The Resilient Spacecraft Executive: An Architecture for Risk-Aware Operations in Uncertain Environments

Catharine L. R. McGhan* and Richard M. Murray†
California Institute of Technology, Pasadena, CA, 91125, USA.

Tiago Vaquero‡ and Brian C. Williams§
Massachusetts Institute of Technology, Cambridge, MA, 02139, USA.

Michel D. Ingham¶ , Masahiro Ono‖ , Tara Estlin**,
Ravi Lanka††, Oktay Arslan‡‡,
and Maged E. Elaasar***
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA.

In this paper we discuss the latest results from the Resilient Space Systems project, a joint effort between Caltech, MIT, NASA Jet Propulsion Laboratory (JPL), and the Woods Hole Oceanographic Institution (WHOI). The goal of the project is to define a resilient, risk-aware software architecture for onboard, real-time autonomous operations that can robustly handle uncertainty in spacecraft behavior within hazardous and unconstrained environments, without unnecessarily increasing complexity. The architecture, called the Resilient Spacecraft Executive (RSE), has been designed to support three functions: (1) adapting to component failures to allow graceful degradation, (2) accommodating environments, science observations, and spacecraft capabilities that are not fully known in advance, and (3) making risk-aware decisions without waiting for slow ground-based reactions. In implementation, the bulk of the RSE effort has focused on the parts of the architecture used for goal-directed execution and control, including the deliberative, habitual, and reflexive modules. We specify the capabilities and constraints needed for each module, and discuss how we have extended the current state-of-the-art algorithms so that they can supply the required functionality, such as risk-aware planning in the deliberative module that conforms to mission operator-supplied priorities and constraints. Furthermore, the RSE architecture is modular to enable extension and reconfiguration, as long as the embedded algorithmic components exhibit the required risk-aware behavior in the deliberative module and risk-bounded behavior in the habitual module. To that end, we discuss some feasible, useful RSE configurations and deployments for a Mars rover case and an autonomous underwater vehicle case. We also discuss additional capabilities that the architecture requires to support needed resiliency, such as onboard analysis and learning.

*Postdoctoral scholar, Department of Control and Dynamical Systems, 1200 E. California Blvd., Mail Code 305-16, Member.
†Professor, Department of Control and Dynamical Systems, 1200 E. California Blvd., Mail Code 107-81.
‡Postdoctoral scholar, Department of Aeronautics and Astronautics, 32 Vassar Street, 32-224, Member. Joint appointment with Caltech.
§Professor, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue, 33-330, 32-227, Member.
¶Software Systems Engineer, 4800 Oak Grove Drive, Mail Stop 301-490, Associate Fellow.
‖Robotics Technologist, 4800 Oak Grove Drive, Mail Stop 198-219, Member.
**Group Supervisor, 4800 Oak Grove Drive, Mail Stop 158-242, Member.
††Scientific Applications Software Engineer, 4800 Oak Grove Drive, Mail Stop 158-242, Member.
‡‡Robotics Technologist, 4800 Oak Grove Drive, Mail Stop 198-219, Member.
***Software Systems Engineer, 4800 Oak Grove Drive, Mail Stop 179-206, Member.

I. Introduction

Several distinct trends will influence space exploration missions in the next decade – hazardous conditions, unknown or unpredictable conditions, multi-element missions, and long-duration flight. Destinations are becoming more challenging, science questions more sophisticated and – as mission experience accumulates – the most accessible targets are visited, advancing the knowledge frontier to more difficult, harsh, and inaccessible environments. This leads to new challenges including: hazardous conditions that limit mission lifetime and require graceful degradation of components, such as the caustic, heavy atmosphere of Venus, and the high radiation levels surrounding Europa; navigation hazards on planetary bodies like Mars, including sand traps, sharp rocks and cliffs, and other risky environmental interactions such as digging and drilling; and long-range missions, such as Kuiper belt exploration, that must survive equipment failures over the span of decades.

Some representative mission concepts that would require greater resilience include Venus Lander, Europa Lander, Trojan Tour and Rendezvous, Mars Sample Return, and, even more ambitious, an Interstellar Probe. Such missions would need to be successful without a priori knowledge of the most efficient data collection techniques for optimum science return. Science objectives would have to be revised 'on the fly', calculating the risk-reward tradeoffs onboard, to accommodate new data collection and navigation decisions on short timescales. And, all this needs to be done without increasing system complexity to the point that we can no longer guarantee an acceptable baseline of mission performance.

We have discussed in a previous paper how the required resilience to implement these potential missions cannot be achieved by simply incrementally building on and extrapolating from the current state of the practice; it requires a fundamental paradigm shift in the way we conceptualize, design, implement, validate, operate, and evolve our systems.[1] The novel risk-aware paradigm that we have realized in our Resilient Spacecraft Executive (RSE) architecture is more analogous with human behavior, which can be categorized roughly as a combination of "reflexive" behavior hardwired in the nervous system; "habitual" behaviors that are performed by rote once learned through repetition and muscle-memory training; and finally, "deliberative" reasoning behavior that is used to make decisions, handle novel situations, learn from mistakes, and so forth. In effect, the new paradigm is an attempt to make the spacecraft more 'self-aware' of its own internal state and processes, its environment, its evolving tasks and goals, and the relationship between them, and to include state-of-the-art techniques that allow for onboard processing of the risk-versus-reward tradeoffs necessary for goal accomplishment to be made in real-time as circumstances evolve. The Resilient Spacecraft Executive (RSE) architecture (shown in Figure 1) is meant to autonomously run onboard the spacecraft, making decisions in real-time when remote missions are being conducted that cannot include ground control in the loop from Earth, due to the timescales and delay involved.

Nominally, the deliberative module receives and reasons about mission goals specified by an operator; it computes a mission plan that satisfies those goals within the risk bound. It then communicates a limited timescale plan with constraints to the habitual module. The habitual module elaborates and executes the plan; it handles 'normally-seen' risks and failures while satisfying the risk bounds embodied in the constraints from the deliberative module, and decides on the behavioral mode of the system. Then the habitual module, in turn, outputs the local state trajectories to be executed by the reflexive module's closed-loop controller. Figure 1 shows the interaction between modules. However, various tradeoffs can be made according to the level of abstraction and type of information passed between modules for different scenarios; for instance, contingency plans require additional front-end computation time, but such precomputed policies generally allow for a faster response than real-time replanning when constraints are violated.

In this paper we will describe the results to date of our joint project, a proof-of-concept RSE architecture and implementation that is intended to robustly handle uncertainty in the spacecraft behavior and hazardous and unconstrained environments, without unnecessarily increasing complexity. We also discuss our advancement of state-of-the-art techniques to include uncertainty and the associated risk as part of the planning process, to find a high-level action sequence to achieve mission goals within acceptable risk levels, at the (deliberative) risk-aware planning level. This includes the extension of RSE's planning and reasoning algorithms to include temporal uncertainty, uncertainty in action-completion, and uncertainty in action-outcome. We discuss the risk-bounded algorithms available for trajectory planning at the (habitual) risk-bounded planning level, including extensions to the algorithms that consider uncertainty in position. We also briefly discuss the synthesis of correct-by-construction control policies,[2] for use either as risk-bounded hybrid controllers at the habitual level that satisfy specified safety and performance constraints, or as discrete symbolic risk-aware planners for fast real-time replanning or contingency planning at the deliberative level.[3] Finally, we present

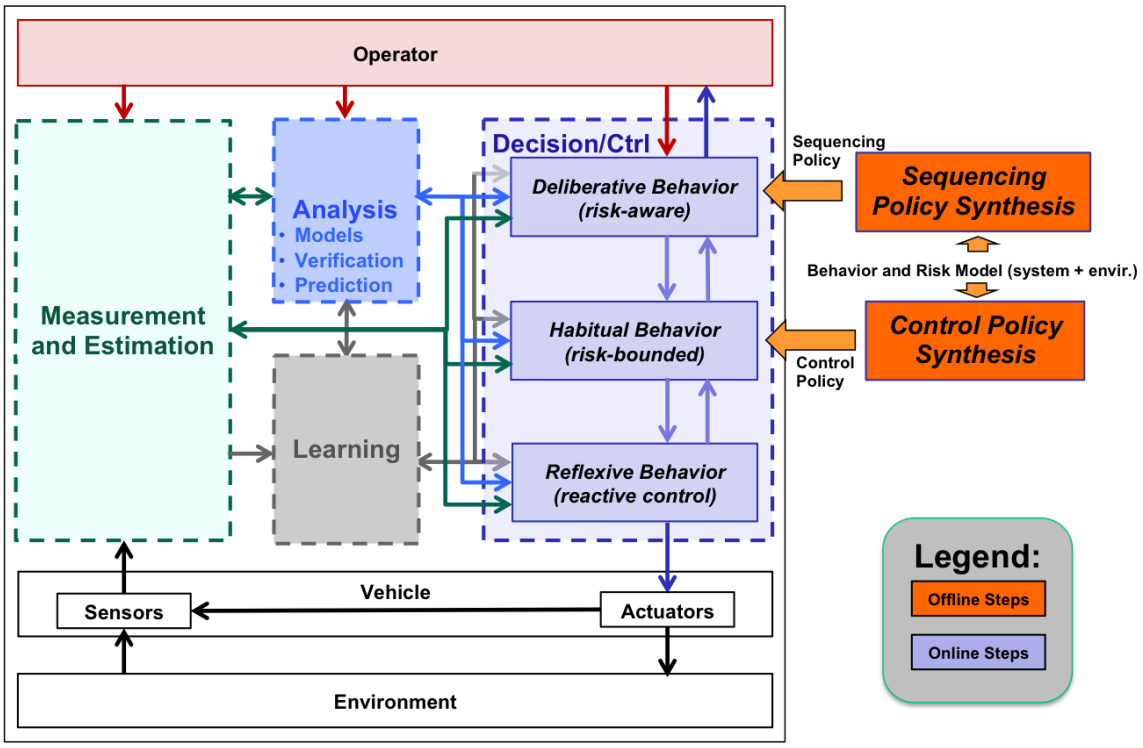


Figure 1: Resilient Spacecraft Executive Architecture.

and discuss results from executing RSE in a Mars rover scenario and an autonomous underwater vehicle (AUV) scenario with measurement uncertainty and injected hardware failures/degradations.

