# Modeling Challenges in Procedure Synthesis for Earth Independent Anomaly Response

**J. Benton**[1], **Irina Kostitsyna**[1, 2], **Richard Levinson**[1, 2], **Alison Paredes**[1]

[1]NASA Ames Research Center
[2]KBR Wyle Services, Inc.
j.benton@nasa.gov, irina.kostitsyna@nasa.gov, richard.j.levinson@nasa.gov, alison.s.paredes@nasa.gov

## Abstract

Future long-duration human spaceflight will require Earth-independent operations, in which communication delays force crews to diagnose and respond to anomalies with limited real-time ground support. Procedure synthesis can support this need by constructing situation-specific diagnostic procedures from reusable fragments, which we call snippets. In practice, however, the central difficulty often lies in knowledge engineering rather than in search alone. Existing procedures, manuals, subject matter expertise, and systems models capture operational knowledge at different levels of abstraction and rarely provide the explicit preconditions, effects, and observation structure that a procedure planner requires.

This paper examines three recurring modeling challenges that arose in our work on anomaly response procedure synthesis: (1) representing information gathering and state-changing snippets without losing tightly coupled operational behavior, (2) defining snippet conditions in terms of system state rather than implicit procedure sequence, and (3) choosing simplifying assumptions that support tractable reasoning while preserving operational structure. Together, these challenges motivate a layered modeling approach in which modelers incorporate operational source information into systems models that capture valid configurations, topology, and snippet structure. These models are then exported to an intermediate domain representation containing plant-model state data and snippet-library information, which solver-specific translators use to produce procedure synthesis encodings.

## Introduction

Future human exploration space missions will require crews to handle anomalies with limited real-time support from the ground during Earth-independent operations. As communication delays grow beyond low-Earth orbit, crews must diagnose off-nominal situations, evaluate candidate responses, and execute procedures under uncertainty and resource constraints (Maddox, Vera, and Martinez 2024; Parisi et al. 2024). Procedure synthesis can help meet this need, and automated planning provides one important family of techniques for carrying it out.

The Earth-Independent Operations (EIO) Anomaly Response portfolio includes a procedure synthesis effort aimed at generating contingent diagnostic procedures from fault hypotheses. This work sits within NASA's broader EIO effort to equip future Mars crews to respond to onboard problems with substantially reduced real-time ground support (Maddox, Vera, and Martinez 2024). Figure 1 places this effort in the larger anomaly response process, from anomaly detection and fault diagnosis through procedure synthesis, procedure validation, and crew execution. In the broader portfolio, anomaly response also includes response and recovery activities; we focus here on the diagnosis-oriented procedure-synthesis portion of that larger problem.

Our efforts have focused on a pragmatic subset of procedure synthesis. Rather than attempting de novo synthesis of full procedures from first principles, we construct ad hoc, situation-specific procedures by recombining fragments of existing procedures, which we call *snippets*. A snippet represents one or more procedure steps and plays a role similar to a macro action, with preconditions and effects. This choice grounds the problem operationally, but it also forces modelers to convert human-authored procedural steps into declarative snippet models that can support novel recombination. As a result, the physical preconditions and effects required for snippet-based planning often remain implicit and must be elicited and modeled explicitly.

The planning model builds from multiple sources. Diagnostic hypotheses, procedural knowledge, subsystem structure, and execution constraints are distributed across legacy procedures, subject matter expertise, and engineering models, often at different levels of abstraction and created for different purposes. Converting these materials into a formal model is therefore not a straightforward translation step. Prior work in planning knowledge engineering has repeatedly identified model construction, elicitation, refinement, and maintenance as central challenges in real applications (Vaquero et al. 2013; Lindsay et al. 2020).

