

Testing Autonomous Systems by Analyzing Goal Interactions

A New Proposal for 632-07

Task Leaders

Ben Smith, JPL, benjamin.d.smith@jpl.nasa.gov, 818-393-5371 (AI Group)
 Martin Feather, JPL, martin.feather@jpl.nasa.gov, 818-354-1194 (Software Quality Assurance)

Product Description

Autonomous systems achieve specified goals in a manner that satisfies encoded con-straints on their behavior. Thorough validation is a prerequisite to fielding such systems. Our experience with validating the DS1 Remote Agent Experiment (RAX) indicates that two kinds of flaws account for most of the validation effort: unexpected goal interactions, and missing or incorrect constraints. We propose to investigate and develop automated methods for analyzing the constraints in order to automatically generate a tractable set of test cases that will expose those flaws. We will initially apply these methods to AI planning systems, and then to the Mission Data System (MDS) control architecture, a multi-mission flight software architecture being developed by JPL that achieves commanded goals via interacting goal-achieving autonomous software modules.

This is a **new, pull** task. It is expected to advance the state of the art in validation of autonomous systems as summarized in the following table.

	State of the Art	FY 2000	FY 2001	FY 2002
test selection	ad-hoc	governed by coverage metrics (principled)		
test generation	manual	automated		
missing constraints (e.g., safety) tested	none	b/t all goal pairs, manual pruning	between all goal pairs with semi-auto pruning of irrelevant cases	
interactions tested	ad-hoc pairs	principled 2,3-wise	principled, k-wise	
applied / evaluated wrt	AI planners	AI planners	MDS prototypes	MDS (infused)

Benefits

Existing test-case selection methods are *ad-hoc* and provide correspondingly spotty coverage with no way to determine whether the system has been adequately tested. This technology will enable *automatic, principled* generation of test cases that will provide high confidence in the tested system. By selecting tests in a *principled* manner governed by formal coverage metrics we believe we can generate a reasonable number of test cases that will expose a high proportion of the flaws attributable to goal interactions and missing/incorrect constraints. A formal coverage metric also provides an objective basis for evaluating test adequacy. Generating test cases in an *automated* fashion reduces the demand on testers and allows the test suite to be quickly revised when the constraints change between software releases.

Technical Approach

Verification and Validation (V&V) is an active area of both research and practice, but very little of this work has focused on autonomous systems. As autonomy becomes increasingly important for NASA, so does verification and validation of these systems.

Autonomous systems, such as AI planners and the MDS control architecture, achieve commanded goals according to constraints on their behavior specified in a declarative *model* comprised of *constraints*. For

example, a “take image” goal may require 10 watts of power and a specified spacecraft attitude. In flight qualifying the DS1 planner, we observed two major classes of flaws: failure to properly resolve conflicts among constraints, and missing or incorrect constraints within the model. The first is a verification issue specific to goal-achieving systems; the second is consistent with a recent study of software systems (Kelly, 1999) that attributes 66% of major flaws to missing requirements (constraints).

We will investigate two different constraint analysis approaches for exposing these flaws. The first approach generates test cases with good coverage of goal interactions since these interactions expose conflict resolution flaws and some incorrect/missing constraint flaws. The second approach hypothesizes missing constraints and generates test cases that would expose their absence.

The first approach, of verifying a system against tests selected according to some coverage metric, is used commonly by industry to verify safety-critical systems (e.g., code coverage [RST]). However, to our knowledge, there are no coverage metrics specifically suited to AI planners or autonomous systems. The current best strategy is to select an *ad-hoc* set of cases based on the tester’s intuition, but this may miss critical cases and provides no objective basis for assessing whether the system has been adequately tested. We tested all pairs of an ad-hoc selection of goal instances on RAX, but this method missed at least 9% of the goal-interaction flaws (Nayak *et al.*, 99). Principled selection and attention to higher-order interactions would have caught those flaws.

The second approach of hypothesizing potentially missing constraints between pairs of goals or plan elements is similar to techniques for determining the consistency and completeness of embedded systems whose behavior can be expressed as a (finite) set of transition rules (Heimdahl & Leveson, 1996; Heitmeyer *et al.*, 1995), but again, these techniques do not readily extend to AI planners, whose state space is much larger.

A complementary validation approach is proving that the design enforces desirable properties (e.g., Holzman 97). Penix *et al.* (1998) use this approach to prove properties of a plan model, such as whether goals are unachievable or only achievable from certain initial states. These methods are good for validating formally specifiable properties, but cannot expose flaws such as missing or incorrect constraints. Test-based validation (such as ours) is good at detecting these flaws but is not well suited to proving formal properties. Both of these efforts are important and complementary aspects of validation. We have had fruitful discussions with the Ames Automated Software Engineering Group, and will continue to discuss and coordinate our efforts.