II. RSE Architecture

Spacecraft control technology relies heavily on a relatively large and highly skilled mission operations team that generates detailed time-ordered and event-driven sequences of commands. This approach will not be viable in the future with increasing number of missions and a desire to limit the operations team and Deep Space Network (DSN) costs. Future spaceflight missions will be at large distances and light-time delays from Earth, requiring novel capabilities for astronaut crews and ground operators to manage spacecraft consumables such as power, water, propellant and life support systems to prevent mission failure. In order to maximize the science returns under these conditions, the ability to deal with emergencies and safely explore remote regions are becoming more and more important.

Some examples of limited deployed resilience include the autonomous navigation capability used by the Deep Impact missions impactor spacecraft to assure an accurate impact with a cometary body,[4–6] the Cassini spacecraft's onboard delta-energy calculations to ensure robust Saturn Orbit Insertion, even in the presence of system reboots and failures during this critical sequence,[7,8] and the robust and painstakingly-developed entry, descent and landing sequences[9,10] of recent Mars surface missions, including the Mars Exploration Rovers Spirit and Opportunity, the Phoenix lander, and the Mars Science Laboratory Curiosity rover. These missions deployed focused capabilities that target resilient execution of very specific critical functions. The challenge, therefore, is to generalize from these types of capabilities, to provide resilient autonomous behaviors across the entire system and its mission.

There have been limited examples of truly resilient behavior deployed on-board spacecraft to date. Perhaps the most comprehensive demonstration of sophisticated resilience-enabling autonomy is the Remote Agent eXperiment (RAX), which was flown on the Deep Space One mission.[11–13] The RAX architecture integrated technologies for onboard planning and scheduling, smart execution, and model-based diagnosis and recovery. In this architecture the planner and executive have different representations and strictly operate on different levels of abstraction. Planning is performed in a batch fashion where the planning system only

runs when required. If re-planning is required, the spacecraft is put in a safe mode until a new plan has been generated, which often takes a significant amount of time. Another state-of-the-art system-level autonomy capability is the CASPER (Continuous Activity Scheduling Planning Execution and Replanning) system,[14] which addresses the limitations of batch planning by instead utilizing a continuous planning approach to achieve high-level goals while still respecting resource and temporal constraints. The CASPER system was integrated into the Autonomous Sciencecraft Experiment (ASE), deployed on the Earth Observing One (EO-1) mission. The ASE software uses CASPER's on-board continuous planning capability, in addition to robust task- and goal-based execution, and on-board machine learning and pattern recognition, to radically increase science returns through intelligent downlink selection and autonomous retargeting. Although the emphasis for EO-1 was not on enabling resilience, the ASE/CASPER autonomy capability could certainly be adapted for this purpose.

Key distinctions and innovations in RSE[1] involve:

(i) The architecture's leveraging of sequencing and control policies that are "correct by construction" in both the deliberative and habitual layers. The use of model-based policy synthesis addresses the current challenge of assuring correctness of the system behavior in the face of growing complexity.

(ii) The architecture's emphasis on risk-aware onboard deliberative reasoning, which is critical to managing the unprecedented amount of uncertainty in the environments to be explored in future missions, and managing a space of possible executions that is far too large to be completely covered by design-time control policies. Light-time delays preclude effective ground-based deliberation and planning for many future mission scenarios, and environmental uncertainty introduces significant risk and precludes any guarantees of correct behavior, even though we are employing formally correct-by-construction policies. Endowing our architecture with the ability to assess risk and make decisions based on risk in real-time fills this resilience gap.

(iii) The amenability of the RSE architecture to the use of formal architectural analysis to perform tradeoffs and inform the appropriate allocation of capabilities to the deliberative, habitual and reflexive modules. This will result in systems with flexibility to adapt to their uncertain environments and potential mission changes. This is in contrast to the informal allocation of capabilities to layers in current architectures, which results in comparatively brittle architectures, with properties that may be inappropriately tuned to the mission context (e.g., favoring responsiveness over flexibility, even for mission scenarios without strict time-criticality requirements)[a].

These innovative features of the RSE architecture enable autonomous operation that is resilient enough to support space exploration in uncertain and high-risk environments, and thus enable more ambitious science collection capabilities.

A. Architecture Overview

The aim is to fly through most anomalies and non-critical failures, i.e., eliminate most traditional spacecraft safing occurrences and operator-in-the-loop interactions, limiting them to only those cases where these is high risk that the missions goals cannot be achieved without help from the ground. To this end, the architecture (shown in Figure 1) defines modules for deductive reasoning, pre-validated habitual behaviors and reflexive reactions, analogous to the three types of human behavior. An overview of the architecture is as follows:

- The *deliberative module* of the architecture leverages a Risk-Aware Goal-Directed Executive capability. This module is responsible for managing overall achievement of the mission-level goals, by (i) elaborating and scheduling these goals into sequences of control goals that nominally achieve the specified mission goals, (ii) dispatching these goals appropriately to the habitual module below, and (iii) adjusting the sequence of control goals in response to an onboard assessment of risk, which is based on current (and future) goals, current (and projected) systems state and current (and projected) environment state.

- The *habitual module* of the architecture is responsible for achieving the control goals dispatched by the deliberative module, by elaborating and executing the goals subject to the risk bounds embodied in constraints also provided by the deliberative module. The habitual module handles 'normally-seen' risks and

[a]This formal architectural analysis capability is still under development

failures, as long as the associated responses still satisfy the goals and constraints specified by the deliberative module. This module may also leverage a Correct-by-Construction Hybrid Control capability, which executes actions determined by a set of pre-compiled robust policies that are computed off-line and loaded onboard the spacecraft. These policies can be synthesized from a formal specification of desired behavior and a model of the system and its environment, so as to be provably-correct under a set of specified conditions.

- The *reflexive module* of the architecture is based on existing low-level control and device-level embedded software. Although critically important, the development of robust system software with reflexive characteristics is comparatively well-studied and understood.

- The *state estimation and diagnosis module* of the architecture is responsible for providing to the other modules accurate information about the state of the system and the environment. This module uses a combination of traditional state estimation techniques and state-of-the-art hybrid (discrete/continuous) state estimation and diagnosis techniques.

- *learning module* of the architecture is responsible for updating the models that the RSE algorithms use for reasoning, based on the results of executing in the environment, as well as the parameters used by these algorithms. This module may include a suite of state-of-the-art fault detection and diagnosis that can take into account sliding set points and handle transient conditions.

- The *analysis module* encompasses a set of model-based analysis capabilities that may be leveraged by the other modules of the architecture. For example, it may include algorithms that use the system behavior models, along with the current state estimates, to predict future states with some level of certainty, and provide these predictions to other modules. Another analysis capability may be deployed to verify the plans and policies being produced by other modules in the RSE.

The current implementations of the architecture described in Section IV of this paper do not include analysis and learning modules; consequently further discussion of these modules is beyond the scope of this paper. However, current work is underway to augment the current RSE implementations with learning and analysis capabilities.

A key paradigm in the architecture is to make use of the layered protocols with levels that abstract/virtualize resources. This makes it possible to integrate the above-mentioned modules into an effective resilience-enabling autonomy system. A generalized interface between the modules has been developed to enable the required interaction and coordination between modules; this generalized interface is described in the following section.

B. Canonical Software Architecture

In order to allow for a fair assessment and comparison between one RSE architectural implementation and another, we need (1) our implementation to be modular, and (2) to be able to formally specify the structure and behavior of the components and their interconnections. As part of this, we use a Canonical Software Architecture (CSA) format for the control modules in the RSE architecture,[15,16] in order to support the module decomposition at the levels of abstraction we choose, and clean separation of concerns and functionality between modules/components, while still maintaining the necessary communication and contingency management between components in the architecture. CSA also supports our need to explicitly and formally specify the algorithmic components and interconnections (rather than implicitly encode them into the architectural structure), enabling us to better leverage the benefits of model-driven software development and autocode generation techniques. We can then verify and validate the policies that the spacecraft uses when interacting with the environment, and internally between each module (e.g., to track down possible deadlock conditions between modules, or identify gaps in the handling of off-nominal execution between modules). This also scales up, allowing us to check and evaluate the combined procedures for each module across the entire architecture. Figure 2 shows an example CSA module.

Note that CSA builds off of the state analysis framework developed at JPL.[17–20] Another reason we use CSA is because allowing only one source of state knowledge to each module prevents the modules from getting out of sync and helps to disallow inconsistency in state knowledge. Each CSA module can be broken down into an Arbitration, Control, and Tactics component:[15,16]

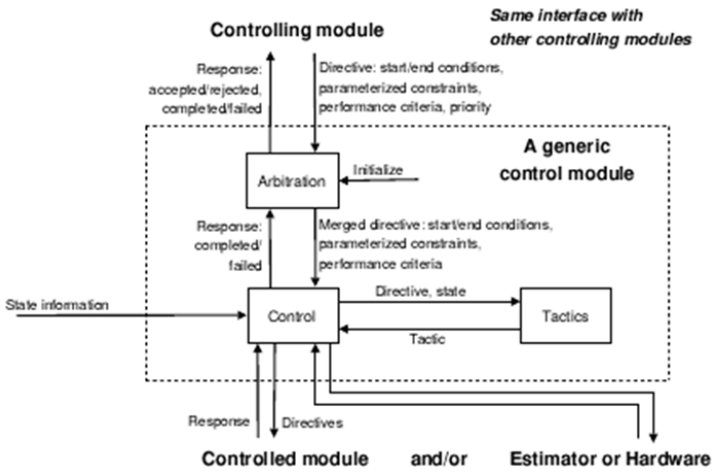


Figure 2: A generic control module in the Canonical Software Architecture.[15,16]

- The Arbitration component manages the overall behavior of the module by issuing a merged directive (goal) computed from all the received directives to Control, and reports goal status back to the issuing module.
- The Control component computes the output directives to other module(s) based on Arbitration's merged directive, responses received from other modules, and state information received; it also reports failure and completeness of a merged directive to Arbitration.
- The Tactics components selects or generates a control tactic or a contiguous series of control tactics for Control to use, based on the current state and directives in effect.

