satellite (Benard et al. 1999) for days, conducting closed-loop planning and execution that resulted in the successful application of batch commands to control the spacecraft (Jónsson et al 2000). This technology was further improved by the first successful application of a continuous integrated planning and scheduling system in 2004. The capable 3-tiered Autonomous Sciencecraft Experiment (ASE) flying aboard the Earth Observing One (EO-1) spacecraft has been enabling continuous response and coverage of real-time science events and efficient dissemination of the data obtained since 2004 (Chien et al. 2004; Chein et al. 2005; Sherwood et al. 2005). Thus, the integration of P&S for all elements of the spacecraft—including its application instrumentation—has finally been realized.

Clearly the last decade has shown much progress in the automation of space-based systems. It would seem that the successful application of space-based P&S technologies is rounding the bend of a "J-curve." However, as researchers, industry and government professionals in this domain well know, there are many significant challenges to the wide-spread realization of space-based automation. These include:

- The dearth of hardware and software standardization
- The extensive time-frames associated with the design, construction and implementation of space-systems
- Economic and practical difficulties of launch
- Complexity of spacecraft
- The harshness of the operating environment(s)
- Limited on-board resources and communication opportunities
- The complexities of implementing dynamic real-time response systems
- Political and fiscal challenges, etc.

Though the success of the ASE application has definitively shown that a fully embraced P&S automation architecture can work in space, there are questions as to the future direction of these technologies. This is especially evident when we consider space-based missions requiring the interdependent autonomous operations of multiple spacecraft (e.g., Atkins, Moylan and Hoskins 2006; Izzo et al. 2005; Curtis et al. 2003; Estlin et al. 2002; Frank et al. 2001) or deep space missions with prohibitive communication delays (Sherwood et al. 2005).

[1] The script was written in Spacecraft Command Language (SCL).

To this end we make a small contribution in this paper by synthesizing some of the work in this field. We present a literature review and then isolate the corresponding research trends that present themselves with a focus on some of the problems that confront the proliferation of automation within this domain. We then discuss these limitations from the perspective of an important space-based application necessarily requiring autonomy—self-assembly. We conclude with a compilation of recommendations from the research we have surveyed.

Selective Research Review

The job of P&S in any domain is to generate a set of low-level activities from a set of high-level goals without violating any system rules or constraints. Depending on the scale of the application and the domain in which it is applied, this can become a very challenging problem. As discussed in the introduction, P&S in the domain of space-based systems is particularly challenging due to the various impediments involved before mission planning even begins. Consequently, we find many different approaches and few real-world implementations. However, as we will show, many aspects of these approaches tend to be similar. This review is organized first by flight-proven architectures and then by other proposals and frameworks.

Flight-Proven P&S Systems

RAX. As discussed in the introduction, the RAX system was the first primary flight-proven integrated example of P&S for satellite operations (Benard et al. 1999). The Remote Agent's (Muscettola et al. 1998) planning system was a compilation of the Heuristic Scheduling Testbed System (HSTS) that provided a temporal database and modeling facility and an incremental refinement search engine (IRS). The HSTS implemented a least-commitment search and constraint propagation strategy, resulting in reduced search space and flexible plans. This system differs from traditional hierarchical task network (HTN) planners in that it distinguishes between allowable transitions and heuristics to guide the search. This decoupling allows mission specialist to validate domain knowledge and algorithm specialist to generate heuristics for performance (Chien et al. 1998).

ASE. The ASE system encapsulates two planning systems—The high-level deliberative Continuous Activity Scheduling Planning Execution and Replanning system (CASPER) and the lower-level spacecraft command language (SCL). The current SCL implementation is a derivative of a previous system called ASPEN, whose development fell out of the same HSTS that RA's planning system was based on.

Since the time of RAX the ASE system has been constantly improved upon, tightening algorithm performance and increasing the system's capabilities. The ASE system differs from the RAX system in two fundamental respects: its ability to implement autonomous spacecraft control with continuous planning rather than batch-driven commands and its ability to use accurate and effective science models to achieve science-driven autonomy.