We will implement algorithms for each of these approaches (goal-interaction and constraint omission) and evaluate their coverage, tractability, and effectiveness at exposing flaws. The initial evaluations will be with respect to the Remote Agent Experiment planner (Muscettola *et al.*, 1996), for which we have an extensive corpus of software versions with known bugs, and on current Aspen (Fukunaga *et al.*, 1997) planning applications. Aspen is an MDS-compatible planning system developed by the JPL AI group that supports a number of planning applications for JPL missions. Later applications and evaluations will be with respect to relevant components of the MDS control architecture, which we expect by early FY 01.

Goal Interactions

This analysis method is targeted at exposing combinations of goals whose constraints interact, leading to conflicts that the system cannot resolve even though a solution exists. Since testing all combinations is intractable, the idea is to analyze the constraints to determine how the goals interact, and only test goal combinations that yield qualitatively different conflicts. For example, if goals A and B used power, we would test cases where power is oversubscribed by several A goals, by several B goals, and by a combination.

The algorithm will generate a set of test cases by analyzing the constraints in the model to determine what conflicts will result from a set of goals, and to partition goal parameters into value ranges that will result in qualitatively similar conflicts. Constraints conflict if they are mutually unsatisfiable.

This approach extends on prior work on detecting goal interactions in planners for the purposes of avoiding them in order to improve the plan search. These methods are designed for planning systems that do not reason about metric time and aggregate resources (e.g., power). Our approach will extend these methods for goal-achieving systems that utilize these capabilities, which are critical for spacecraft applications and central to the RA planner, Aspen, and MDS. These methods are also limited to pair-wise goal interactions. We hope to extend them to *k*-ary interactions, since this is where the more subtle flaws manifest. STATIC (Etzioni, 93) generates a problem solving graph from the constraints and identifies search control rules for

avoiding goal interactions. Alpine (Knoblock 94) identifies interactions to find non-interacting sub-problems, and universal plans (Schoppers 87) derive reactive control rules from pair-wise goal interactions. The number of test cases can be controlled by reducing the scope of the coverage. We envision that missions might take the following approach. First generate tests with partial coverage of all the goals. This can be done by limiting the number of goal interactions considered (e.g., only consider pairs and triples) or only considering strongly interacting goals and constraints. This provides reasonable confidence that the system can handle any set of goals. The next step is to augment these general tests with ones that provide full coverage of a narrow range of goals that the mission expects to perform. For example, this approach could validate that a fixed set of goals will not have unexpected interactions with fault-protection goals. This provides very high confidence in a few specific goals.

Missing or Incorrect Constraints

A major source of errors is missing or incorrect constraints. We assume that an expert can recognize missing or incorrect constraints if shown a test case in which that flaw is exercised. The objective is to identify the set of potentially missing and incorrect constraints, and generate a set of test cases that would expose those flaws.

For example, in DS1 the spacecraft must not take images while the ion propulsion engine is on. Had this constraint been omitted from the model, the algorithm would hypothesize its existence and generate a test case in which the engine is thrusting while taking a picture. The expert would then indicate that this behavior is incorrect, and add an appropriate constraint to the model.

We propose the following approach for identifying missing constraints:

1. Hypothesize potentially missing constraints. We assume that there are only a few kinds of constraints and that they are binary. This is true in many planning systems, including the RAX planner and Aspen. For every pair of plan elements between which a constraint could exist, hypothesize a constraint of that class. There are $O(n^2)$ potentially missing binary constraints for n activity types. This number can be significantly reduced by an expert's directives to not consider certain combinations.
2. Present the hypothesized constraints to a human expert for further pruning.
3. Construct test cases that will expose the missing constraints. A single test case can expose several constraints. The expert reviews the cases for flaws.

Incorrect constraints are exposed in any output behavior in which the system enforces the constraint. The effects of the constraint manifest in the output behavior, and we assume an expert can recognize these flaws. For example, the expert knows activity A should use 10 watts, but if it only uses 5 watts in the output (because of an incorrect constraint) the expert will recognize that as a flaw. A small handful of cases can exercise all of the constraints. For the DS1 planner, three cases was sufficient.

Status and Milestones

This is a new task. However, for the past two years the proposers have been developing tools and methods for testing autonomous systems, and applying them operationally. Feather and Smith (1998) developed an automated plan-checker for verifying the correctness of RAX plans with respect to the model and an *ad-hoc* list of requirements. Smith lead the validation effort for the RAX planner and the final system-level validation of RAX on the DS1 testbed (Smith, 1999; Nayak *et al.*, 1999). He is a senior member of the JPL AI Group, where he pursues research in autonomous systems and planning systems. Martin Feather pursues active research in the area of software specification and validation. The automated plan-checker is based on earlier work (Feather, 1998) on "lightweight" formal validation methods.

