15th ICCRTS
”The Evolution of C2”

Plan Maintenance for Continuous Execution Management

Planning, Plan Repair, Execution Monitoring

John McCormick, Nadya Belov, Phil DiBona, and Jeff Patti
Lockheed Martin Advanced Technology Laboratories
3 Executive Campus, Suite 600
Cherry Hill, New Jersey 08002
Email:<jmccormi, nbelov, pdibona, jpatti>@atl.lmco.com

Abstract

Research in the planning community has focused on improving plan generators to produce better, more efficient plans faster. However, the environments in which these plans are usually executed are highly dynamic and even the best plans cannot account for unforeseen circumstances. Additionally, the decision-making processes are still handled by human operators, rendering fully automated approaches to continuous execution management unsuitable. In this paper, we highlight the need for robust continuous execution management capable of bringing together plan generators and monitors to improve dynamic plan maintenance and repair. To that end, we present the Plan Execution Understanding Service (PLEXUS), a continuous plan execution and maintenance system that addresses the challenges outlined above, discuss its role, and contribution on the Joint Air/Ground Operations Unified Adaptive Replanning (JAGUAR) program.

I. Introduction

The Artificial Intelligence (AI) planning community has traditionally focused its research attention on plan generation and, to a lesser extent, plan repair. This approach presumes that plans are static and, once generated, can be executed without encountering unforeseen circumstances. This is not the case for continuous execution management applications, such as air operations, where the plan generator does not have access to the *actors* executing the plan. *Continuous Execution Management* (CEM) is vital in enabling *plan repair* by informing *plan generators* about deviations that occur during *plan execution*. Because deviations can help steer plan repair, plan monitoring and impact assessment are crucial in deviation detection. We believe that plan maintenance contributes a critical capability for continuous execution, connecting *plan generators* with *monitors*, and while the AI community has made great strides toward robust execution monitoring in dynamic environments, dynamic plan maintenance still remains to be fully addressed.

To address the challenges outlined above, we propose a *continuous plan maintenance system* named *Plan Execution Understanding Service* (PLEXUS). PLEXUS provides the ability to splice plan updates into actively executing plans and process updates based on incoming execution observations. PLEXUS has been integrated into the Joint Air/Ground Operations Unified Adaptive Replanning (JAGUAR), a semi-automated system targeted towards oversight and management of a large number of interdependent missions (see § III-B).

The remainder of this paper is laid out as follows: Recent developments and motivation for *Continuous Execution Management* are discussed in § II. A concept of operations is provided in § III to highlight the challenges PLEXUS aims to address followed by an in-depth discussion of our approach in § IV. We present our system's performance based on empirical evaluations in § V and conclude in § VI.

II. Motivation

Until recently, the majority of focus in the AI planning community has centered on plan generation [1]. Plan generation, however, is only a small portion of the overall continuous planning and execution domain. Myers, in [4], attempts to address the challenges associated with continuous planning by developing the Continuous Planning and Execution Framework (CPEF). Specifically, the CPEF is aimed at bringing together the planner and the monitor to enable an execution status feedback loop [4]. While this work provides a valuable contribution toward continuous planning and execution, it is unclear how deviations are assessed from observables or how the CPEF addresses the presence of conditional or branch plans. Others, like Ayan et al., have proposed planning algorithms intended to address plan repair (replanning) in dynamic environments. For example, the Hierarchical Ordered Task Replanning in Dynamic Environments (HOTRiDE) [1] algorithm assesses the dependencies of a failed activity or a mission to maximize the objectives satisfied within a plan. HOTRiDE shows promise, but the algorithm's efficiency in generating and repairing plans in highly dynamic operating environments with many objectives has not been assessed.

The research discussed above highlights the technological strides made in the active plan repair capabilities. Plan generation techniques have matured as a result of decades of focused research. Most plan generators can repair an active mission with a deviation if the extent and specific type of deviation are known. However, the availability of such information is dependent on the type and maturity of execution monitoring component. Bouguerra et. al., in [2], address execution monitoring through the application of planning and semantic knowledge. That is, given a plan to navigate a house; a plan is created that describes each room as having a set of objects in it that help identify the room. Specifically, a kitchen would have a sink and an oven whereas a living room would have a couch or an armchair [2]. In their system, semantic knowledge is used to establish expectations in the monitored environment. While this approach is unique, its success hinges on complete or at least comprehensive knowledge of the environment. Others like Sellner and Simmons [9], propose monitoring of plan execution based on task duration prediction. While task duration may be an effective monitoring heuristic for some domains, it may prove to be insufficient in domains where tasks and missions may be late or unserviceable due to non-temporal constraints. Consider an example where an airborne fuel tanker is expected to refuel a fighter jet; if the tanker experiences an equipment failure and must return to base, the task to refuel the fighter jet will not be late or early, it will not happen at all. § III provides an in-depth concept of operations that further motivates the need for rich semantic knowlege and diverse monitoring heuristics.