ASE is predicated on a tight coupling between the onboard planner, CASPER/ASPEN, and the detailed models of spacecraft constraints and science data analysis. They key reasons for a successful application of the ASE P&S system can basically be distilled to (Sherwood et al. 2005):

- Redundant functionality—especially with respect to spacecraft safety
- Streamlined engineering—the software was reduced by more than half its original size before upload in terms of models and efficiency
- Accurate spacecraft, mission constraints and science models
- Use of iterative repair methods
- Using fixed values for temporal offsets—greatly reducing search space

Drawbacks to the ASE architecture include:

- Slow build releases due to very consuming testing times
- Reduced observational flexibility to increase response time (can be improved with more onboard resources)

The continuous planning abilities of the CASPER system is primarily accomplished by using iterative repair techniques (Rabideau et al. 1999; Chein et al. 2000). By classifying constraints and using heuristics, the iterative repair algorithm is able to make streamlined decisions and implement iterative repair to the plan based on the methods available given a set of constraints.

ASE has surpassed its original charter, prompting NASA to continue using it on a regular basis aboard EO-1 due to its very successful science returns and the many dollars its saving in operations. The success of this system has largely been a result of an evolved engineering approach that combines accurate spacecraft models and empirically derived heuristics with a capable temporal planner and highly redundant architecture, designed to build safety in each layer.

In conjunction with the autonomous operations of the EO-1 spacecraft, a "sensorweb"[2] facilitates the rapid response and dissemination of science data (Chien et al. 2005). By using a combination of low resolution sensors to detect broad brush science data and internet-based science agents to detect important events from rapid data processing, ground-based and spacecraft-based automated planning systems (ASPEN/CASPER) can generate command sequences to re-task spacecraft observations, issue the commands and then succinctly forward the processed data to appropriate science teams.

PROBA. The Project for On Board Autonomy (PROBA) is an ESA micro-satellite launched in 2001 that was designed to validate on-board and ground-station autonomy along with other technology experiments. Onboard functionality includes nominal operation/resource management and instrument commanding driven by user requests. The ground-segment autonomy encompasses spacecraft pass operations and performance evaluations, as well as high-level Internet user requests.

Like ASE, the successful data returns have compelled an extension of the original 1-year mission. Unlike ASE, however, PROBA does not use extensive modeling features and does not work autonomously with anomalies.

Following the success of this mission, a mission called Proba-2 is currently under development with a launch scheduled for September 2007.

Other Architectures and Frameworks

IDEA. Another architecture, Intelligent Distributed Execution Architecture (IDEA) (Muscettola et al. 2002), though it has not flown aboard a spacecraft yet, builds on the success of the flight-tested RAX system (Muscettola et al. 1998). IDEA is a distributed temporal planner centered upon the interval of an agent's procedural execution, called a token. Agents use communication wrappers to send messages that initiate the execution of procedures by other agents or to receive goals (tokens). Token parameters are used by an agent's internal problem solving algorithms to decide its next action.

IDEA agents are not restricted by a communication or control topology involving multiple agents. A central model controls agent communication by describing which procedures can be exchanged with which external agents. It also specifies which procedure arguments are expected to be determined before a goal is sent to another agent (input arguments) and on which arguments the agent is expecting execution feedback from some other agent executing the token (output and status arguments). Agents communicate via external inter-process communication (IPC) mechanisms. These design strategies enable IDEA to implement a simple sense-plan-act loop as a virtual machine within a communication wrapper, delivering reactive-based real-time guarantees.

Muscettola et al. also present an accompanying framework to the IDEA architecture. The framework was designed to unify both the representation and inference mechanisms required to accommodate the reactive and deliberative behaviors in an agent. They accomplish this by creating a common language applicable to all levels of abstraction, along with a tightly integrated reactive executive component.

CLARAty. The Coupled Layer Architecture for Robotic Autonomy (CLARAty) (Volpe et al. 2001) makes a departure from traditional 3-tiered architectures (Gat 1997) by tightly coupling the planer and executive into one decision layer that interacts with a functional layer at all levels of the system. The advantages of this two-tired approach yield an explicit 3D representation of system layer resolution and the blending of the declarative and procedural techniques for decision making. CLARAty provides a single database to interface planning and executive functionality which increases system efficiency and time for iterative repair capabilities.