Milestones

FY 00 Prototype basic test-case generation algorithms.

Evaluate effectiveness on RAX & Aspen planner models.

FY 01 Advanced generation algorithms for improved coverage & fewer cases.

Develop initial methods for identifying missing constraints.

Initial application to selected goal-achieving components of MDS; evaluate

FY 02 Infuse into MDS test process. Mature application to MDS components; evaluate.

In FY 00 we will develop and prototype algorithms for analyzing the constraints and generating test cases. We will evaluate their effectiveness and coverage with respect to the RAX and Aspen planners, for which we have a large set of models and known bugs.

In FY 01 we will improve the analysis and test-case generation algorithms in order to provide increased coverage and smaller test suites. We will also extend our algorithms to prototypes of selected goal-achieving components of the MDS control architecture, which are expected by early FY 01. The most likely candidate is the long-term estimator, which is a planner-like system for resolving long-term interactions among goals.

In FY 02 we will apply our algorithms to mature MDS components, and deploy the algorithms in tool usable by MDS test engineers (TRL 6). We will guide its infusion into the MDS test process, and evaluate its effectiveness in the context of operational testing by MDS test engineers.

Customer Relevance

This technology supports NASA Enterprise goals for fielding autonomous spacecraft and other autonomous systems in that validation is a pre-requisite for operations:

- **Space Science**, *Autonomous Spacecraft Systems* [Sun-Earth Connection (S4)]
- **Earth Science**, *On-board Spacecraft Autonomy* [Information Synthesis (E4)]
- **HEDS**, *Intelligent Agents for Spacecraft Systems Control* [Operations (H1)]

This technology also supports MDS needs for validation technologies. MDS is expected to be an early user and has agreed to provide co-funding. The following people have endorsed this technology. Please see the attached letters of support.

John Carraway	Mission Operations Systems Manager, Outer Planets/Solar Probe
Kirk Reinholtz	MDS Test Tools Lead
Tom Starbird	MDS Control Architecture Lead

Technical References

- [Etzioni, 1993] "Acquiring Search-Control Knowledge via Static Analysis," *Artificial Intelligence*, **62**(2),
- [Feather, 1998] "Rapid Application of Lightweight Formal Methods for Consistency Analyses," *IEEE Transactions on Software Engineering*, **24**(11), November 1998.
- [Feather and Smith, 1998] "Verification and Validation of a Spacecraft's Autonomous Planner through Extended Automation," Proc. 23rd Software Engineering Workshop, NASA Goddard, 1998.
- [Fukunaga *et al.*, 97]"Toward an application framework for automated planning and scheduling," ISAIRAS
- [Heimdahl & Leveson, 1996] "Completeness and Consistency Analysis of State-Based Requirements," *IEEE Trans. on Software Eng.* 22(6) pp. 363-377, May 1996.
- [Heitmeyer et al 1995] "Consistency Checking of SCR-Style Requirements Specifications," IEEE Int. Symposium on Requirements Engineering., York, England, March 1995 pp. 56-63
- [Holzmann, 1997] "The Model Checker SPIN," *IEEE Trans. on Software Eng.* **23**(5) , pp. 279-295
- [Kelly, 1999] "Practical Application of Software Formal Methods," NASA Center Directors Meeting, NASA IV&V Facility, June 23-25, 1999.
- [Knoblock, 1994] "Automatically generating abstractions for planning," *Artificial Intelligence*, **68**(2).
- [Nayak, *et al.*, 1999] "Validating the DS1 Remote Agent," ISAIRAS. 1999.
- [Patel and Reinholtz, 1998] "Testing Autonomous Systems for Deep Space Exploration," IEEE Aerospace Conference, 1998. Vol 2, pp 283-288.
- [Penix *et al.*, 1998] "Using model checking to validate AI planner domain models," Proceedings of the 23rd Software Engineering Workshop, NASA Goddard, 1998.
- [Reinholtz & Patel, 98] "Testing autonomous systems for deep-space exploration," IEEE Aerospace Conf.
- [Schoppers, 1987] "Universal plans for reactive robots in unpredictable environments," *Proc. IJCAI*, 1987.
- [Smith, *et al.*, 1999] "Verification and Validation of the Remote Agent," IEEE Aerospace Conf., 1999.

June 22, 1999

To: Cross-Enterprise Technology Development Program

From: John Carraway, Outer Planets/Solar Probe Project

Subject: **Endorsement of the “Testing Autonomous Systems” proposal**

I am writing this letter to express my support of the “Testing Autonomous Systems by Analyzing Goal Interactions” task proposed by Dr. Ben Smith and Dr. Martin Feather at JPL.

