Towards a Framework for Continuous Planning and Execution

Karen L. Myers
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com

Abstract

This paper reports on the first phase of the *Continuous Planning and Execution Framework* (CPEF), a system that employs sophisticated plan generation, execution, monitoring, and repair capabilities to solve complex tasks in unpredictable and dynamic environments. CPEF embraces the philosophy that plans are dynamic, open-ended artifacts that must evolve in response to an ever-changing environment. In particular, plans and activities are updated in response to new information and requirements to ensure that they remain viable and relevant. Users are an integral part of the process, providing input that influences plan generation, repair, and overall system control. CPEF has been applied successfully to generate, execute, and repair complex plans for gaining and maintaining air superiority within a simulated operating environment.

Introduction

The AI planning community, until recently, has focused its attention almost exclusively on the problem of *generation*: producing a schedule of activities that when performed in some initial state will guarantee achievement of a specified set of goals. With the exception of plan repair, important topics related to the *use* of plans (robust execution, reactivity, monitoring, evaluation) have received significantly less consideration. In realistic domains, however, plan generation is only a small component of the overall package.

This paper reports on the first phase of an effort to develop a system, the *Continuous Planning and Execution Framework* (CPEF), that combines sophisticated plan generation and plan use capabilities to solve complex tasks in unpredictable and dynamic environments. CPEF embraces the philosophy that plans are dynamic, open-ended artifacts that must evolve in response to an ever-changing environment. In particular, plans must be updated in response to new information and requirements in a timely fashion to ensure that they remain viable and relevant. Plan execution involves more than blind adherence to previously generated plans. Rather, runtime decisions are made to adapt, activate, or abandon plans and activities in response to current considerations within the operating environment.

To date, the emphasis for CPEF has been to produce a distributed, multiagent framework in which plan generation and execution are fluidly integrated. The system provides timely adaptation of its activities based on monitoring of critical events within its operating environment. Users are an integral part of the overall process, providing input that will influence the types of plans that are generated, the number of options to consider, failure assessments, plan repair strategies, and overall control of system behavior.

One unique characteristic of CPEF is that it supports both *direct* execution, in which activities and actions are undertaken by the system itself, and *indirect* execution, in which the system supervises execution of plans by a collection of distributed execution entities. The indirect model of execution is essential for many domains, including workflow management and many classes of military operations, where software control of plan entities is impossible.

CPEF leverages several sophisticated AI technologies as components. SIPE–2 (Wilkins 1988) provides *hierarchical task network* (HTN) planning and plan repair capabilities. The Advisable Planner (AP) (Myers 1996) supports user provision of advice to guide the process of plan generation. The Procedural Reasoning System (PRS) (Georgeff & Ingrand 1989; Myers 1993), a reactive plan execution system that integrates goal-oriented and event-driven activity in a flexible hierarchical framework, is used both as an executor for plans, and a high-level controller for the overall system. Additionally, CPEF builds on aspects of the Multiagent Planning Architecture (Wilkins & Myers 1998), primarily to provide distributed communication and plan storage services.

CPEF draws upon experience gained in building Cypress (Wilkins *et al.* 1995), an integrated planning and execution system that also employed SIPE–2 as a generative planner and PRS as an executor. Like Cypress, CPEF employs a common *procedure library* (encompassing both plans and operators) encoded in the Act representation language (Wilkins & Myers 1995). Elements of the library span multiple abstraction levels and are usable for both plan generation and execution, thus supporting smooth transitions between the two capabilities. In particular, plan generation can proceed to arbitrary levels of refinement, with the executor applying additional procedures at runtime to refine tasks to executable activities. As with Cypress, planning and execution operate asynchronously within CPEF, in

a loosely coupled fashion. CPEF components communicate domain knowledge, plans, requests, and situation information as required to fulfill their respective responsibilities.

CPEF differs from Cypress in several key ways. First, rather than leaving control of the overall system implicit in the activities of the execution module, an explicit *Plan Manager* oversees activity within the system. Second, CPEF supports a much richer set of interactions between the planner and executor, as well as a broader set of plan adaptation and repair mechanisms. Third, Cypress did not support *indirect* plan execution, thus limiting its applicability to domains in which planned actions could be executed and monitored by the system itself. Finally, the grounding of CPEF in a powerful multiagent architecture (MPA) enables distributed operations, which were not directly supported within Cypress.

CPEF is being developed within the context of supporting a Joint Forces Air Component Commander (JFACC) in the execution of realistic air campaigns. This document describes one demonstration that illustrates the ability of CPEF to generate, execute, and repair complex air campaign plans while remaining responsive to changes in guidance and tasking.

CPEF Architecture

Figure 1 provides an overview of the CPEF system. Boxes in the figure represent key functional capabilities, while arrows depict information flow. Attached to each box is the name of one or more technologies – the Advisable Planner (AP), the Procedural Reasoning System (PRS), SIPE–2 – that implement the associated functionality.

CPEF Components

The Plan Manager lies at the heart of the system, directing the overall plan generation, monitoring, and execution processes. The Plan Manager is always active, continuously monitoring the world for new tasks and information to which the system should respond.

The Planner provides core plan generation and adaptation capabilities, ranging from fully automated, to interactive and *advisable* planning in which users can express recommendations and preferences on the types of plans that are to be produced. Advisable planning is valuable both as a way of supporting user customizability of generated plans, and for enabling user-directed exploration of qualitatively different options.