In the RSE implementation, each of these module components follows an explicitly-defined policy that chooses the component's internal action (algorithms) and can change the module's or the component's internal status, based upon the content of the various inputs to the component and its current internal status. For instance, the Arbitration component can include a state machine that determines whether a goal and constraints are accepted or rejected, and handles status messages and requests between the modules; if a goal and constraints are accepted it passes the information along to the Control component to be processed further.

III. Resilience Scenarios

We discuss a Mars rover scenario and an autonomous underwater vehicle (AUV) scenario below. Note that while the robots being used are very different physically, with different sensing and actuation, many aspects of the two scenarios are the same. Both vehicles have obstacles they need to avoid (rocks on the ground and crater rims versus ice flows above and the sea floor below); both have limited data rates and communication time windows that they have to meet (Mars communication relay satellites passing overhead at specific times versus geo-stationary Earth-orbiting satellites in view only at the surface of the water); both have uncertainty in position for long periods of time until an 'end-of-traversal'/'end of day' fix comes through (mission operators provide an updated location based on panoramic pictures sent by the rover, versus GPS fixes available only upon surfacing). Thus, at a higher level, many of the same algorithms can be used to solve for activity schedules, and test and map out trajectories that avoid obstacles, to remain at a risk level low enough for safe operations. This includes vehicle damage, as well as a risk of not meeting the science objectives for the given mission. We consider both the risk of damage to the vehicle as well as risk of not meeting the science objectives for the given mission.

A. Rover scenario – Mars simulation

1. *Nominal case*

In a map with obstacles containing five locations of interest ($l1$ through $l5$), a Mars rover is given a set of science requests, including 1) taking pictures from pre-specified locations; and 2) collecting two rock samples from three potential collection sites believed to have scientifically interesting rocks. The rover is equipped with two cameras: Mastcam is a high resolution camera specifically designed to take panoramic pictures, while Hazcam is a lower resolution camera whose purpose is to perform visual detection of unanticipated obstacles on the rover's path. However, when Mastcam is unavailable, we allow Hazcam to be used as a backup to take pictures of the target map location. In order to perform a rock sample collection, the rover has to first do a survey at the target site in order to detect the rock of interest. If it is found and targeted, the robot then performs the sampling procedure to collect the data.

Upon completion of its data collection (both taking pictures and collecting rock samples), the rover is required to drive to a location from which its science data can be transmitted to an orbiting satellite. The orbiter is visible from these locations within limited windows of time. When the data from each request is transmitted, the request is deemed completed. Figure 3 illustrates an instance of the scenario where the rover is tasked with transmitting three picture requests (at $l2$, $l3$, and $l5$) collecting two rock samples from three potential collection sites ($l2$, $l4$, and $l5$), coming to a total of five requests, i.e. science goals (g_1 to g_5). Data can only be transmitted from $l2$ and $l4$ within a time window between 50 and 2000 minutes from the start of the mission.

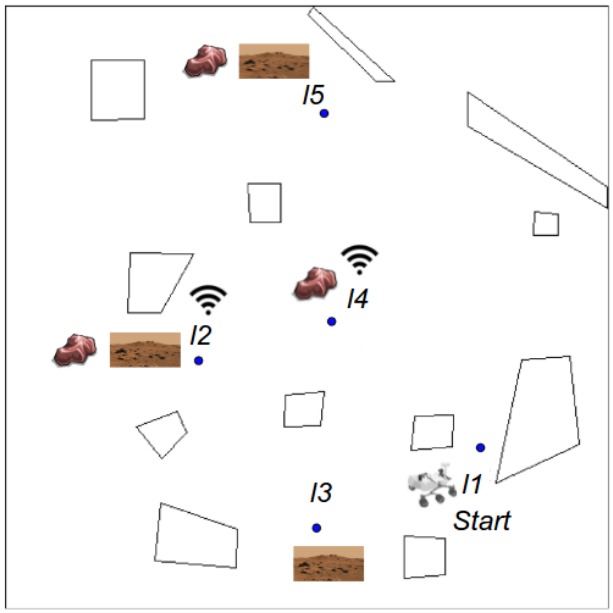


Figure 3: Mars Rover scenario example.

In order to achieve the specified mission, six types of actions are available to the rover. The available actions, along with their corresponding temporal duration probability density functions (Uniform or Gaussian distribution models, or Set-Bounded in which no information is available about the duration other than its lower bound and upper bound), are the following:

1) perform a traversal between two locations (Gaussian duration model);

2) activate a camera, which automatically turns off after use (Uniform duration model);

3) use an active camera to take a picture (Set-bounded duration model);

4) survey an area, which confirms or refutes the presence of a desired rock sample (Set-bounded duration model);

5) collect a rock that has been detected by a survey operation (Set-bounded duration model); and

6) transmit a request (picture or rock sample analysis) to the satellite, which can only be done during the period of time in which the satellite is visible (Uniform duration model).

In order to successfully complete its mission, the rover must complete all five science goals, and transmit the resulting data from a location in view of the relay orbiter. A valid plan for the mission is required to bound the risk of collision while traversing the environment, as well as the risk of missing the orbiter communication window.

2. Off-nominal cases

In addition to the challenge of generating a sequence of actions that achieve the aforementioned mission goals within acceptable risk levels, we investigated off-nominal cases in which unexpected failures and events occur during plan execution in the Mars rover scenario. In particular, we considered two off-nominal cases: 1) a Mastcam camera failure and 2) the inability to detect the presence of a scientifically-interesting rock to sample in a target location. For off-nominal case 1), we introduced a Mastcam activation fault at location $l3$, thus forcing the rover to activate Hazcam as a backup. For off-nominal case 2), the rover fails to detect a scientifically-interesting rock after its survey at location $l5$, forcing it to investigate other locations to find and transmit the target rock sample data.

B. AUV scenario – Earth-analogue mission

1. Nominal case

In this work we investigate underwater exploration missions on Earth that are analogues to our target planetary exploration missions. Herein we perform demonstrations of the RSE architecture in a Slocum glider, an autonomous underwater vehicle manufactured by Teledyne Webb Research. The Slocum glider is designed for long endurance missions so as to give a synoptic overview of some large area or phenomenon of interest. Before the start of a mission, a a plan (in the form of a script) must be uploaded to the glider that contains a list of waypoints to reach, the minimum and maximum depths to use, and how often to surface in order to obtain a GPS fix or communicate with the operators. These parameters can be updated during the mission, but only when the glider is surfaced and able to use its short-range radio or satellite phone.

In order to obtain hands-on experience with the glider in a real-world scenario, a preliminary partial implementation of the RSE was used during a technology validation cruise on board the R/V Falkor at the Scott Reef lagoon in the Timor Sea from March 24 to April 6, 2015. This expedition included AUVs from multiple research institutions and had an overarching goal of understanding the issues involved with having multiple AUVs operating in close proximity.[21] In particular, there were six underwater vehicles exploring the reef, but only the glider was controlled with (an off-board deployment of) RSE.

To specify AUV missions goals, operators discretized a specific area of the lagoon in fifteen regions of interest, cells, to be visited by the glider. Figure 4 illustrates the mission goals for the glider deployed during the expedition. Each cell was assigned a priority and a path (dashed red line) for the glider to traverse. All AUVs on the deployment (five others) shared the cells, but each had unique goals in each cell. In order to avoid collisions, a constraint was placed on the AUVs that no more than one AUV could occupy a cell at a time. These constraints were presented to the executive as temporal constraints on when regions were available (the other vehicles used manually programmed scripts and their plans were available while planning the glider's mission). The glider's goals in each region were chosen based on the location of interesting features of the ocean floor that would be visible to the glider's sonar.

Given the mission goals, the executive's task was to select and schedule a sequence of cell visitations around the schedule of the other AUVs while avoiding collision and maximizing science return. When planning paths in each cell there were two primary concerns. First, the planned paths should avoid obstacles, using user-specified buffers around the obstacles. Second, the paths should be minimum energy.

2. Off-nominal cases

During the AUV deployment at the Scout Reef a few off-nominal cases were encountered which resulted in interesting lessons learned to future deployments,[21] such as the one off the coast of Santa Barbara. First, unexpected changes in current, spatially and temporally (at depths and surface) drifted the glider from

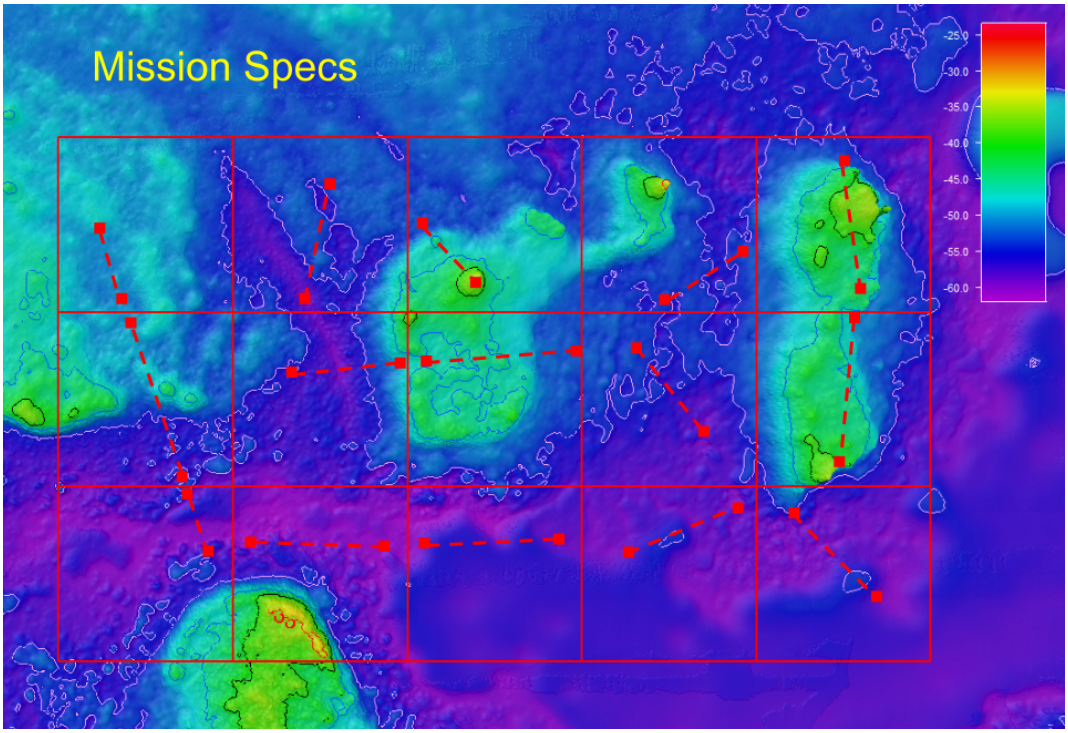


Figure 4: Mission specification provided during cruise expedition and used as input to the activity planner.