The difficulty of turning procedural knowledge from manuals and subject matter experts (SMEs) into formal planning models is a recognized knowledge engineering bottleneck (Bonasso and Boddy 2010; Lindsay et al. 2017). Within the spaceflight crew-procedure domain, significant foundational work includes the Procedure Representation Language (PRL) (Kortenkamp et al. 2008) and efforts to elicit planning constraints from SMEs to enhance PRL (Bonasso, Boddy, and Kortenkamp 2009; Bonasso and Boddy 2010). While our approach focuses on automated procedure synthesis, case-based planning has a long history of reusing and adapting prior plans for new tasks (Spalazzi

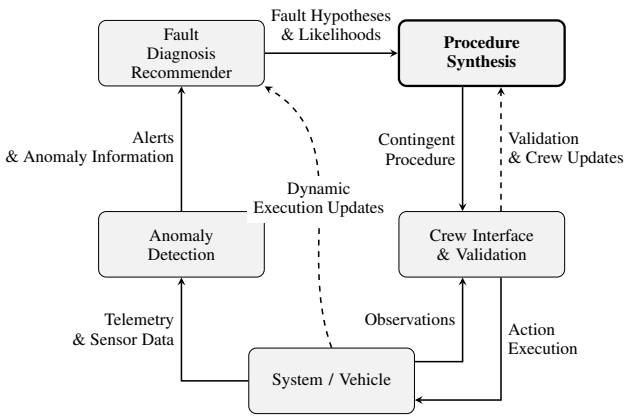


Figure 1: Operational context for anomaly response in EIO. Telemetry deviations trigger anomaly detection and fault diagnosis, which in turn drive contingent procedure synthesis. Dashed feedback paths indicate feedback mechanisms that are part of the intended anomaly response architecture but outside the current implementation.

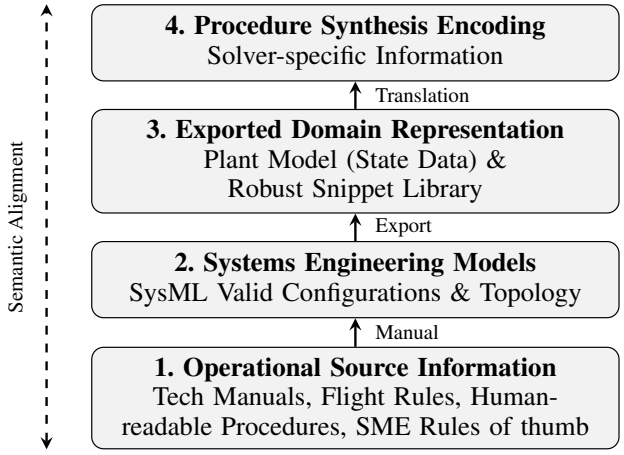


Figure 2: Modeling stack for procedure synthesis. Operational source information is incorporated into SysML models that capture valid system configurations, topology, and snippet structure. A SysML export produces an intermediate domain representation containing plant model state data and the snippet library. Solver-specific translators then produce planner encodings. Semantic alignment must be maintained across the full stack.

2001). Work on learning action models from plan traces faces a related ambiguity because observed action sequences provide evidence about preconditions and effects, but the modeler or learner must still distinguish causal structure from regularities that merely reflect how the examples happened to be executed (Yang, Wu, and Jiang 2007; Zhuo and Kambhampati 2013). Building on these foundations, we argue that human-aware procedure generation and semantic alignment with source materials deserve renewed attention.

We examine three recurring knowledge engineering issues that arose in our work on anomaly response procedure synthesis. The first is how to separate observational activities and state-changing snippets without misrepresenting operational behavior. The second is how to abstract reusable snippets in terms of system state, including their preconditions and effects, rather than inherited procedure sequence. The third is how to choose simplifying assumptions that support tractable reasoning while preserving the structure of real operational procedures. Taken together, these issues motivate a layered modeling approach. Modelers incorporate operational source information into system models (in SysML) that capture valid configurations, topology, and snippet structure. An export process produces an intermediate domain representation containing plant model state data and snippet-library information, which solver-specific translators use to produce planner encodings. Figure 2 summarizes this stack.