III. Concept of Operations

A. Continuous Execution Management

A Continuous Execution Management (CEM) system should provide a structure for integrating planning, execution, and assessment. Figure 1 depicts a conceptual architecture to guide development of CEM systems. The *Planner* provides initial plans and all revisions, due either to new objectives or repair requests. The *Execution* performs the actions defined in the plans to bring about objective satisfaction. The *Monitor* assesses the produced plans, tracking progress and detecting significant divergences, from execution observations. *Plan Maintenance* serves as a coordinating function for the other three components.

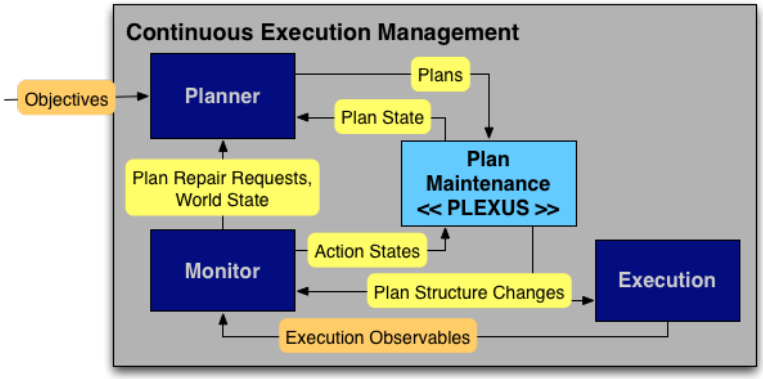

Fig. 1. Continuous Execution Management Concept

Plan Maintenance provides current plan state to the *Planner* whenever new plans or plan repairs are required. Resulting plan revisions are spliced into the actively executing plan and notifications of plan changes are issued to both the *Execution* and *Monitor* components. The *Execution* alters its performance accordingly, and the *Monitor* updates its assessment behaviors to account for the revised expectations. As the *Monitor* assesses changes to action states, *Plan Maintenance* updates and propagates the localized change across the broader plan. The CEM architecture allows for a mix of human and software agent participation in satisfying the *Planner*, *Monitor*, and *Execution* component responsibilities. PLEXUS (see § IV-B) is our solution that addresses the requirements for *Plan Maintenance*.

B. Joint Air/Ground Operations Unified Adaptive Replanning (JAGUAR)

Lockheed Martin Advanced Technology Laboratories (ATL) has completed work with the Defense Advanced Research Projects Agency (DARPA) on the Joint Air/Ground Operations Unified Adaptive Replanning (JAGUAR) program. The developed system employed a model adaptive approach to satisfy the following goals:

1) Generate plans that satisfy a set of objectives
2) Monitor the plans as they execute, identifying deviations
3) Repair plans to correct for detected deviations
4) Adapt models based on changes to operational capabilities

The JAGUAR program aims to augment current United States Air Force mission (re)planning and execution monitoring practices by automating both the planning and monitoring highly complex plans (over 300 concurrent interdependent missions) within the Air Operations Center (AOC). While the U.S. Air Force has very capable tools to support Air Tasking Order production, JAGUAR addresses the dynamic replanning and plan repair challenge for which decision support tools within the Combat Operations cell are less robust. Unlike autonomous control systems [6], the JAGUAR planning and monitoring components are intended to provide decision-support to human AOC operators. An important challenge, therefore, is to maintain synchronization between the operators and automation with regards to the unfolding execution of plans. Because the environment in which JAGUAR is designed to operate is very dynamic, the results of the planning component are considered as desired expectations for comparison with execution observations.