the expected trajectory and while surfacing. This required replanning the glider trajectory to account for corrections to the modeled currents in the lagoon. This motivates the need for routines to adaptively estimate currents in the current modeled location when transit durations start diverging significantly from modeled durations in future deployments. Second, due to the uncertainty of the currents and the travel time to target sites, it is non-obvious to a human operator when instruments should be programmed to turn on in order to observe regions of interest. In order to save energy and maximize deployment time, instruments should be placed into a standby state when not in use and then activated at proper times for optimal measurements.

Future deployment will incorporate generative planners to plan and schedule instrument configuration.

IV. Implementations/Realizations of the RSE Architecture

In this section of the paper, we first introduce the various resilience-enabling algorithms that we are incorporating in RSE. We then discuss three specific instantiations of RSE, which combine different subsets of these algorithms resulting in different resilience and autonomy features at the system level. Finally, we discuss a key tradeoff in implementation between flexibility and responsiveness of the system.

A. Resilience-enabling Algorithms

1. *CLARK – risk-aware activity planning*

The Conditional Planning for Autonomy with RisK (CLARK) system[22] is a combination of different tools developed for the generation of chance-constrained, conditional temporal plans for autonomous agents operating under uncertainty. It is an enabler for cognitive systems to decide what activities to perform and how to react to their outcomes, but also how to drive around terrain with obstacles and to schedule activities under spatial and temporal uncertainty.

The planner reads as input a chance-constrained partially observable Markov decision process (POMDP) and generates an optimal conditional temporal plan represented as a Temporal Plan Network (TPN).[23] Intuitively, the output TPN output consists of a set episodes (representing executable primitive actions or nested TPNs) and a set of temporal constraints relating the episodes. An example of such a TPN plan is

given in Figure 5.

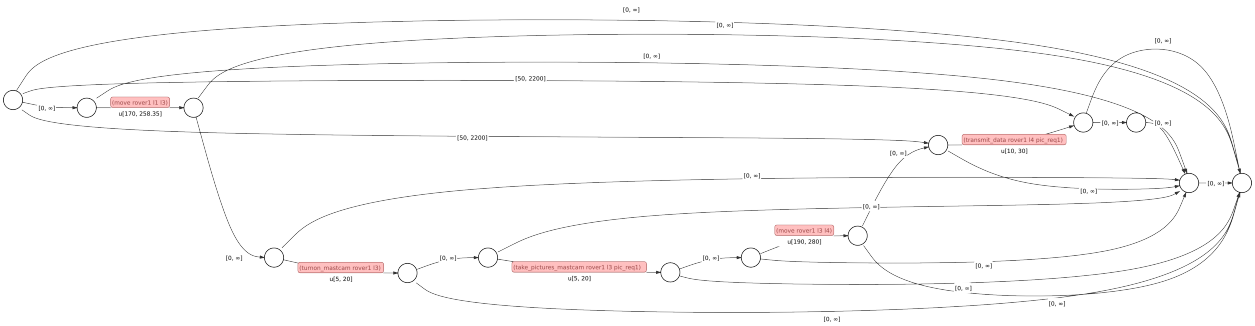


Figure 5: Simple TPN generated by CLARK.

The master mechanism, Risk-bounded AO* (RAO*, pronounced r-a-o-star),[23] builds upon a large body of research on probabilistic planning models, more specifically the AO* algorithm and its variants.[24] RAO* generates risk-bounded conditional policies from a chance-constrained a (POMDP) model by propagating risk bounds forward during a heuristic forward search while quickly pruning policy branches that are guaranteed to be too risky, ensuring that the conditional plans it generates are optimal and satisfy user-specified risk bounds. The algorithm guides the search towards optimal, chance-constrained policies by taking into account the "science value" of exploring each site, as well as how much mission risk has been incurred so far.

For the underlying chance-constrained path planning and scheduling problems, CLARK uses, respectively, the p-Sulu and PARIS[25] algorithms, both described below.

2. Kirk

Kirk is an optimal temporal planner that chooses between enumerable possible threads of execution so as to choose a plan that both is temporally feasible and minimizes cost. Kirk accepts a TPN as input; the TPN consists of a set of finite domain "decision variables" (choices), a set of guarded episodes, a set of guarded temporal constraints relating the episodes, and a cost function defined over the decision variables. The guards on episodes and temporal constraints state they are active only when the decision variables take on certain values. Kirk then outputs the set of assignments to the decision variables that is temporally consistent and minimizes cost also in the form of a TPN.

The current implementation of Kirk extends the first implementation as a graph-based search.[26] This version adds three new capabilities. First, we can now reason over probabilistic temporal constraints (PTCs) and chance constraints defined over them. We use a temporal consistency checker[27] to ensure a feasible schedule exists for any set of activated PTCs and associated chance constraints. Second, we have added the ability to specify non-decision variables and state constraints over both the decision and non-decision variables. This allows us to efficiently encode problems such as the vehicle routing problem without an exponential explosion in the number of episodes contained in the input TPN. Third, we have added the capability to call specialized sub-planners during Kirk's planning process. We currently use this with p-Sulu to compute risk-bounded paths for episodes that represent transits.

3. p-Sulu – risk-aware and risk-bounded trajectory planning

An efficient path planning approach for dynamic systems to handle non-convex state constraints is to formulate it as a Model Predictive Control (MPC) problem. Robustness against uncertainties (like state estimation error) is an important issue when MPC is applied to real-world robotic systems. The principal shortcoming of MPC based control techniques to explicitly uncertainties was addressed in the considerable body of work on Robust Model Prediction Control (RMPC), which assumes bounded disturbance.[28] However, in many practical cases, disturbance is often stochastic and unbounded.

Iterative Risk Allocation (IRA) is a two-stage optimization method for robust Model Predictive Control with Gaussian disturbances.[29] IRA divides the optimization problem into two stages; the upper-stage that optimizes risk allocation, and the lower-stage that optimizes control sequence with tightened constraints. It exploits the convexity of the upper-stage for fast convergence and a small suboptimal iterative descent algorithm.

Thus, the iterative risk allocation (IRA) algorithm[30] provides an optimal risk allocation capability for a wide range of problems. This was the basis for the *p-Sulu* algorithm.[31] p-Sulu takes a chance-constrained qualitative state plan (CCQSP)[32] representation as an input and outputs an optimal sequence of actions as a schedule; it is built upon chance-constrained model predictive control (CCMPC) methods[29,33,34] and works over continuous state spaces. Two example applications are vehicle path planning[30,35,36] and building control.[37]

4. PARIS – risk-bounded scheduling

The subproblem of chance-constrained strong scheduling is handled by the PARIS algorithm.[25] PARIS receives a TPN as input and returns a strong schedule (if one exists). While the state-of-the-art invariably resorted to general-purpose nonlinear solvers to implement probabilistic scheduling methods, PARIS leverages a full linear encoding of the probabilistic scheduling problem that allows it to drastically reduce its runtime and memory requirements while provably running in polynomial time. Such property is particularly useful in planetary rovers in which on-board computation and energy are scarce.

5. Pike – execution and monitoring

Once a planner (e.g., CLARK, Kirk, or a PDDL planner) has found and elaborated a plan, its execution has to be properly managed and monitored. Pike[38] is responsible for both dispatching the actions in the plan and monitoring the estimate of the state of the world to ensure the plan is being executed correctly. Pike takes in a contingent plan in the form or a probabilistic temporal plan network (pTPN), as well as sensory inputs and state estimates. It outputs a schedule for its actions such that the plan is expected to succeed. If an issue or off-nominal situation is detected at runtime (such as an action taking much longer than expected and threatening the rest of the plan), Pike alerts the planner of the issue and requests a new plan with the same goal subset.