## Modeling Action and Observation

A central modeling challenge in anomaly response procedure synthesis is deciding how to represent snippets that primarily gather information versus snippets that primarily change the physical system. In the formal domain-abstraction layer of our modeling stack, we encode this distinction explicitly in the plant model. In this work, the plant model is the formal system model that captures the physical states and state transitions relevant to anomaly response, together with the observation outcomes needed for diagnosis, before translation into solver-specific encodings.

In our current formulation, separating physical state change and observation shapes the contingent procedures that the planner produces. Some snippets primarily change the physical state of the system, while others primarily reduce uncertainty by producing information relevant to diagnosis. This separation is related to the familiar distinction in planning with sensing actions between changes in the world and changes in the agent's knowledge (Son and Baral 2001). Fully observable nondeterministic planning formalisms represent branching over possible action outcomes in the world (Muise, McIlraith, and Beck 2012). Here, however, we use a simplified abstraction that represents information-gathering and physical state change as separate modeling roles.

Our current approach constructs diagnostic contingencies offline. The outcomes of observations appear explicitly as branches in a contingent procedure, rather than as inputs to an online replanning loop. This branching structure is standard in contingent planning under partial observability, where later behavior depends on observations obtained during execution (Hoffmann and Brafman 2005; Maliah et al. 2014). In diagnosis-oriented settings, some snippets serve this information-gathering role more strongly than others, even when they also involve physical manipulation.

Human-authored procedures can make this separation hard to maintain. Procedures sometimes combine physical manipulation, observation, and immediate reaction into a single operational unit. For example, a hand-written leak-check step may instruct a crew member to open a valve briefly, inspect for liquid, and immediately close the valve

if liquid is found. Operationally, this is one coherent instruction that exposes the condition, checks the evidence, and restores the system if the evidence indicates a hazard. In a contingent representation, however, the same behavior may need to be split into separate model elements in which a physical action opens the valve, an observation distinguishes whether liquid is present, and a branch-specific response closes the valve only on the hazardous outcome.

This difficulty arises because operational observations are often active tests rather than passive readings. A crew member may need to change a valve position, power a component, command a mode transition, or connect test equipment before the relevant evidence becomes observable. The resulting snippet is therefore neither purely observational nor purely physical. Instead, it changes the system in order to expose diagnostic information, and the observation may determine whether the system must immediately be restored to a safe configuration.

If a modeler attempts to consolidate the physical action and the diagnostic check into a single step, the immediate conditional response remains difficult to encode in our abstraction. In a contingent tree representation, the response must appear only on the branch where the relevant observation occurs. Trajectory constraints in PDDL3 (Gerevini et al. 2009) or Linear Temporal Logic (LTL) search-control knowledge (Bacchus and Kabanza 2000) can express temporal requirements over linear plans or execution traces. However, requirements over all branches of a contingent procedure are different: they concern a policy or plan tree rather than a single realized sequence. Branching-time logics such as Computation Tree Logic (CTL) provide one formal way to reason about such branching structures (Clarke and Emerson 1981; Daniele, Traverso, and Vardi 1999), but asking modelers to encode fluid operational reactions as branching temporal formulas would shift the knowledge engineering burden rather than eliminate it. Consequently, this boundary can shape the model and generated procedure as much as the planning solver itself.

## Modeling Conditions: Sequence vs. State

A major challenge in building reusable snippets is defining the conditions under which they apply and the conditions they establish. This problem arises early in our modeling stack, when modelers translate legacy procedures, operational source information, and subject matter expertise into reusable snippet models. Legacy procedures often leave both prerequisites and resulting state changes implicit, so modelers must recover them with help from subject matter experts (SMEs). More broadly, the planning community has long recognized model acquisition and maintenance as major bottlenecks, whether the starting point is natural language descriptions, example plans, or evolving operational practice (Lindsay et al. 2017; Yang, Wu, and Jiang 2007; Cresswell, McCluskey, and West 2009; Bryce, Benton, and Boldt 2016).