IV. Approach

A. JAGUAR Architecture

JAGUAR, depicted in Figure 2, comprises a set of networked components with well defined functions (services) and interfaces. Communications are achieved using standard publish and subscribe techniques over JMS-based messaging middleware (WebLogic). The functionality of each component has been designed in cooperation and coordination with the other components within JAGUAR, rather than as separate and independent services. Nevertheless, each component's function is known, defined

and can be accessed by external agents adhering to the JAGUAR interface implementation. Within the JAGUAR architecture continuous execution management is primarily divided between the Plan Generation and the Plan Monitoring components.

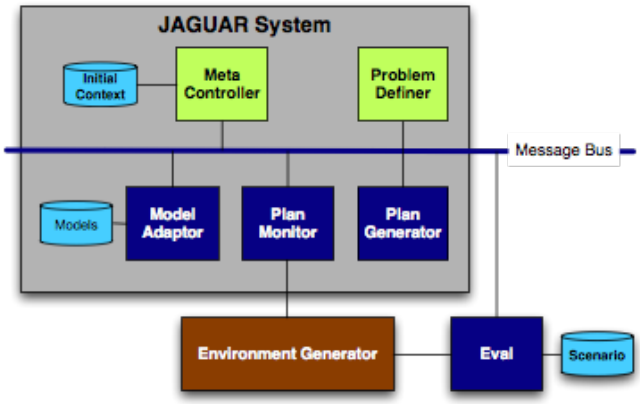

Fig. 2. JAGUAR Architecture

1) Plan Generator: The Plan Generator is responsible for creating the initial plan and for creating repairs to plan deviations assessed and disclosed by the Plan Monitor. The Plan Generator considers available resources, battle objectives and spatiotemporal constraints when creating or repairing a plan. Plans are composed of an action-based Hierarchical Task Network (HTN) [8] [10]. Atomic actions represent the leaf nodes of the HTN. The plans created by the Plan Generator support concurrent actions by an actor, but does not support conditional or branch plans.

2) Plan Monitor: The Plan Monitor is responsible for monitoring the progress of planned missions and for observing the combat environment. The Plan Monitor continuously evaluates observations from ongoing air operations (i.e. datalink reports) in an attempt to identify situations that could jeopardize planned activities. Deviation messages are generated by the Plan Monitor for the Plan Generator in the event of an observation with potential to jeopardize objective satisfaction. The Deviation messages specify the missions and objectives impacted. Furthermore, the Plan Monitor maintains the World and Plan State on behalf of the overall system, and provides a current operational picture to other components upon request.

3) Model Adaptor: The Model Adaptor provides models of systems, missions and processes for all resources available to the CFACC. It provides these models for use by other JAGUAR components. The Model Adaptor uses the plan and deviation messages to assess the existing models. Operator notifications of significant discrepancies between a model and the actual performance of a system, mission or process allow for model revisions for use in subsequent mission planning and monitoring. This process is described in more detail by Mulvehill et. al. [3].

B. Plan Maintenance with PLEXUS

Lockheed Martin ATL designed and developed the Plan Monitoring component of the JAGUAR system leveraging the Interaction Design and Engineering for Advanced Systems (IDEAS) process by combining the best User-Centered Design practices with software development throughout the entire development process (see [7]). The IDEAS process centers on the mediation between the engineers explanation of potential technical capabilities and functional requirements provided by Subject Matter Experts (SMEs). With respect to the JAGUAR system, the Plan Monitor component encompassed both the Monitor and Plan Maintenance functions of the CEM architecture.

PLEXUS is a distributed software component that provides the capability to maintain and reason upon plans, specifically:

- *Action State Maintenance*: tracking actions through their life cycle, activating propagation algorithms on state changes.
- *Plan Structure Maintenance*: splicing new plans and plan repairs into existing structures. A version history of plan structural revisions is also maintained.

PLEXUS interfaces to the *Monitor* and *Planner* CEM components through XML-based plan-related change requests and queries. PLEXUS asynchronously accepts and processes these requests, issuing XML responses to subscribed components. The plan models are domain independent, with the capability to be extended to model domain-specific information. The model employed by PLEXUS is an enhancement of the DARPA Core Plan Representation (CPR) developed by Teknowledge [5]. The CPR enhancements used by PLEXUS were designed to transition CPR for use in plan execution monitoring applications as shown in Figure 3.