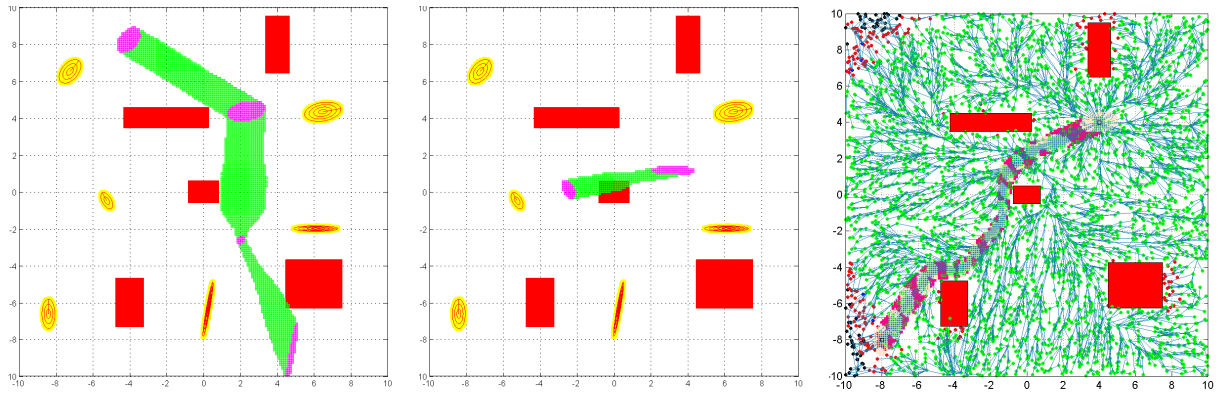
6. RRT$^{\#}$ *– risk-bounded trajectory planning*

Early implementations of the habitual module of RSE integrated the RRT* trajectory elaboration algorithm (a variant of the Rapidly-exploring Random Tree algorithm[39] with asymptotic optimality guarantee).[40,41] However, this algorithm does not incorporate uncertainty during the computation of the motion plan. Hence, even though the deliberative module generates a risk-aware plan with path constraints, the path computed by the RRT* algorithm may not be risk-aware and violate the given risk bounds (e.g., positional uncertainty). In order to remedy this, we developed a robust motion planner that uses the RRT$^{\#}$ algorithm,[42] which is a sampling-based motion planning algorithm with asymptotic optimality guarantees. Given an initial configuration and a goal configuration, the RRT$^{\#}$ algorithm incrementally builds a graph in the configuration space and computes the best path encoded in that graph at every iteration by using replanning procedures that are similar to that of the Lifelong Planning A* algorithm (LPA*).[43] In this work, we are given a list of configurations with their associated (position) uncertainty ellipse information, and the planner considers bounded uncertainty in the motion, which guarantees that there is no obstacle within given uncertainty ellipses over the course of motion.

The risk-bounded RRT$^{\#}$ algorithm has been integrated into the RSE architecture as follows:

- deliberative module: p-Sulu provides waypoints and uncertainty information based on the risk allocation.
- habitual module: the robust planner (RRT$^{\#}$) computes a path that respects the risk bounds.

This robust motion planner is able to compute a path between the input waypoints that yields the minimum uncertainty. To do so, first define a simple *uncertainty prediction* procedure in order to compute the uncertainty ellipse for a given query point in the environment. Then, given a set of training points and their corresponding uncertainty ellipse information, the uncertainty ellipse of an arbitrary point is computed by using locally weighted learning as shown in Figure 6(a). This procedure is repeatedly called for a discrete set of points along a given path, and the overall uncertainty of the path – the *cost evaluation* – is computed as the measure of union of the ellipse for each point along the path as shown in Figure 6(b). A simulation result is provided in Figure 6(c). The planner is tasked to find a path between bottom-left and top-right points such that it yields minimum uncertainty. As shown in Figure 6(b), the planner returns a path that has enough clearance from obstacles where there is high uncertainty.



(a) (b) (c)

Figure 6: RRT# example run. Obstacles are shown in red, sets of training points and uncertainty ellipse information are shown in yellow, uncertainty for a given path is shown in green and purple.

7. *TuLiP – risk-bounded activity planning and hybrid control*

TuLiP is an implementation of one of a set of new approaches that has been created within the last decade for the specification, design, and verification of embedded control systems.[44] These approaches make use of models of the dynamics of the system, descriptions of the external environment, and formal specifications to either verify that a given design satisfies the specification or synthesize a controller that satisfies the specification, as summarized in Figure 7.

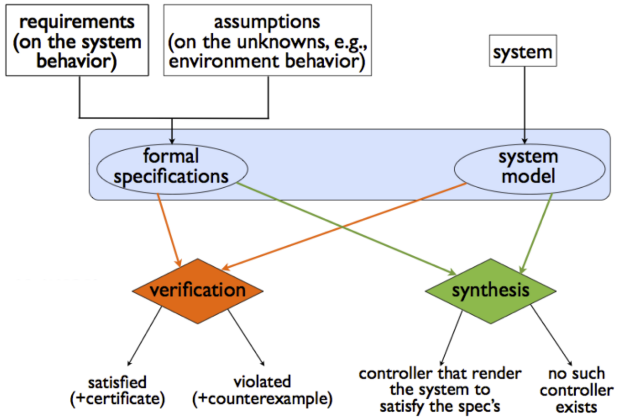


Figure 7: Verification and synthesis framework.

As part of this project, we have used TuLiP for both symbolic activity planning (in the deliberative module) and hybrid control problems (in the habitual module). The former application was set up to produce satisficing plans (when they exist) by allocating risk effectively between PDDL-like actions, to achieve the goals within the risk constraints given,[3] looking into the possibility of using patching techniques with TuLiP[45] and the efficiency of choosing differing levels of abstraction for rover trajectory planning. In the future, we plan to make a version of this algorithm available for real-time planning that leverage the gaming and turn-taking aspects of the algorithm, to be used in conjunction with other deliberative components as a plan verification tool for the creation and evaluation of contingency plans.

8. *PDDL planners*

In order to benefit from the large body of research and recent development on automated planning and scheduling,[46] we have integrated traditional activity planning approaches into the RSE architecture. In this work we focus on the use of off-the-shelf domain-independent planners which accept input in the *Planning*

Domain Definition Language (PDDL).[47] This language was designed to encode the domain physics of a wide range of applications and it is considered the standard input representation for planners in the AI Planning and Scheduling community. Although traditional PDDL planners do not implement risk-awareness, one could model risk as a resource in PDDL and allow the planner to consider risk bounds during mission planning. In this work we have integrated a set of PDDL planners, including OPTIC,[48] POPF,[49] and SGPlan.[50]

9. Bones – hybrid state estimation

Bones is a state estimator for hybrid discrete/continuous systems modeled as a set of concurrently operating probabilistic hybrid automata (PHAs). Each PHA is described by a set of discrete operating modes as well as continuous state and interface variables.[51,52] The interface variables are shared between PHAs and are use to model interactions between the different components of the system. The dynamics of the state and interface variables are determined by the active operating mode and described as a set of algebraic and differential equations. A subset of the continuous variables are observable (with noise) and the operating modes are not directly observable.

Bones maintains a probability distribution over possible modes of the entire system, as well as a probability distribution over the continuous state for each mode. The current implementation uses extended Kalman filters to estimate the continuous state, but any continuous estimator could be substituted. Bones is based on previous work[51] and manages the exponential growth in the discrete estimate using a search algorithm to focus on the most likely best estimates and pruning the remaining modes. The search algorithm used is A* with Bounding Conflicts,[53] a variant of conflict-directed A*[54] that uses information learned during the search to improve its heuristic.

B. Tested Resilient Architecture Implementations

As specified previously, the RSE architecture is modular to enable extension and reconfiguration, as long as the embedded algorithmic components exhibit the required risk-aware behavior in the deliberative module and risk-bounded behavior in the habitual module. To that end, we discuss in this section three particular RSE configurations that we have implemented and tested for the Mars rover and AUV cases. Of the two Mars rover realizations, the first focuses on the mobility capability, while the second focuses on multi-activity planning and scheduling capability (where traversals is just one of rover's available activity). The AUV-focused realization corresponds to the partial implementation of RSE deployed for the autonomy demonstration in the Scott Reef, described above in Section III.

We have adopted the Robot Operating System (ROS) messaging system (and ROSbridge) for our proof-of-concept architecture.[55,56] Herein, ROS serves as a foundation for intercommunication among the different modules and algorithmic components. ROS' publisher-subscriber message-passing framework is robust, and there exist a wide range of robots and simulated robots that have pre-existing interfaces to the software package, which allows us to easily test our architecture across a wide range of use cases. In the planetary roving cases, for simplicity we built off of the ROSARIA API[57] used for communicating with Pioneer robots (command velocities in the body frame and raw sensor data return). The ROS implementations of the RSE includes further messaging support for waypoint-following, status queries, and other requests between modules (e.g., goal and constraint passing, state space updates, consistent with the CSA framework described in Section II.B, above). Moreover, the RSE realization for the planetary rover case uses a medium-fidelity Gazebo-based simulator[58] to represent the vehicle and the environment. The gazebo software allows for a wide range of robots to be tested using the main RSE software backbone. The RSE realization for the AUV case executes on a off-board computer that communicates with the AUV control software onboard the glider, or a high-fidelity glider simulator.

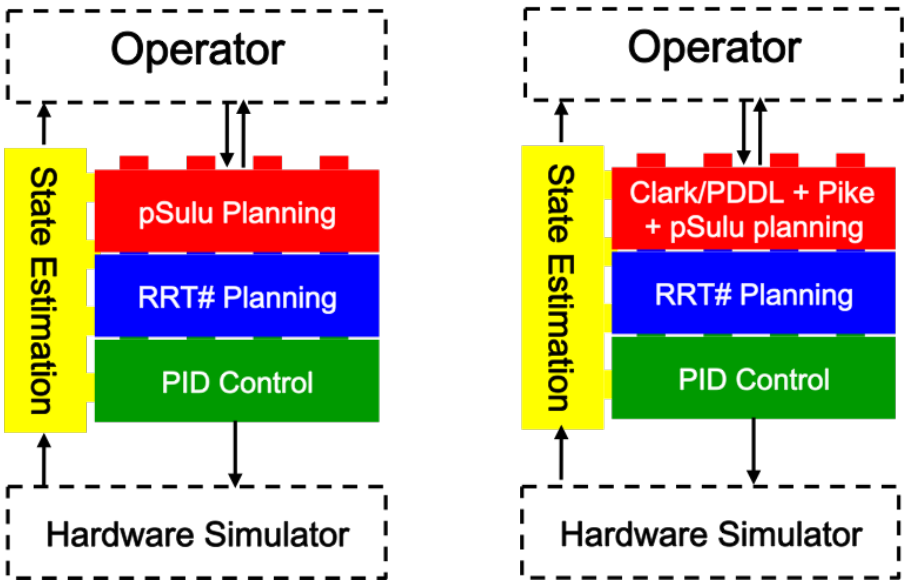
1. Rover Mobility Implementation

Our initial proof-of-concept RSE demonstration (discussed in a previous paper[3]) focused on rover mobility capabilities, and thus it deployed risk-aware path planning capabilities in the deliberative module, and risk-bounded trajectory elaboration in the habitual module.