When SMEs define snippet conditions, they often begin with procedure-dependent markers. For example, an SME may describe the precondition for Snippet $N{+}1$ as `(completed-snippet-N)`, indicating that Snippet $N$ must be completed before Snippet $N{+}1$ is executed. That description may fit an original human-created procedure, but it does not generalize well to synthesis. A planner that reasons over causation needs the physical conditions established by an earlier snippet, rather than merely the fact that the previous snippet occurred. In practice, however, SMEs often describe procedures in terms of ordered steps rather than in terms of the physical conditions those steps establish.

This "non-robust" modeling practice can severely limit the planning system because it blinds the solver to alternative paths. Consider a simple spaceflight-inspired example: a diagnostic snippet requires a fluid line to be depressurized. In the physical system, the crew may be able to achieve this condition through more than one operational path, such as opening a bleed valve or commanding an alternate vent path. Both methods can establish the same abstract condition, `(depressurized fluid-line)`, even though they differ in the lower-level configuration changes they make.

However, if an SME is modeling a legacy procedure that nominally uses the bleed valve, they may define the diagnostic step's precondition using an action-specific execution flag, such as `(executed-open-bleed-valve)`. By doing so, they hardcode the original manual's rigid sequence directly into the model. Consequently, if the bleed valve is unavailable, the planner cannot synthesize a valid procedure using the alternate vent path, because the precondition demands the historical execution of a specific action rather than the achievement of a physical state.

It is important to note that an SME's reliance on sequence often has a practical basis. In spaceflight operations, procedures encode operational knowledge derived from SME experience, testing, and training. Procedure order may also reflect flight rules and other operational constraints intended to protect crew safety and spacecraft systems (Kortenkamp et al. 2008; Barreiro et al. 2010). When the original SME procedure authors specify an execution order, that order may reflect information that has not yet been represented as explicit planner-visible conditions, resources, temporal constraints, or trajectory constraints. Meaningful operational procedure knowledge must be converted into planner-usable information about temporal relationships between tasks, resources, preconditions, effects, constraints, and decompositions (Bonasso and Boddy 2010). An automated planner that sees only final-state goals will not preserve such path-dependent requirements unless the model represents them explicitly. If the translation omits these constraints, the resulting model can lose requirements that the original procedure sequence preserved implicitly. The friction arises because the sequence that is obvious to operators must be translated into declarative structure before a solver can safely reason about alternative orderings.

## Aligning Formal Models with Operational Reality

Automated procedure synthesis operates through a formal model that must support tractable reasoning. The central

challenge lies in formalizing the plant model and ensuring that the systems engineering content corresponds to it in a way that supports procedure synthesis. The recurring modeling issue is that operational procedures often carry structure implicitly, while automated planners require that structure to be represented explicitly. Procedure authors may rely on step order, location grouping, tool staging, crew conventions, or conservative diagnostic practice to make a procedure workable. A synthesis system can preserve those properties only when the model exposes them as conditions, costs, preferences, constraints, or other planner-visible structure.

This task requires more than a direct translation from one representation to another. Beyond typical modeling concerns, such as abstraction granularity, modelers must determine which actions procedures contain, what requirements those actions must satisfy, how those actions may interact with the system, and how those interactions should appear in the plant model. These modeling decisions give the planner a reasonable computational structure, but they also introduce assumptions that make the synthesis problem tractable. Those assumptions shape the procedures the planner can produce.

One important example is the single-fault assumption. In our architecture, the diagnostic recommender and the procedure planner both treat a given anomaly as arising from one root cause. This shared assumption gives the interface between the two components a clear structure. The recommender can provide a ranked list of candidate faults with associated likelihoods, and the planner can use those likelihoods to optimize the expected cost of isolating a fault. At the same time, operational procedures may need to preserve competing diagnostic hypotheses when evidence remains ambiguous, when faults can mask one another, or when an initial hypothesis captures only part of the situation. Multiple-fault diagnosis has long been recognized as a distinct and difficult problem, especially in dynamic systems where interacting faults can manifest in different ways over time (Daigle et al. 2016; de Kleer and Williams 1987).