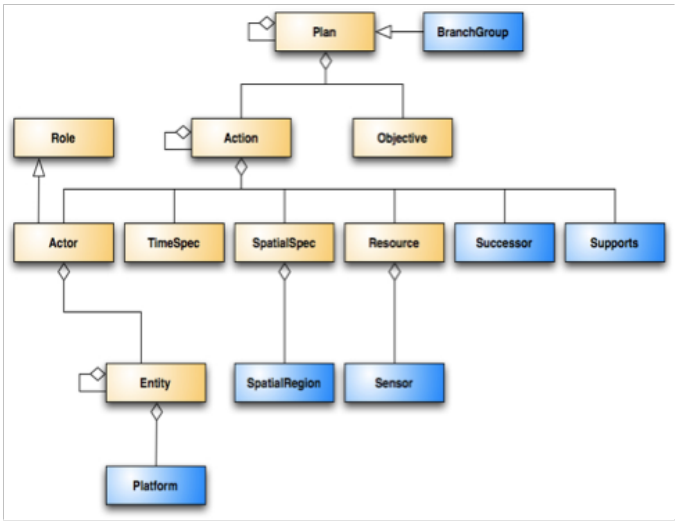

Fig. 3. CPR with enhancements to support execution monitoring

PLEXUS models both the structure and state of plans. The structure of a plan refers to the plan objects (e.g., plans, activities, objectives, entities, roles and resources) and how they are inter-related with each other (e.g., compound actives, has-a relations, causal linkages, interdependencies), as identified in Table I. In operational environments, plans are continually added, modified, and removed by the plan generator. This flux complicates execution monitoring because large and frequent changes can cause unintended inefficiencies inherent in re-tasking monitoring agents or monitoring irrelevant or deleted activities. PLEXUS facilitates monitoring by analyzing changes to plans by looking for orphaned relationships and objects, grouping related objects, and notifying monitoring agents of only those constructs that specifically changed.

TABLE I
PLEXUS PLAN CONSTRUCTS

Term	Type	Definition
Action	Plan Object	A specification of a task in a plan. May specify spatial and temporal constraints. The performing entities (actors) are identified.
Actor	Action - Entity Relation	Specifies the role that an entity will perform in the execution of an action.
Branch Group	Plan Object	Extension of Plan denoting a logical grouping of conditional branches. Each sub-plan in a branch group is conditional. All actions that are immediate children of a branch group are mandatory however. This is an ATL extension to CPR.
Depends-On	Action - Action Relation	Relationship denoting an action dependent on another action for its success. This is the inverse relation of supports.
Entity	Plan Object	An object in the plan environment (i.e. an actor, track, sensor, or resource).
Objective	Plan Object	An intended goal of a plan. It may contain sub-objectives.
Plan	Plan Object	A logical collection of actions, sub-plans, and objectives.
Successor	Action - Action Relation	A causal linkage relationship between actions. A predecessor must succeed and complete in order for the successor to initiate successfully.

Each plan construct is modeled with several required and optional attributes. One of the critical attributes of a plan object is its state, which assists in identifying its context within the plan′s overall execution. The following section defines the lifecycle semantics and state propagation logic implemented in PLEXUS.

1) Life Cycles and State Propagation: PLEXUS allows for a rich representation and reasoning of plan and action state. Atomic and compound actions are supported. Atomic actions must have no sub-actions and may be associated as a child of a single compound action. Compound actions must have one or more sub-actions in their definition, each of which may be

atomic or compound, and may be a child of a single compound action. Table II describes the default set of states supported by PLEXUS. In addition to defining a set of allowable states, PLEXUS supports state lifecycle processing.

TABLE II
PLEXUS STATE DEFINITIONS

Type	State	Definition
Plan	Planned	No actions or sub-plans have begun execution.
	Active	The plan is currently being executed.
	Complete	All actions have finished successfully.
	Realized	All tasks in the plan have finished successfully and all objectives have been met.
	Failed	One or more deviations have occurred in this or a supporting plan.
	Deprecated	This plan is no longer in use.
Action	Planned	Action has not yet commenced.
	Active	The action is currently being executed.
	Interrupted	The action has been interrupted during execution.
	Complete	The action and all sub-actions (if present) have finished successfully.
	Failed	One or more deviations have occurred in this action or a sub-action.
	Deprecated	This action (and all sub-actions if present) is no longer in use.