This particular rover scenario includes external goals that require traversal to various map locations, and can be run with or without the injection of sensor and actuator degradations/failures, as well as measurement uncertainty. In this demonstration, the deliberative module performs trajectory planning on a

lower-resolution global scale using p-Sulu, while the habitual module performs more refined local trajectory planning between waypoints by employing RRT#. The reflexive module employed a simple PID controller. Figure 8(a) shows a high level overview of the building blocks used in this demonstration. The functionality embedded in each of the modules is listed in Table 1. Our previous paper[3] describes the timeline of onboard rover operations and communications expected to occur between the two upper-most layers similar to our scenario, with an exception that we now employ a risk-bounded sample-based motion planner at the habitual layer.

The actual obstacles for the demonstration are retrieved from information about the topography prior to the deployment of the rover. They cover the regions in the environment forbidden for roving (depicted by the green rectangles in Figure 9). Figure 9(a) shows the computed waypoints (in red) for a low-risk plan, meaning that the planner allows for a more efficient trajectory passing very close by the obstacles while going to the target. Alternatively, Figure 9(b) shows a high-risk plan; in order to meet the stronger risk requirements, the planner increases the margin with respect to forbidden regions, resulting in a longer overall path to the objective.



(a) p-Sulu and RRTSharp

(b) CLARK/PDDL, Pike, p-Sulu and RRTSharp

Figure 8: Resilient Spacecraft Executive Architecture realizations.

2. *Rover Activity Planning Implementation*

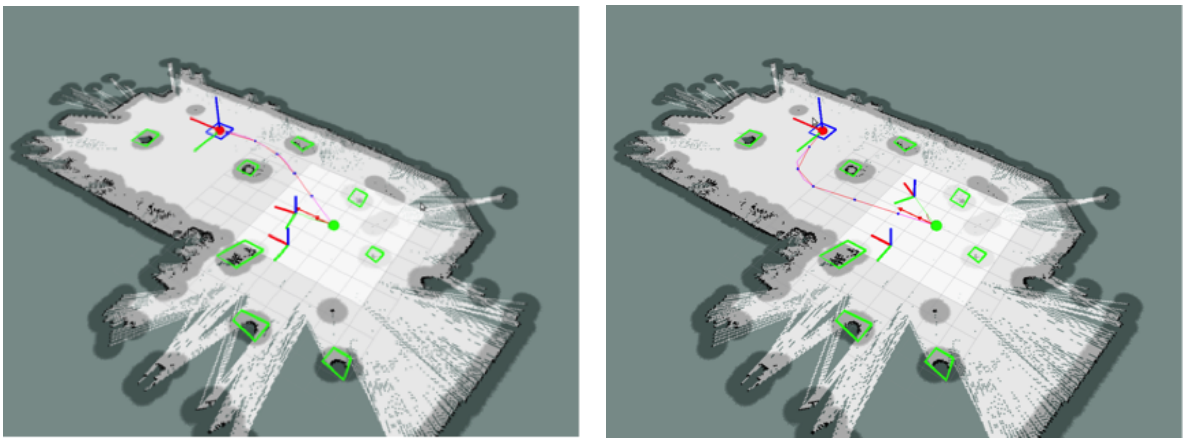
More recent work has augmented the RSE implementation by integrating activity planning capability into the deliberative module, in which the rover is able to reason not only about traversal actions, but also different science gathering and communication activities, along with their different duration models (as presented in Section III, A, and Figure 8(b)). In this case, a high level risk-aware activity planner (CLARK)[22] synthesizes goal-based sequences to resiliently accomplish the mission. We have also previously integrated PDDL planners as the activity planning component, but in this work we focus on the use of a risk-aware activity planner: CLARK.

Once the CLARK planner generates a feasible plan under the given risk constraints, an execution and monitoring system (Pike)[38] is responsible for both dispatching the action and monitoring its progress. A unique capability of CLARK as a risk-aware activity planning component is that it allows consideration of different temporal duration models to represent the uncertainty in the duration of rover activities. We use the p-Sulu[30] algorithm for risk-aware path planning in the deliberative module to implement each traversal activity.

In the the habitual module we incorporate the risk-bounded trajectory planner RRT# in planning further motions between waypoints generated by p-Sulu. State estimates are sent to the planners and monitors

Module	Functionality	Algorithms
Deliberative	• Risk-aware path planning under state uncertainty • Visual Odometry control (On/Off) • Resource-bounded plan generation	p-Sulu
Habitual	• Sample-based motion planning • Resource checking • Command dispatching	RRT#
Reflexive	• Command execution • Position and velocity control	PID Controller
State Estimation	• Provides state estimation to all modules • Position & orientation state estimation • Health state estimation	Simple state filters

Table 1: Overview of the Functionality and Algorithms used in each module in the Rover Mobility Implementation.



(a) High Risk (b) Low Risk

Figure 9: Demonstration of the risk awareness using p-Sulu and RRT*.

to communicate the current state of the world and for monitoring the progress of goal achievements. A hybrid estimation capability (Bones) enables diagnosis of subtle degraded and failure modes of behavior and components (e.g., health of the Mastcam and Hazcam dependent on their current operating temperature). Table 2 provides an overview of the suite of algorithms within each module while Figure 10 shows the simulated environment in gazebo and the different components coordinating to execute the mission and adapt to unexpected events.

In this implementation, we also focus on resilience regarding off-nominal cases. In the current implementation of RSE, we follow a replan-from-scratch approach in which the activity planner generates a new plan for the remaining goals at every unexpected disturbance that cause the current plan to fail to achieve the missions goals: 1) camera failure; and 2) the inability to detect a scientifically-interesting rock to sample in a target location. In both cases, the exogenous events are detected by the RSE monitoring system (Pike) and then managed by RSE's activity planner (CLARK), which replans the mission in light of these unexpected contingencies. In particular, it handles the failure of the Mastcam by replanning all remaining imaging activities to use the Hazcam, and it handles the absence of a scientifically-interesting rock at one site by replanning the activities at a later site to include the sampling activity that was missed. More details about this instance of RSE with activity planning can be found in Santana et al.[22] A video of the demonstration can be found at `http://mers.csail.mit.edu/video-files/rss/Resilient_Space_Systems_Midyear_Review_April_`

Module	Functionality	Algorithms
Deliberative	• Risk-aware activity planning • Risk-bounded activity scheduling with different temporal duration models • Activity dispatching • Activity failure and delay monitoring • Replanning under failure or delay • Probabilistic kino-dynamic path planning for traversal activities	CLARK (or PDDL Planners) Pike p-Sulu
Habitual	• Risk-bounded sample-based motion planning • Command dispatching	RRTSharp
Reflexive	• Command execution • Position and velocity control control	PID Controller
State Estimation	• Provides state estimation to all modules • Hybrid state estimation for camera failure detection • Position & orientation state estimation	Bones Simple state filter

Table 2: Overview of the Functionality and Algorithms at each module in the Rover Activity Planning Implementation.

`5th_2016.mp4`.

3. AUV Implementation (Scott Reef, Australia)

In this deployment, a simplified version of the risk-aware goal-directed executive was used as a decision support system for a Slocum glider with an attached scanning sector sonar. The main funtionalities and algorithms used in this implementation and demonstration are shown in Table 3.

The operators used the executive to plan a series of observations of target regions between surfacings for data communication. They then transformed the plans into command scripts that were directly executable by the glider.

The executive received as input the missions goals depicted in Figure 4, along with temporal constraints, the glider dynamics, and the lagoon's bathymetry. Kirk's task was to select and schedule a sequence of cell visitations around the schedule of the other AUVs while avoiding collision and maximizing science return. When planning paths in each cell there were two primary concerns. First, the planned paths should avoid obstacles, using user-specified buffers around the obstacles. Second, the paths should be minimum energy. While shortest paths in the reef were easy to find (there was a straight line path between most points), the shortest paths typically required the glider to pass over obstacles at a shallow depth. Due to the glider's method of propulsion, these shortest distance paths would require more inflections, rather than taking a longer, deeper path. Figure 11 provides an example of an efficient path computed by the path planner.

At the beginning of the deployment, the p-Sulu path planner was used to plan transits for nine days in initial testings. At the time of the cruise, p-Sulu used a simplified dynamics model and relied on the operator for risk allocation. The activity and path planner prototypes were then used in conjunction to successfully plan for two days of eight hour operations for the glider. The activity planner efficiently (1) selected subset of science goals with highest return based on science preference, and (2) ordered and scheduled visitation to respect the aforementioned constraints. Ocean currents in Scott Reef changed frequently and posed a challenge for the AUVs deployed during the expedition. The path planning component successfully planned safe routes around the reef. Moreover, we demonstrated the executives capability to support re-planning after each glider surface activity. To the best of our knowledge, a Slocum glider has never before been used inside a reef before, due to the challenges present in that environment.