A similar issue arises in the objective function. Many cost-based planning encodings assign costs to individual actions and optimize an additive cost objective. This approach supports efficient optimization and gives the planner a clear numerical criterion, but it can miss execution burdens that arise from the structure of the plan as a whole. Human-authored procedures often organize work around location, tool usage, system configuration, or crew workflow so that execution proceeds smoothly. For example, a synthesized diagnostic procedure might first ask a crew member to inspect a valve panel in one module, then return to a workstation to review telemetry, then go back to the same panel to take a manual measurement, then return again to the workstation to run a command. Each individual action may have a modest cost in the planner's model, but the resulting procedure forces repeated changes in location, tools, interfaces, and cognitive task sets. A human procedure author might instead group the panel-side checks together, then group the workstation activities together, even if the individual action costs are unchanged. Human-aware planning has likewise emphasized that plan quality depends on more than the agent's internal objective, since people also care about how a plan aligns with their expectations and constraints (Chakraborti, Sreedharan, and Kambhampati 2019). This perspective also fits the broader cognitive literature on task switching, which shows measurable performance costs when people repeatedly shift tasks or task sets (Monsell 2003).

These issues illustrate the broader knowledge engineering problem. A human-authored procedure may already encode operational structure through its organization, for example by grouping activities by location, minimizing tool changes, delaying commitment to a diagnosis, or preserving a safe configuration while evidence is gathered. A planner cannot infer these intentions from the sequence alone. To reason safely about alternative procedures, the model must represent the underlying structure explicitly, whether as state variables, resource constraints, temporal constraints, trajectory preferences, cost terms, or decomposition choices (Gerevini et al. 2009; Bacchus and Kabanza 2000). Modelers may therefore need to augment the representation with auxiliary state variables or reshape the encoding so that synthesized procedures better reflect human burden and operational flow. The challenge is deciding which implicit operational commitments must become part of the formal model.

## Conclusion

Earth-independent anomaly response offers a demanding setting for automated procedure synthesis. Our experience suggests that progress in this area depends as much on knowledge engineering as on planning and optimization. The central challenge lies in transforming operational knowledge from procedures, manuals, SMEs, and systems models into formal abstractions a procedure planner can use.

We examined three recurring modeling issues that arise in that transformation: representing information-gathering and state-changing snippets without losing tightly coupled operational behavior, defining snippet conditions in terms of system state rather than inherited procedure sequence, and choosing simplifying assumptions that support tractable reasoning while preserving operational structure. Each issue requires abstraction decisions that strongly influence the procedures the system can generate.

These modeling challenges point toward a layered modeling approach. In our work, modelers incorporate operational source information into system models that capture valid configurations, topology, and snippet structure. An export process then produces an intermediate domain representation containing plant model state data and snippet information, which solver-specific translators use to produce planner encodings. This structure emphasizes the need to maintain semantic alignment across layers, since the quality of the synthesized procedure depends directly on the quality of the intermediate representation. Robust anomaly response procedure synthesis will depend on better ways to construct, preserve, and evolve the models on which planners rely.

## References


Bacchus, F.; and Kabanza, F. 2000. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artificial Intelligence*, 116(1–2): 123–191.

Barreiro, J.; Chachere, J.; Frank, J.; Bertels, C.; and Crocker, A. 2010. Constraint and Flight Rule Management for Space Mission Operations. In *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*. Toronto, Canada.

Bonasso, R. P.; and Boddy, M. S. 2010. Eliciting Planning Information from Subject Matter Experts. In *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*. Toronto, Canada.