The lifecycle of atomic action state changes is shown in Figure 4. All actions are created by the Planner and initialized to a *planned* state. Based upon execution observables, the Monitor may change an action state to *active* or *complete*. The Planner may modify an action still in a *planned* state as part of a replan, and PLEXUS will return the action to a *planned* state following the transform operation. Plan repairs involving changes to an active or failed action place it in an *interrupted* state, which is usually intended to have a very short duration before being transitioned to *complete* by the Monitor. Completed and deleted actions are archived by PLEXUS for use by operational analysis components. The lifecycle of compound actions, in which state changes are driven primarily by changes in sub-activity state is depicted in Figure 5. Because all execution related state changes to compound activities are made via propagation, the Monitor has no direct involvement in their lifecycle.

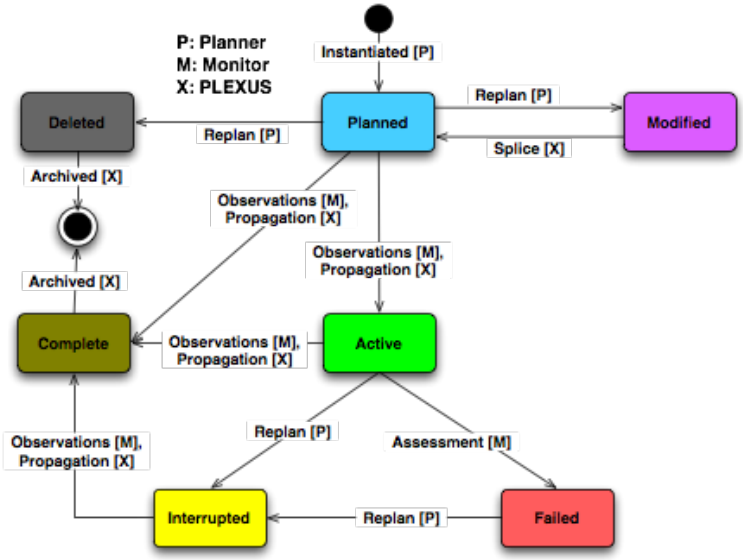

Fig. 4. Atomic action life cycle

As shown in the lifecycle diagrams, the state of the various plan objects can be assigned explicitly by the Monitor or the Planner, as well as implicitly based on the context of the plan. Whenever a new state value is assigned to a plan object, PLEXUS will analyze related objects to determine if any second-order state changes can be inferred based on this new information. This

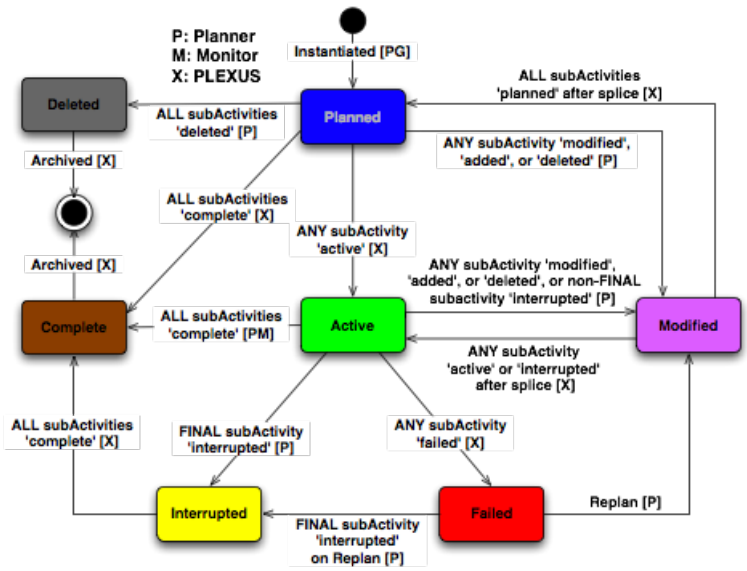

Fig. 5. Compound action life cycle

process, referred to as *state propagation*, will traverse the plan structure and apply a set of state change rules as defined in Tables III and IV. For example, looking at Table IV, an action will transition to a complete state regardless of its current state if a successor action becomes active. The rules are highly dependent on the structure of the plan, and allow domain-specific applications to develop plans whose state propagation behavior can be customized simply by structuring the plan in various ways.