Another deployment of the glider with the RSE and a full version of the goal-directed executive with a full deployment of the RSE capability is planned for September 2016 off the coast of Santa Barbara, CA. This

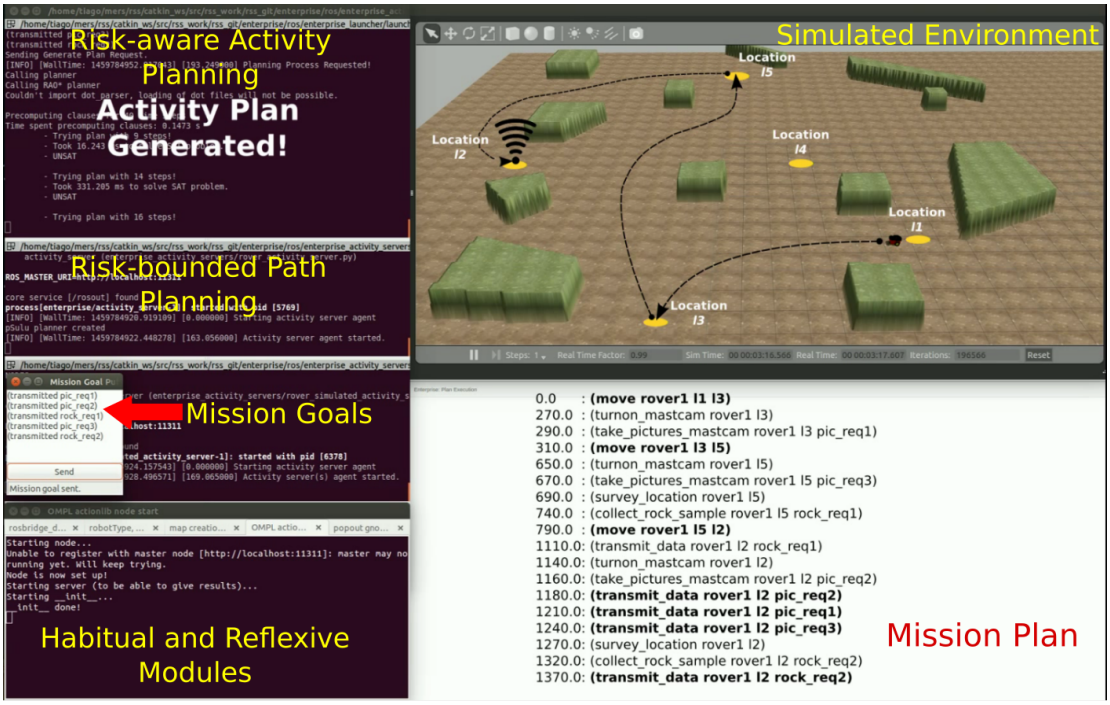


Figure 10: Rover Activity Planning Implementation with mission plan example.

upcoming deployment will use as input areas of interest along with preferences and temporal constraints.

C. Implementation Tradeoffs: Flexibility versus Responsiveness

Our ultimate aim for this project is to develop (1) an autonomous control architecture that can exhibit system behavior within each module in the architecture and every level of abstraction, and (2) a rigorous analysis framework that enables appropriate allocation of capability to each level depending on the problem at hand (i.e., the system onto which we are deploying our architecture, the environment it is operating in, and the mission it is intended to perform).[1] Consider the following example of the latter analysis capability, from a prior paper:[1] A reasonable design choice for a rover system operating in a particularly complex and hazardous planetary surface environment might allocate path planning to a deliberative module, trajectory following control to a habitual module, and low-level mobility control to a reflexive module; this capability

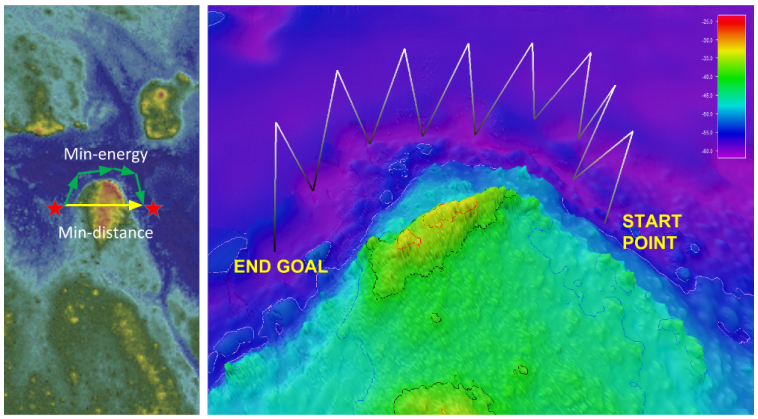


Figure 11: Example of an efficient glider trajectory generated by the path planner. A top view (left) and a perspective view (view) of the path taken from start point to end point in the lagoon.

Module	Functionality	Algorithms
Deliberative	• Science goals selection based on science preference • Site visitation ordering and scheduling • Planning safe routes around the reef and other vehicles • Command script generation	Kirk p-Sulu
Habitual	• Buoyancy engine depth inflection (onboard glider) • Rudder heading adjustment (onboard glider) • Pitch adjustment (onboard glider)	simple sequenced behaviors
Reflexive	• Safety monitoring to detect abnormal events (onboard glider) • Power systems control (on-board glider) • Communications sequencing (on-board glider) • Emergency ascent and shutdown (on-board glider)	simple controllers
State Estimation	• Health state estimator (onboard glider) • Navigation state estimation (onboard glider) • Sensor/environmental state estimation (onboard glider)	simple state filters

Table 3: Functionality and Algorithms used in each module for the AUV Implementation. Note that the deliberative module for this implementation executes offboard the glider, and communicates the generated plans ("command scripts") with the habitual and reflexive modules onboard the glider.

allocation would enable the system to be robust in its ability to flexibly replan its trajectory as obstacles come into view, but the need for additional computation in the control loop would consequently slow the overall progress of the rover, preventing it from achieving high traverse speeds. Conversely, a rover system operating in a much more benign environment might implement path selection as a pre-validated behavior that does not require deliberation, and might push trajectory-following control down into the more responsive but less flexible reflexive module; this capability allocation would help enable faster driving, but would come at the expense of costly backtracking if an obstacle is ever encountered. Thus, the size and scope of the problem domain, and how it is decomposed, can cause issues in providing just-in-time solutions within given constraints, depending on the problem; a mismatch in decomposition versus the constrained use case can lead to a bad design and an unworkable system. Simply put: the choice of level of abstraction and the appropriate allocation of functionality across the architecture are important.

In order to support this, we need a 'toolbox' of algorithms that we can choose from that may provide an overlapping functionality – but this overlap makes them useful alongside or in conjunction with other similar algorithms. No one algorithm is good at solving every problem; each algorithm has its strengths and weaknesses and best uses. Some of these algorithms may have different strengths or weaknesses or work better in some domains under some assumptions than others; others may just resolve quickly enough to be useful because they can be rerun almost instantaneously when issues occur, or conversely require a long time to compute but are guaranteed to successfully execute within a set of specified constraints. Thus, as above, there is a motivation to including both relatively simple algorithms that can give a quick and useful result (e.g., an action-plan that takes risk into account), and algorithms that are more complex. For instance, the risk-aware planning algorithms have been extended to include probabilistic uncertainty in time duration and outcome of actions, but do not currently produce plans that take into account the possibility of off-nominal environment events; in the event that an unexpected obstacle is sensed or actuator degradation occurs, the risk-aware planners would essentially need to restart the entire planning process, using a new obstacle map or robot model. However, alternate algorithms like TuLiP can be used to supply this functionality if and when needed. For this reason, we seek to add to our library of algorithms for RSE, particularly to include

algorithms with different relative strengths and weaknesses compared to the current set in the library.

We are also beginning to attempt to use model-based system analysis tools that will enable us to formalize such system tradeoffs, e.g., between flexibility and responsiveness,[1] and determine what we call total system stability for the entire architecture. More information on this subject can be found in another paper.[59]

V. Conclusions and Future Work

We have discussed the developments on the RSE architecture that allows resilient, risk-aware operations in real-time in uncertain and changing environments. We have discussed new and updated algorithmic capabilities that have been developed over the course of the project to allow for autonomous risk-aware and risk-bounded decision-making by the robotic system. We have discussed the results from two RSE implementations for a Mars rover scenario, and a successful AUV deployment. Currently-funded projects are expanding of the scope of the RSE work into another domain with different requirements, and enabling the integration of machine learning capabilities for even greater resilience. Future work will leverage a formal architectural model we have developed of the RSE software, to enable more rigorous architectural analyses and autocode generation of the software structure and many of the default behaviors of the RSE components.

Acknowledgments

The authors would like to thank the Model-based Embedded Robotic Systems Group at MIT for their input and feedback throughout the development process, especially Pedro Santana and Eric Timmons for all their help in developing the risk-aware planning executive (the Enterprise system) and its demonstrations. We would also like to thank Rich Camilli and Erez Karpas for their input and their efforts. The authors would also like to thank the Keck Institute of Space Studies for its initial study and final report on Engineering Resilient Space Systems, from which this effort originated.

The research described in this paper was carried out at the Jet Propulsion Laboratory under a contract with the National Aeronautics and Space Administration, and at the California Institute of Technology, the Massachusetts Institute of Technology and Woods-Hole Oceanographic Institution under a grant from the Keck Institute for Space Studies.

References

[1]McGhan, C., Murray, R., Serra, R., Ingham, M., Ono, M., Estlin, T., and Williams, B., "A risk-aware architecture for resilient spacecraft operations," *Aerospace Conference, 2015 IEEE*, March 2015.

[2]Wongpiromsarn, T., Topcu, U., Ozay, N., Xu, H., and Murray, R. M., "TuLiP: A Software Toolbox for Receding Horizon Temporal Logic Planning," *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, HSCC '11, ACM, New York, NY, USA, 2011, pp. 313–314.

[3]McGhan, C. L. and Murray, R., "Application of Correct-by-Construction Principles for a Resilient Risk-Aware Architecture," *AIAA SPACE 2015 Conference and Exposition*, 2015.

[4]Muirhead, B. K., "Deep Impact, the mission," *IEEE Aerospace Conference Proceedings*, Vol. 1, No. 1968, 2002, pp. 147–155.

[5]Brown, D., "NASA's Deep Impact Produced Deep Results," URL: `http://www.nasa.gov/mission_pages/deepimpact/media/deepimpact20130920f.html`, 2013.

[6]NASA Jet Propulsion Laboratory, "JPL Missions - Deep Impact / EPOXI," URL: `http://www.jpl.nasa.gov/missions/deep-impact-epoxi`, 2014.

[7]Gray, D. L. and Brown, G. M., "Fault-Tolerant Guidance Algorithms for Cassini's Saturn Orbit Insertion Burn," *Proceedings of the American Control Conference (ACC 905)*, June 1998.