Bonasso, R. P.; Boddy, M. S.; and Kortenkamp, D. 2009. Enhancing NASA's Procedure Representation Language to Support Planning Operations. In *Proceedings of the International Workshop on Planning and Scheduling for Space (IWPSS)*. Pasadena, CA.

Bryce, D.; Benton, J.; and Boldt, M. W. 2016. Maintaining Evolving Domain Models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 3053–3059.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2019. Balancing Explicability and Explanations for Human-Aware Planning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 1335–1343.

Clarke, E. M.; and Emerson, E. A. 1981. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. In Kozen, D., ed., *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, 52–71. Springer.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2009. Acquisition of Object-Centred Domain Models from Planning Examples. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, 338–341. AAAI Press.

Daigle, M. J.; Bregon, A.; Koutsoukos, X.; Biswas, G.; and Pulido, B. 2016. A Qualitative Event-Based Approach to Multiple Fault Diagnosis in Continuous Systems Using Structural Model Decomposition. *Engineering Applications of Artificial Intelligence*, 53: 190–206.

Daniele, M.; Traverso, P.; and Vardi, M. Y. 1999. Strong Cyclic Planning Revisited. In *Proceedings of the Fifth European Conference on Planning (ECP'99)*, volume 1809 of *Lecture Notes in Computer Science*, 35–48. Springer.

de Kleer, J.; and Williams, B. C. 1987. Diagnosing Multiple Faults. *Artificial Intelligence*, 32(1): 97–130.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6): 619–668.

Hoffmann, J.; and Brafman, R. I. 2005. Contingent Planning via Heuristic Forward Search with Implicit Belief States. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005)*, 71–80.

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)*. Pasadena, CA.

Lindsay, A.; Franco, S.; Reba, R.; and McCluskey, T. L. 2020. Refining Process Descriptions from Execution Data in Hybrid Planning Domain Models. In Beck, J. C.; Buffet, O.; Hoffmann, J.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling*, volume 30, 469–477. AAAI Press.

Lindsay, A.; Read, J.; Ferreira, J. F.; Hayton, T.; Porteous, J.; and Gregory, P. 2017. Framer: Planning Models from Natural Language Action Descriptions. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, volume 27, 434–442. AAAI Press.

Maddox, I. D.; Vera, A.; and Martinez, A. 2024. Earth Independent Operations Development for NASA's Mars Campaign Office. In *ASCEND 2024*. American Institute of Aeronautics and Astronautics.

Maliah, S.; Brafman, R. I.; Karpas, E.; and Shani, G. 2014. Partially Observable Online Contingent Planning Using Landmark Heuristics. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, volume 24, 163–171. AAAI Press.

Monsell, S. 2003. Task Switching. *Trends in Cognitive Sciences*, 7(3): 134–140.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling*, volume 22, 172–180. AAAI Press.

Parisi, M.; McTigue, K.; Wu, S.-C.; Karasinski, J.; Panontin, T.; Landon, L.; and Vera, A. 2024. The Impact of Delayed Communications on NASA's Human-Systems Operations: Preliminary Results of a Systematic Review. In *Advances in Human Factors of Transportation*, volume 148, 49–57.

Son, T. C.; and Baral, C. 2001. Formalizing Sensing Actions: A Transition Function Based Approach. *Artificial Intelligence*, 125(1-2): 19–91.

Spalazzi, L. 2001. A Survey on Case-Based Planning. *Artificial Intelligence Review*, 16(1): 3–36.

Vaquero, T. S.; Silva, J. R.; Tonidandel, F.; and Beck, J. C. 2013. itSIMPLE: Towards an Integrated Design System for Real Planning Applications. *The Knowledge Engineering Review*, 28(2): 215–230.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning Action Models from Plan Examples Using Weighted MAX-SAT. *Artificial Intelligence*, 171(2–3): 107–143.

Zhuo, H. H.; and Kambhampati, S. 2013. Action-Model Acquisition from Noisy Plan Traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2444–2450. AAAI Press.