TABLE III
PLAN-LEVEL STATE TRANSITION RULES

Plan State		State Conditions		
Current State	New State	Actions	Sub-Plans	Objectives
Planned	Active	Any Active	Any	Any
Planned	Active	Any	Any Active	Any
Any	Complete	All Complete	All Complete	Any
Complete	Archived	All Complete	All Archived	None
Complete	Archived	All Complete	All Archived	All Met
Any	Failed	Any	Any	Any Not Met
Any	Failed	Any Failed	Any	Any
Any	Failed	Any	Any Failed	Any
Failed	Planned	None Failed	None Failed	None Not Met

TABLE IV
ACTION-LEVEL STATE TRANSITION RULES

Action State		State Conditions		
Current State	New State	Sub-Actions	Successor	Depends-On
Planned	Active	Any Active	Not Active or Complete	Any
Any	Complete	All Complete	Any	Any
Any	Complete	Any	Active	Any
Any	Complete	Any	Complete	Any
Any	Failed	Any Failed	Any	Any
Any	Failed	Any	Any	Any Failed
Failed	Planned	None Failed	Any	None Failed

2) Plan Transformations: While structural changes to plans may only be performed by the Planner and execution state progression is determined by the Monitor, re-plans or plan repairs can involve structural changes to plan elements currently in execution. PLEXUS permits the following set of actor-centric operations to be performed by the Planner:

- Instantiate plan for a new actor
- Remove an actor′s plan entirely
- Full replacement of an actor′s plan
- Modification of an actor′s plan
- Interruption of an actor′s plan

The instantiation, removal, and full replacement operations are straightforward, with the primary validation check ensuring the plan to be removed or replaced has not initiated execution. The plan modification operation allows the Planner to change future actions of a plan that are either *planned* or *active*. An example scenario for this operation would be tasking an air alert while the actor is still en-route to the orbit, or has not taken off yet. The most challenging plan transformation within our Continuous Execution Monitoring system involves activity interruption. This operation allows the Planner to interrupt a current action and change future actions of an actor′s plan. An example scenario where this operation would be used is tasking an air alert that is currently in orbit. Figure 6 depicts an initial condition for such a scenario.

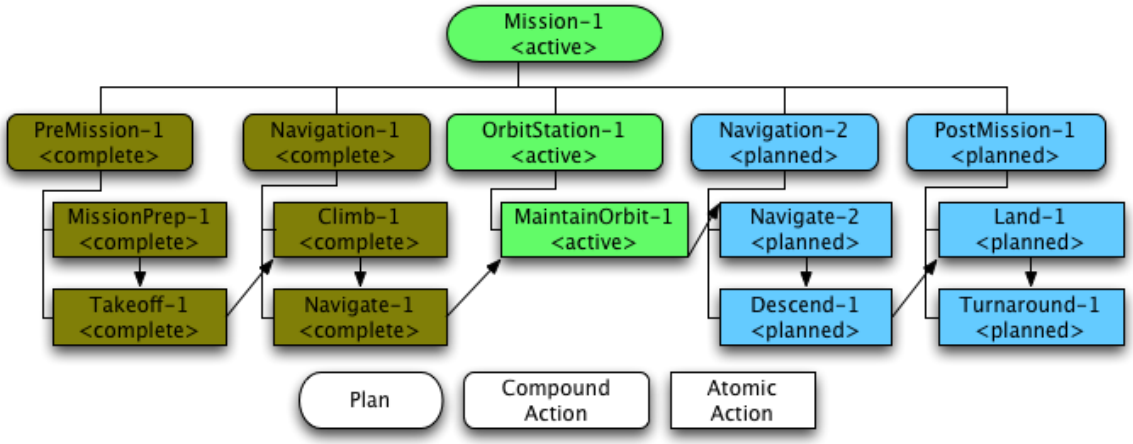

Fig. 6. Example mission prior to an interrupt

The modification and interruption operations require understanding and adherence to action lifecycle transition rules to achieve the desired results shown in Figure 7. Of particular note are plan repair related transitions restrictions on the Planner. During a replan, if the Planner adds, modifies or deletes any sub-action, the compound action state must be set to *modified*. Similar to atomic actions, PLEXUS will set the compound action back to a *planned* or *active* state after performing the modifications during the splice based upon the sub-action state(s). If during a replan, the final sub-action is set to an *interrupted* state, then the compound action is also set to *interrupted*. The interruption of any other sub-action places the compound action into a *modified* state.