Volpe et al. cite several challenges to be overcome in the realization of a unified mobile robotics architecture. These challenges can be basically distilled down to an astigmatism in the research community, where automation architecture is primarily designed, created and implemented. Automation research—being technically challenging and promoted by disparate sources of funding with independent agendas—is subject to cultural biases. Quoting from Volpe,

> Simply put, no one architecture can be optimal for all problems, and if one is too flexible it quickly loses any structure that gives it value.

Vople at al. argue that community acceptance, spanning the various divides and leveraging standard practices and existing software are key ingredients to a successful solution.

CLEaR. To realize autonomous sequencing capabilities for NASA planned rover mission, the Closed-Loop Execution and Recovery (CLEaR) system was developed (Estlin et al. 2002). It consists of a hybrid controller built on top of the CASPER planner and a TDL (Task Description Language) executive system. By building upon the CLARAty framework previously reviewed, CLEaR provides a tightly coupled approach to coordinating goal and event-driven behavior. Shared plan information and continual updates of state enables effective planner and executive interaction that results in shared responsibility for decision making. Heuristic

[2] a networked set of instruments in which information from one or more sensors is automatically used to reconfigure the remained of the sensors, as defined by Chien (Chien et al. 2005)

support allows CLEaR to decide when certain plan conflicts should be handled by the planner vs. the executive. For example, they illustrate a rover navigation problem, whereby the executive eventually fails plan changes due to a violation of planner-allocated time constraints, relenting control to the more globally aware planner.

LOGOS & ACT. Some searchers are focused on the autonomic properties of an autonomous system (Truszkowski et al. 2006). Truzkowski maintains that autonomous system operation will vary over time, preventing the same patterns of behavior on a recurring basis, thus making system validation a difficult and critical component of these systems. His group at NASA GSFC has focused on the development of two prototype agents: Lights-Out Ground Operations System (LOGOS) and the Agent Concept Testbed (ACT).

LOGOS is a community of 10 JAVA-based software agents that cooperate to automate operator functionality. Some agents interface with legacy software, some perform services for the other agents in the community, and others interface with an analyst or operator. ACT agents use an adaptable component architecture where components can be easily removed/replaced to be consistent with new AI technologies as they become available, without necessitating significant redesign or development.

Truskowski et al. contend that these agents embody the four basic properties of autonomic systems—self-configuring, self-optimizing, self-healing, and self-protecting—and embrace specification and verification methods required by autonomy to guarantee correct operation. They demonstrate the effectiveness of these systems by applying them to a hypothetical scenario involving future technologies of NASA's Autonomous NanoTechnology Swarm (ANTS) mission.

IXTET-EXEC. A framework to integrate deliberative planning, plan repair and execution control that takes into account resource level updates and temporal constraints has been proposed by Lemai and Ingrand (Lemai and Ingrand 2004). It is based on lifted partial order temporal planning techniques which produce flexible plans and allow conditional plan repair interleaved with plan execution. A tight coupling between the planner and the executive implements a cycle that integrates external messages, repairs the plan—if needed—and executes actions. Cycles enable this interaction by taking into account runtime failures and timeouts, updating the plan accordingly.

Interleaving plan repair and execution is very desirable as some parts of the plan may remain valid and executable while being temporally flexible, thus allowing the postponement and insertion of actions. Their plan repair algorithm uses non linear planning techniques under certain assumptions. The proposed framework has been implemented in IXTET-EXEC, using the planning system IXTET. The authors intend to test their implementation using various rover simulations.

Tickets. Another architecture strategy revolves around the idea of viewing commands rather than states as resources (Gat 2000). Complex real-time control structures are encapsulated into objects called tickets—software objects that act as intermediaries between control processes and low-level commands. By imposing restrictive access to commands and computational overhead limits, hard-real-time application software becomes more reliable and easier to verify. Gat considers autonomous spacecraft control abstractly as a computational process whose outputs are spacecraft commands. He also defines resources as anything that causes dependencies among components of system state and thus leads to potential goal conflicts.

Autonomy Trends and Limitations

As Kortenkamp points out (Kortenkamp 2003), the mission planning phase of human spaceflight and uncrewed spacecraft/rovers are quite similar in that both require the collaboration of humans and planning technologies well before a mission begins—the significant departure is in the plan's execution.