[8]Anderson, J. D., Campbell, J. K., and Nieto, M. M., "The energy transfer process in planetary flybys," *New Astronomy*, Vol. 12, No. 5, 2007, pp. 383–397.

[9]Braun, R. and Manning, R., "Mars exploration entry, descent and landing challenges," *IEEE Aerospace Conference*, Vol. 44, No. 2, 2006, pp. 310–323.

[10]Gostelow, K. P., "The Mars Science Laboratory Entry, Descent, and Landing Flight Software," *Proceedings of the 23rd AAS/AIAA Spaceflight Mechanics Meeting*, 10-14 Feb. 2013.

[11]Bernard, D., Dorais, G., Fry, C., Gamble, E., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P., Pell, B., Rajan, K., Rouquette, N., Smith, B., and Williams, B., "Design of the Remote Agent experiment for spacecraft autonomy," *IEEE Aerospace Conference*, 1998, pp. 259–281.

[12]Nayak, P. P., Bernard, D. E., Dorais, G., Kanefsky, E. B. G. J. B., Gamble, E. B., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Rajan, K., Rouquette, N., wen Tung, Y., Smith, B. D., and Taylor, W., "Validating The DS1 Remote Agent Experiment," 1999.

[13]Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C., "Remote Agent: To Boldly Go Where No AI System Has Gone Before," 1998.

[14]Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., "Using iterative repair to improve the responsiveness of planning and scheduling," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, 2000, pp. 300–307.

[15]Burdick, J. W., du Toit, N., Howard, A., Looman, C., Ma, J., Murray, R. M., and Wongpiromsarn, T., "Sensing, Navigation and Reasoning Technologies for the DARPA Urban Challenge," *Technical report, DARPA Urban Challenge Final Report*, 2007.

[16]Wongpiromsarn, T. and Murray, R. M., "Distributed mission and contingency management for the DARPA Urban Challenge," *International Workshop on Intelligent Vehicle Control Systems, 2008*, IEEE, 2008, p. Submitted.

[17]Dvorak, D., Rasmussen, R. D., Reeves, G., and Sacks, A., "Software architecture themes in JPL's Mission Data System," *Proceedings of 2000 IEEE Aerospace Conference*, 2000.

[18]Rasmussen, R. D., "Goal based fault tolerance for space systems using the Mission Data System," *Proceedings of 2001 IEEE Aerospace Conference*, 2001.

[19]Barrett, A., Knight, R., Morris, R., and Rasmussen, R., "Mission planning and execution within the mission data system," *Proceedings of the International Workshop on Planning and Scheduling for Space*, 2004.

[20]Ingham, M., Rasmussen, R., Bennett, M., and Moncada, A., "Engineering complex embedded systems with state analysis and the mission data system," *Journal of Aerospace Computing, Information and Communication*, 2005.

[21]Timmons, E., Vaquero, T., Williams, B. C., and Camilli, R., "Risk-aware Planning Executive for Autonomous Underwater Gliders," *Proceedings of ICAPS Planning and Robotics Workshop*, 2016.

[22]Santana, P., Vaquero, T., Timmons, E., Williams, B., McGhan, C., Murray, R., and Toledo, C., "Risk-aware Planning in Hybrid Domains: An Application to Autonomous Planetary Rovers," *AIAA SPACE 2016 Conference and Exposition*, 2016. (Accepted).

[23]Santana, P., Thibaux, S., and Williams, B., "RAO*: an Algorithm for Chance-Constrained POMDP's," *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

[24]Nilsson, N. J., *Principles of artificial intelligence*, Springer, 1982.

[25]Santana, P., Vaquero, T., Toledo, C., Wang, A., Fang, C., and Williams, B., "PARIS: a Polynomial-Time, Risk-Sensitive Scheduling Algorithm for Probabilistic Simple Temporal Networks with Uncertainty," *Proceedings of the 26th International Conference on Automated Planning and Scheduling*, 2016.

[26]Kim, P., Williams, B. C., and Abramson, M., "Executing Reactive, Model-based Programs through Graph-based Temporal Planning," *Proceedings of the International Joint Conference on Artificial Intelligence*, Seattle, WA,, 2001, pp. 487–493.

[27]Wang, A. J. and Williams, B. C., "Chance-constrained Scheduling via Conflict-directed Risk Allocation," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, January 2015.

[28]Kothare, M. V., Balakrishnan, V., and Morari, M., "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, Vol. 32, No. 10, October 1996, pp. 1361–1379.

[29]Ono, M. and Williams, B. C., "Iterative Risk Allocation: A New Approach to Robust Model Predictive Control with a Joint Chance Constraint," *Proceedings of 47th IEEE Conference on Decision and Control*, 2008.

[30]Ono, M. and Williams, B. C., "An Efficient Motion Planning Algorithm for Stochastic Dynamic Systems with Constraints on Probability of Failure," *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08)*, 2008.

[31]Ono, M., *Robust, Goal-directed Plan Execution with Bounded Risk*, Ph.D. thesis, Massachusetts Institute of Technology, 2012.

[32]Blackmore, L., *Robust Execution for Stochastic Hybrid Systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[33]Blackmore, L., Li, H., and Williams, B., "A probabilistic approach to optimal robust path planning with obstacles," *American Control Conference, 2006*, IEEE, 2006, pp. 7–pp.

[34]Ono, M., "Joint Chance-Constrained Model Predictive Control with Probabilistic Resolvability," *Proceedings of American Control Conference*, 2012.

[35]Ono, M., Williams, B., and Blackmore, L., "Probabilistic Planning for Continuous Dynamic Systems," *Journal of Artificial Intelligence Research*, Vol. 46, 2013, pp. 449–515.

[36]Jewison, C., BcCarthy, B., Sternberg, D., Fang, C., and Strawser, D., "Resource Aggregated Reconfigurable Control and Risk-Allocative Path Planning for On-orbit Assembly and Servicing of Satellites," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AAAI, 2014.

[37]Ono, M., Graybill, W., and Williams, B. C., "Risk-sensitive Plan Execution for Connected Sustainable Home," *Proceedings of the 4th ACM Workshop On Embedded Systems (BuildSys)*, 2012.

[38]Levine, S. J. and Williams, B. C., "Concurrent plan recognition and execution for human-robot teams," *ICAPS–14*, 2014.

[39]LaValle, S. M., "Rapidly–exploring random trees: A new tool for path planning," *TR 98–11*, Computer Science Department, Iowa State University, October 1998.

[40]Karaman, S. and Frazzoli, E., "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, Vol. 104, 2010.

[41]Karaman, S. and Frazzoli, E., "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, Vol. 30, No. 7, 2011, pp. 846–894.

[42]Arslan, O. and Tsiotras, P., "The Role of Vertex Consistency in Sampling-based Algorithms for Optimal Motion Planning," *arXiv*, 2012, pp. 1–26.

[43]Koenig, S., Likhachev, M., and Furcy, D., "Lifelong Planning A*," *Artificial Intelligence*, Vol. 155, No. 1, 2004, pp. 93–146.

[44]Wongpiromsarn, T., Topcu, U., and Murray, R. M., "Synthesis of Control Protocols for Autonomous Systems," Vol. 1, 2013, pp. 21–39.

[45]Livingston, S. C., Prabhakar, P., Jose, A. B., and Murray, R. M., "Patching task-level robot controllers based on a local -calculus formula," *Proceedings of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, May 2013, pp. 4573–4580.

[46]Ghallab, M., Nau, D., and Traverso, P., *Automated planning: theory & practice*, Elsevier, 2004.

[47]Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D., "PDDL—The Planning Domain Definition Language," 1998.

[48]Benton, J., Coles, A. J., and Coles, A. I., "Temporal Planning with Preferences and Time-Dependent Continuous Costs." *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS-12)*, June 2012, pp. 2–10.

[49]Coles, A. J., Coles, A., Fox, M., and Long, D., "Forward-Chaining Partial-Order Planning." *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, 2010, pp. 42–49.

[50]Hsu, C.-W. and Wah, B. W., "The SGPlan Planning System in IPC-6," *In the booklet of the International Planning Competition (IPC), International Conference on Planning and Scheduling (ICAPS)*, 2008.

[51]Hofbaur, M. W. and Williams, B. C., "Mode estimation of probabilistic hybrid systems," *Hybrid Systems: Computation and Control*, Springer, 2002, pp. 253–266.

[52]Blackmore, L., Funiak, S., and Williams, B. C., "A Combined Stochastic and Greedy Hybrid Estimation Capability for Concurrent Hybrid Models with Autonomous Mode Transitions," *Journal of Robotic and Autonomous Systems*, Vol. 56, No. 2, February 2008, pp. 105–129.

[53]Timmons, E. and Williams, B. C., "Enumerating Preferred Solutions to Conditional Simple Temporal Networks Quickly Using Bounding Conflicts," *AAAI Workshop on Planning, Search, and Optimization*, 2015.

[54]Williams, B. C. and Ragno, R. J., "Conflict-directed A* and its role in model-based embedded systems," *Discrete Applied Mathematics*, Vol. 155, No. 12, 2007, pp. 1562–1595.

[55]Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y., "ROS: an open-source Robot Operating System," *ICRA Workshop on Open Source Software*, 2009.

[56]Crick, C., Jay, G., Osentoski, S., and Jenkins, O. C., "ROS and ROSbridge: Roboticists out of the Loop," *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, ACM, Boston, Massachusetts, USA, 2012, pp. 493–494, ISBN: 978-1-4503-1063-5.

[57]Jurić-Kavelj, S., "ROSARIA - ROS Wiki," URL: `http://wiki.ros.org/ROSARIA`, 2014.

[58]Koenig, N. and Howard, A., "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, Vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[59]McGhan, C. L. R., Wang, Y.-S., Colledanchise, M., Vaquero, T., Murray, R., Williams, B., and Ögren, P., "Towards Architecture-wide Analysis, Verification, and Validation for Total System Stability During Goal-Seeking Space Robotics Operations," *AIAA SPACE 2016 Conference and Exposition*, 2016. (Accepted).