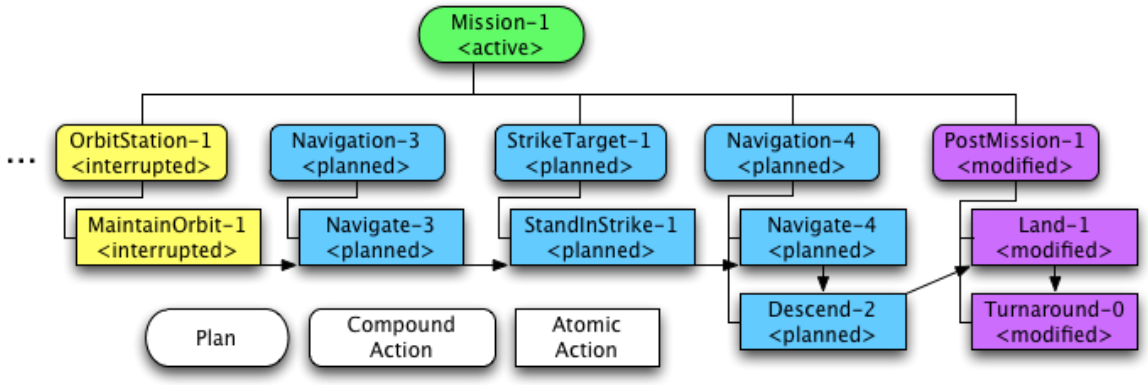

Fig. 7. Example mission following the interrupt

V. Performance Analysis

The evaluation of any system is not to be overlooked. As mentioned in Section III, the goal of this system is to provide plan maintenance for continuous execution management for highly complex plans (over 300 concurrent interdependent missions).

This section discusses our evaluation of the system and its ability to perform continuous plan maintenance and execution management.

A. Approach and Setup

The evaluation involved performing continuous plan maintenance and execution management on small, 54 mission, and large, 450 mission, plans. The missions of each plan are interdependent, meaning that the actors rely on each other for mission success. In addition to actors, JAGUAR plans also contain activities, entities, and spatial specifications. The number of tracked items for each plan is presented in Table V.

TABLE V
THE NUMBER OF ACTORS, ACTIVITIES, ENTITIES, AND SPATIAL SPECIFICATIONS IN EACH TEST PLAN

	Plan 1	Plan 2
Actors	54	450
Activities	1098	7925
Entities	1520	13228
Spatial Specs	1511	10565
Total	4183	32168

The evaluation tests stressed the system with real data collected throughout the life of the JAGUAR program.

- Setup - Initialization of the world state
- Problem Statement - Establishment of mission goals/objectives that the planning component must attempt to satisfy
- Context Request - Assessment and reporting of world and plan state, upon JAGUAR component requests
- Time Jump - Advancement of the virtual clock to force the plan monitor to produce activity state updates, a JAGUAR specific feature for speeding up testing

For each test, we performed several runs to determine an average execution time for each of these steps. The specific order of operations performed for each test was setup, context request, problem statement, plan, context request, and time jump. The evaluation was conducted on a Dell PowerEdge 1900, which has a quad-core Xeon 2.33GHz processor.

B. Results

Table VI shows the computed average execution time for each step and each plan. Details of the time to complete each step are provided in Tables VII and VIII of Appendix A. Based on the average case analysis shown in Table VI we conclude that PLEXUS satisfies the plan maintenance requirements for continuous execution management of plans at the scale and complexity required for air operations.

TABLE VI
EXECUTION TIME OF EACH TEST STEP ON EACH PLAN IN SECONDS

	Plan 1 Average	Plan 1 Std. Dev.	Plan 2 Average	Plan 2 Std. Dev.
Setup	34.3s	2.8s	44.5s	0.7s
Context Request	4.7s	0.5s	6.0s	0.0s
Problem Statement	5.3s	0.5s	8.0s	0.0s
Plan	101.2s	17.2s	4616.5s	47.4s
Context Request	14.5s	0.7s	159.5s	6.4s
Time Jump	44.2s	17.9s	1108.5s	23.3s
Total	204.2s	19.9s	5943.0s	77.8s

VI. DISCUSSION AND CONCLUDING REMARKS

This paper highlights the need and the challenges presented by continuous execution management. In an effort to address the challenges presented by a rapidly changing, unpredictable environment, we described a plan maintenance service, called PLEXUS, which supports a CEM framework. Using extensions to the well-established Core Plan Representation, PLEXUS tracks and enforces domain-independent action lifecycle rules. Most importantly, the system supports the integration of a plans structural changes necessary to satisfy new objectives as a result of repairs. The loosely coupled architecture facilitates full human-in-the loop operational oversight and decision support.