Table 1 Some Qualities of Reviewed Planners (NOT FINISHED)

	ASE	RA	PROBA	IDEA	CLEaR	IXTET-EXEC
Flight-Tested	•	•	•			
Hardware-Tested	•	•	•		•	
Continuous Planning	•			•		•
Anomaly resolution	•			•		•
Plan repair during execution						•
Scales with multiple agents	•			•		
Common representation				•		
Flexible timing						•
Full navigation & guidance						

After reviewing some of the widely accepted and proposed architectures/frameworks in this domain, we begin to see certain themes fall out that clearly represent impediments to wide-spread space-based automation (see Table 1). Perhaps the most acutely recognizable theme arising in the application of space-based planning and scheduling technologies—indeed, in automation in general—presents itself as *a disconnect between communication philosophies in planning action and executing it* (Gat 1997; Volpe 2001). To achieve automation on the levels that future missions call for—especially those involving multiple spacecraft—there must be greater unification in terms of automation strategies in the face of the ever-present unrelenting obstacles of space-based exploration. This seems to be realized more by industry and NASA professionals rather than by academicians.

Another theme that clearly divides researchers in industry/government and academia has to do with how spacecraft are put together. As we have seen, effective autonomy requires a tight coupling between application planners and executives. Each subsystem on a satellite or rover often has distinct components and idiosyncrasies requiring domain specialist to transfer expert knowledge for accurate spacecraft and instrument models. This makes the necessary process of validation extremely difficult depending upon implementation details, as each model change must be tested and verified in a lengthy software validation process. Thus, until spacecraft are more standardized, *validation at all system levels is prohibitive.*

An additional high-level theme involves a field-wide fundamental problem with P&S technology. To date, there are very few approaches that are designed with temporal constraints in mind. This is absolutely essential to the real-time, unpredictable, dynamic domain of space-based P&S. *Durative actions, goal deadlines, and their effects on the planning and execution processes must be capably modeled and executed efficiently—especially in space domains* (Fox and Long 2003).

Though there are many more impediments to cover, we offer a final limitation that cuts to the core of automation technologies—fully autonomous guidance and navigation capabilities. While terrestrial action choices to optimize fuel use and time can be implemented with traditional AI planning tools, the optimization of these resources with respect to space operations requires full-state trajectories—i.e., geometric motion (a path) that includes point velocities/accelerations governed by nonlinear differential equations of motion. The difficulty in this is perhaps no where more apparent than in space-based self-assembly where self-actuated components must rendezvous/dock autonomously. As the action choices for each component and its corresponding trajectories scale with the number of assembly members, we begin to appreciate the difficulty of coordinating the dynamics of these systems in terms of motion and assembly/functional choices (Atkins, Moylan and Hoskins 2006).

Though the ASE architecture is a forerunner in the field currently and was designed to scale with multiple agents (planner communication via goals or execution layer), this application has not been tested in practice (Sherwood et al. 2005). Planning systems applied to self-assembly problems must integrate system trajectories and deliberative capabilities in a very cohesive way. In this respect, *more research in the area of autonomous navigation and guidance capabilities is* needed to implement applications involving multiple spacecraft, such as those found in space-based self-assembly/reconfiguration.

Common Recommendations

Despite the limitations we have discussed, there are many practical lessons that aid the further research and application of P&S technologies to space-based systems. The following is a summary of recommendations we have gleaned out of the work represented in this review. We present these in applicable categories, though there is obvious and necessary overlap.

Architecture

- Design fault protection at every layer for redundancy.
- Develop planning systems to interact with legacy control software and established s/c models.
- Unify the planning and execution processes under the same database and language/representation, making the executive and planner aware of the same constraints.
- Incorporate flexible time execution.
- Enable procedural capabilities during plan generation and repair.
- Develop more sophisticated techniques for dealing with activity/task failures involving exception handling.
- Design planning and executive systems to coordinate with other software to evaluate whether a science operation was successful—science goals drive remote exploration and therefore should be a top priority.

Software

- Design clean and well-specified interfaces so that automatic code generation can be used to create interface code, telemetry, model interfaces, and test cases.