The Smith and Feather proposal will enable the three Outer Planets/Solar Probe Project (OPSP) missions—Europa Orbiter, Pluto-Kuiper Express, and Solar Probe—to validate that the new MDS software correctly achieves commanded goals and that there are no unanticipated interactions. This will allow aggressive use of goal-based commanding by the Europa mission Jupiter satellite fly-by “bonus” science and for Europa orbit extended mission. All three missions will exploit goal based commanding during cruise to minimize ops costs. Existing methods for testing time-ordered sequences are inadequate for validating this aggressive level of goal-based commanding.

The validation technology will be particularly useful for identifying interactions between commanded goals and fault protection goals. Unintended interaction between these goal types could endanger the spacecraft or improperly execute the commanded goals. The successful of the Europa Orbiter prime mission will rely heavily on the ability to resume execution of commanded goals (observations) in the event that fault protection goals get triggered. This proposed technology will enable OPSP missions to validate interactions between the fault protection and commanded goals in a principled manner that is not possible with existing methods. Thorough, principled validation of the fault protection interactions is critical to the success of these missions.

Finally, this technology will allow OPSP missions to thoroughly exercise the goal-achievement capabilities of future MDS deliveries by automatically analyzing the delivery and generating tests that exercise the goal interactions. Existing methods provide only spotty coverage, and the tests must be upgraded manually for each release.

This validation technology will enable OPSP missions to make confident, aggressive use of MDS’s goal-based commanding capability. The proposed technologies are strongly aligned with the goals of the OPSP missions. I strongly support funding for this technology.

John Carraway
Mission Operations Systems Manager,
Outer Planets/Solar Probe Project

May 12, 1999

To: CETDP Program

From: Kirk Reinholtz

Subject: Support for the “Test Generation by Analyzing Goal Interactions” task

This letter is in support of the “Test Generation by Analyzing Goal Interactions” task proposed by Dr. Ben Smith and Dr. Martin Feather at JPL.

Developing testing methods and technologies for autonomous goal-achieving systems is of prime importance to MDS. The MDS commanding architecture consists of several “goal achieving modules” or GAMS that carry out the goals of the operations team according to behaviors and constraints specified in declarative models. Testing a single GAM is fairly straightforward, but the interactions among the GAMS can be numerous and complex. Testing all of the interactions is intractable and we do not currently have test methods for systematically exercising those interactions.

The Smith and Feather proposal promises to create small test-suites that exercise the key goal interactions of planning systems by analyzing the constraints in the declarative plan model. This technology is applicable to testing goal interactions among GAMS, and I believe it will provide a powerful means for validating MDS. We have identified interactions as a critical area for focusing testing effort [Reinholtz & Patel, 98], and this technology is in line with that assessment. Furthermore, the ability to generate tractably sized test suites automatically will free up testing resources that can be put to use in validating other aspects of MDS. Since the test cases are generated systematically they will enable us to better assess the coverage of the test cases and the confidence they engender in the flight software.

The proposed technologies are strongly aligned with the goals of the MDS testing program. I strongly support funding for this task.

Sincerely,

Kirk Reinholtz
Test Tools Lead
Mission Data System (MDS)

June 24, 1999

To: CETDP Review Board
From: T. Starbird, Lead, Execution & Planning Domain, Mission Data System Project, JPL
Re: Testing Autonomous Systems by Analyzing Goal Interactions

This letter expresses my support for the proposal on validation technology by Ben Smith and Martin Feather entitled "Testing Autonomous Systems by Analyzing Goal Interactions."

This technology addresses a critical need for validating the MDS control architecture. The MDS control architecture consists of several software modules that achieve commanded goals within encoded constraints on how those goals can be achieved. The modules interact with each other, which can cause them to achieve the goals in different ways. A primary validation objective is to ensure that the encoded constraints lead to the desired behavior regardless of how they interact.

The best validation method currently available is to select test cases based on the intuition of the test engineer. This approach is time-consuming and may overlook critical interactions. These omissions might not be exposed until flight, when an untested interaction causes a GAM to command the spacecraft in unintended ways, or even endanger the spacecraft. MDS client missions are unlikely to make full use of the goal-based commanding capabilities without better validation guarantees.

The proposed validation technology will enable the automatic generation of test cases that cover the most critical interactions. It also tries to minimize the number of test cases. This will provide better coverage guarantees than existing methods, and will require fewer testing resources since cases are smaller and generated automatically.

We need to explore new ideas for validating the MDS control architecture and instances of its use on flight projects. This proposal offers a promising approach that should be pursued. I therefore support the "Testing Autonomous Systems by Analyzing Goal Interactions." proposal.