Preliminary testing using operational and realistically-scaled scenarios demonstrated the agility and robustness PLEXUS and the overall Plan Monitoring component bring to continuous execution.

ACKNOWLEDGMENTS

This work was conducted as a part of the JAGUAR project, sponsored by the Defense Advanced Research Projects Agency (DARPA) through DARPA Award FA8750-04-C-0004, with additional support from the Lockheed Martin Corporation. The authors wish to thank the PLEXUS development team, Jim Allen, Tim Thayer, and Frank Vetesi.

REFERENCES

[1] N. Fazil Ayan, Ugur Kuter, Fasun Yaman, and Robert P. Goldman, *HOTRiDE: Heirarchical Ordered Task Replanning in Dynamic Environments*, Proceedings of ICAPS'07 3rd Workshop on Planning and Plan Execution for Real-World Systems (Providence, Rhode Island), September 2007.

[2] Abdelbaki Bouguerra, Lars Karlsson, and Alessandro Saffiotti, *Active Execution Monitoring Using Planning and Semantic Knowledge*, Proceedings of ICAPS'07 3rd Workshop on Planning and Plan Execution for Real-World Systems (Providence, Rhode Island), September 2007.

[3] Alice M. Mulvehill, Brian Krisler, and Renu Bostwick, *Deriving Reliable Model Revisions from. Executed Plan Data Analysis*, Proceedings of the 14th International Command and Control Research and Technology Symposium (Washington D.C.), June 2009.

[4] Karen Myers, *Towards a framework for continuous planning and execution*, AI Magazine **20** (1998), no. 4, 13–22.

[5] Adam Pease and Todd M. Carrico, *JTF-ATD Core Plan Representation: A Progress Report*, Proceedings of the AAAI Spring Symposium on Ontological Engineering, 1997, pp. 95–100.

[6] Ola Pettersson, *Execution monitoring in robotics: A survey*, Robotics and Autonomous Systems **53** (2005), no. 2, 73–88.

[7] Susan Regli and Patrice Tremoulet, *IDEAS: Interaction Design and Engineering for Advanced Systems*, American Psychological Association, Division 21, a New Collaborative Frontier:Innovative Approaches and Applications (Fairfax, Virginia), 2007.

[8] Earl D. Sacerdoti, *The nonlinear nature of plans*, IJCAI'75: Proceedings of the 4th international joint conference on Artificial intelligence (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 1975, pp. 206–214.

[9] Brennan Sellner and Reid G. Simmons, *Duration prediction for proactive replanning*, ICRA, IEEE, 2008, pp. 1365–1371.

[10] Austin Tate, *Generating project networks*, IJCAI'77: Proceedings of the 5th international joint conference on Artificial intelligence (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 1977, pp. 888–893.

APPENDIX

TABLE VII
EXECUTION TIME OF EACH TEST STEP ON THE SMALLER PLAN FOR EACH RUN IN SECONDS

Run #	1	2	3	4	5	6	7	8	9	10	Mean	Std Dev
Setup	31	30	35	33	34	33	34	37	38	38	34.3	2.75
Context Request	5	4	5	5	5	4	5	4	5	5	4.7	0.48
Problem Statement	5	5	6	6	5	5	5	5	6	5	5.3	0.48
Plan	95	150	97	95	96	94	94	99	96	96	101.2	17.21
Context Request	14	15	14	15	15	14	15	15	13	15	14.5	0.71
Time Jump	29	24	30	45	67	29	60	38	44	76	44.2	17.92
Total	179	228	187	199	222	179	213	198	202	235	204.2	19.85

TABLE VIII
EXECUTION TIME OF EACH TEST STEP ON THE LARGER PLAN FOR EACH RUN IN SECONDS

Run #	1	2	Mean	Std Dev
Setup	44	45	44.5	0.71
Context Request	6	6	6.0	0.00
Problem Statement	8	8	8.0	0.00
Plan	4583	4650	4616.5	47.38
Context Request	155	164	159.5	6.36
Time Jump	1092	1125	1108.5	23.33
Total	5888	5998	5943.0	77.78