- Allow spacecraft experts to encode models themselves with tools and languages already familiar to them.
- Organize vertically along spacecraft subsystems/ domain boundaries for better modeling.
- Design software to be flexible when state information in uncertain—this is especially important in ground-based missions.

Validation

- Formally validate basic system functionality prior to model development and flight insertion.
- Generate mission scenarios with graphical tools that allow visual inspection/modification with constraint checking to ensure consistency.
- Constraint platform use in terms of different types and fidelity.
- Run end-to-end tests as early as possible at all stages of development.
- Design telemetry early and use it as the only way of debugging/understanding the behavior of the system during integration, test, and operations.

Modeling

- Organize the models around the domain subsystems rather than around agent technology.
- Develop clear methods for characterizing agent behavior with models.
- Develop modular models with validation costs in mind—changes should be easy to implement and test.
- Design formal declarative agent models to facilitate the identification of conditions where performance is guaranteed.

Other

- Provide education for the technology and its value so that it is supported—seek understanding, support and feedback at all levels for acceptance and improvement.
- Every practical application of automation literature should include a "Lessons Learned" section—as systematically as an "Introduction" or "Conclusion"—so that researchers help to educate each other on what works without reinventing the wheel.
- Incorporate path planners that search for paths based on different constraints, e.g., shadows, communication, timing, terrain, etc.
- Focus on plan repair when things go wrong *and* when things go better than expected.

Conclusions

The complicated, harsh challenges of realizing space-based missions gives us unshakable realities which must be confronted if we are to experience wider-spread automation in this domain. While much of the research that has been conducted is promising, there must be a greater collaborative effort between researchers in industry/government and academia towards standardization and information exchange. In addition, more effort towards advancing temporal planning technologies is needed in this unpredictable and dynamic domain.

Despite the obvious advances in the space-based P&S technologies we have presented, there continues to be a significant gap in the approach to consolidate the planning and executive aspects of systems. The community seems to have achieved automating on-board scientists and s/c operators; however, there still seems to be a vacancy for a fully autonomous resident flight-dynamics expert. This is especially important in technologies inherently dependent upon the successful application of AI methodology to autonomous guidance and navigation. To effectively bridge this gap, AI and control researchers must jump the fence and explore each other's research areas. This requires relenting bias and extending ourselves to the various interdisciplinary challenges of automation—especially in space domains.

Acknowledgments

This research is supported by a National Science Foundation under the CAREER Award (0347461).

References

Atkins, E., Moylan, G., and Hoskins, A. 2006. Space-Based Assembly with Symbolic and Continuous Planning Experts. *In Proceedings of the IEEE Aerospace Conference.*

Bernard, D. Dorais, G. Gamble, E. Kanefsky, B., Kurien, J., Man, G.K., Millar, W., Muscettola, N., Nayak, P., Rajan, K.; Rouquette, N.; Smith, B.; Taylor, W.; and Tung, Y.-W. 1999. Spacecraft autonomy Flight experience: The DS1 Remote Agent experiment. *In Proceedings of the AIAA Conference.*

Chien, S., Knight, R., Stechert, A., Sherwood R., and Radibeau, G. 2000. Using Iterative Repair to Improve Responsiveness of Planning and Scheduling. *In Proceedings. of the International Conference on AI Planning and Scheduling*.

Chien, S., Sherwood, R., Tran, D,. Cichy, B., Rabideau, G., Castano, R., Davies, A., Lee, R., Mandl, D. Frye, S., Trout, B., Hengemihle, J., Dapos Agostino, J., Shulman, S., Ungar, S., Brakke, T., Boyer, D., Van Gaasbeck, J., Greeley, R., Doggett, T., Baker, V., Dohm, J., Ip, F. 2004. The EO-1 autonomous science agent. *In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*.

Chien, S. Cichy, B. Davies, A. Tran, D. Rabideau, G. Castano, R. Sherwood, R. Mandl, D. Frye, S. Shulman, S. Jones, J. Grosvenor, S. 2005. An Autonomous Earth-Observing sensorWeb. *IEEE Intelligent Systems and Their Applications,* 20:3, 16-24.

Chory, M.A., Hoffman, D.D., and LeMay, J.L., 1986. Satellite Autonomous Navigation -- Status and History. *In Proceedings of the IEEE Position, Location, and Navigation Symposium,* 110-121.

Curtis, S.A., Rilee, M.L., Clark, P.E., & Marr, G.C., 2003. Use of Swarm Intelligence in Spacecraft Constellations for the Resource Exploration of the Asteroid Belt. *The Third International Workshop on Satellite Constellations and Formation Flying*.

Estlin, T., Fisher, F., Gaines, D., Chouinard, C., S. 2002. Continuous Planning and Execution for an Autonomous Rover. *In Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*.

Fox, Maria, & Long, Derek (2003). PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR*. 20:61-124.

Frank, J., Jonsson, A., Morris, R. and, Smith, D.E. 2001. Planning and scheduling for fleets of earth observing satellites. *In Proceeding of the 6th Int. Symposium on AI, Robotics and Automation for Space*.

Gat, E. 1997. On three-layer architectures. *Artificial Intelligence and Mobile Robots. MIT/AAAI*.

Gat, E. 2000. Hard-real-time Resource Management for Autonomous Spacecraft. *IEEE*.

Izzo, D., L., Pettazzi & Ayre, M., (2005). Mission Concept for Autonomous on Orbit Assembly of a Large Reflector in Space. *56th International Astronautical Congress.*

Jonsson, A.K., Morris, P., Muscettola, N., Rajan, K., Smith, B. 2000. Planning in interplanetary space: theory and practice. *In Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems.*

Kortenkamp, D. 2003. A day in an Astronaut's Life: Reflections on Advanced Planning and Scheduling Technology. *IEEE Intelligent Systems and Their Applications*.

Lemai, S. and Ingrand, F.. 2004. Interleaving temporal planning and execution in robotics domains. *In Proceedings of the National Conference on Artificial Intelligence*.

Muscettola, N., Dorais G., Fry C., Levinson, R. and Plaunt, C. 2002. Idea: Planning at the core of autonomous reactive agents. *In Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Muscettola, N., Nayak, P.P., Pell, B. and Williams, B.C. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103(1-2):5-47.

Rabideau, G., Knight, R., Chien, S., Fukanaga, A., and Govindjee, A. 1999. Iterative planning for spacecraft operations using the ASPEN system. *In Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Rabideau, G., Chien, S., Willis, J., and Mann, T. 1999. Using iterative repair to automate planning and scheduling of shuttle payload operations. *In Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence*, 813-820.

Sherwood, R. Chien, S., Tran, D., Cichy, B., Castano, R. Davies, A. Rabideau, G. 2005. The ST6 Autonomous Sciencecraft Experiment. *IEEE Aerospace Conference*, 18(2):8-11.

Smith, D.E., Frank, J., & Jónsson, A. 2000. Bridging the Gap Between Planning and Scheduling. *Knowledge Engineering Review*, 15(1).

Sorensen, T.C., Oswald, D.C., Shook, R.M., Gaasbeck, J.V. 1995. Spacecraft Autonomous Operations Experiment Performed During the Clementine Lunar Mission. *Journal of Spacecraft and Rockets*, 32(6):1049-1053.

Truszkowski, W.F., Hinchey, M.G., Rash, J.L., Rouff, C.A., (2006). Autonomous and Autonomic Systems: A Paradigm for Future Space Exploration Missions. *IEEE Transactions on Systems, Man and Cybernetics*, Part C.

Truszkowski, Walt, & Rouff, Christopher. 2002. Progressive Autonomy: A Method for Gradually Introducing Autonomy into Space Missions," *SEW, 27th Annual NASA Goddard Software Engineering Workshop.*

Volpe, R. Nesnas, I. Estlin, T. Mutz, D. Petras, R. Das, H., 2001. The CLARAty architecture for robotic autonomy. *In Proceedings of the IEEE Aerospace Conference*.

Zweben, M., Daun, B. & Deal, M. 1994. Scheduling and rescheduling with iterative repair, in M. Zweben & M. S. Fox, eds, *Intelligent Scheduling*. Morgan Kaufmann, Orlando, FL, 8:121-140.