Plan Repair oversees adaptations of plans in response to situation changes and execution results. The Simulator serves as a stand-in for the real-world execution of a plan.

The Plan Server provides a repository for storing multiple plans in an organized and principled fashion. While of limited use within the current system, we envision the Plan Server as essential for managing the large numbers of options and subplans that will be required to support long-term continuous planning.

The Interface module supports interactions between the user and CPEF. Users can supply a range of information and requests to the system including assignment of tasks,

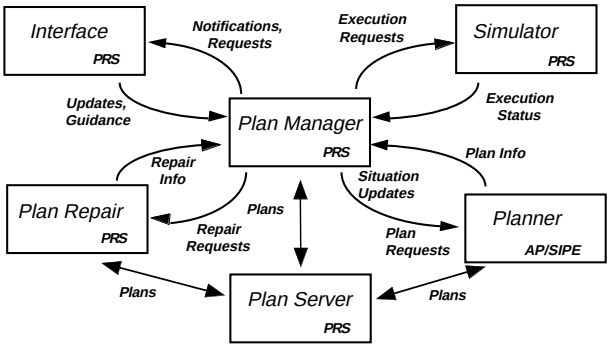

Figure 1: Functional Overview of CPEF

situational updates, evaluation assessments, and planning advice. The system informs the user of critical events and activities, soliciting guidance when appropriate to direct problem-solving.

Agent-Based Organization

CPEF incorporates portions of the Multiagent Planning Architecture (MPA) (Wilkins & Myers 1998) into its infrastructure. MPA is an agent framework that provides a collection of services and capabilities designed to facilitate the management of complex, distributed planning tasks. Each of the functional modules in Figure 1 is encapsulated as a single MPA agent within CPEF.

CPEF makes use of two components of MPA, namely its communication mechanisms and its Plan Server. MPA communication consists of a set of message protocols layered on top of KQML (Finin *et al.* 1992). The protocols define a language for exchanging information and requests related to plans and planning activities.

MPA was developed originally to provide general agent services for plan construction and evaluation tasks. In order for MPA to be used as the underlying agent infrastructure for CPEF, two key extensions were required. First, MPA protocols were defined to support the exchange of requests and results related to plan repair operations. Second, the conceptual framework was extended to include a class of *executor* agents along with corresponding communication protocols.

Plan Manager

The *Plan Manager* is responsible for the overall control of system operation. As such, its main responsibilities are

- to control generation of plans and options for outstanding tasks,
- to oversee execution of plans,
- to provide knowledge management capabilities for plans and plan execution (e.g., monitor for key events, perform information-gathering tasks),

- to provide timely response to user requests and unexpected events,
- to control adaptation of plans in response to plan failures.

The Plan Manager can execute multiple threads of activity at any given time. This multiprocessing enables, for example, one or more secondary plans to be activated in response to unexpected events in addition to the primary plan being executed (e.g., dispatch of a Search and Rescue mission to recover a downed pilot during execution of the main air campaign plan).

Direct vs. Indirect Execution

Previous work on reactive execution of plans has focused on models in which an executor *directly* performs the activities specified in a plan. For example, the software controller for a mobile robot would initiate actual execution of actions by the robot (e.g., turning, increasing speed, or stopping) in the physical (or simulated) world.

This direct model of execution is inappropriate for the JFACC domain since the actions in JFACC plans must be performed by the pilots, soldiers, marines, and support staff who are involved with the campaign. Rather, the JFACC operating environment requires an *indirect* model of execution, in which plan activities are performed by human agents in the real world rather than a software controller. Within this indirect model, the role of an executor is to *track* the execution of the plan rather than to carry out the actions. Tracking involves monitoring progress through the execution of the plan based on information (possibly incomplete) about the outcomes of individual actions within the plan.

While a seemingly subtle distinction, the difference between *direct* and *indirect* execution greatly impacts the design of an executor. With indirect execution, an executor may not have direct access to information about the success or failure of prescribed actions, or may not be able to determine whether those actions ever took place. Even when information is available, there may be a significant time lag between performance of an action and receipt of information about its status. Similarly, there may be substantial delays and cost in redirecting activities of the agents that are performing the actions in the plan.

The Plan Manager employs a *flow model* for tracking plan execution. This model involves waiting for reports on the outcome (*success*, *failure*, or *unknown*) of individual actions, in accordance with the temporal ordering relationships of actions in the plan. For example, if action A_1 precedes action A_2, the flow model dictates that the outcome of A_1 must be determined and appropriate responses taken before the outcome of A_2 can be considered. This tracking mechanism was implemented as a variant of the standard hierarchical task execution mechanisms within PRS. In particular, specialized methods were defined for action achievement and condition testing that implement tracking rather than execution semantics. In the future, we intend to explore more opportunistic tracking models that enable response to action outcomes in arbitrary orders.

Monitors

The creation and deployment of *monitors* is a critical part of CPEF. We define a monitor to be an *event-response* rule for which detection of the specified *event* leads to execution of the designated *response*.

We have begun development of a taxonomy of monitor classes that will enable appropriate measured responses to critical detected events. The different classes derive from variations in both the types of events to be detected, and the nature of the responses. We believe that this classification will enable simpler and more modular specifications of monitors, for both automated and interactive approaches.

The main categories of concern to date have been *failure*, *knowledge*, and *assumption* monitors:

Failure Monitors encode appropriate responses to failures that could occur during execution of a plan.

Knowledge Monitors test for the availability of information about a world-state condition that is needed for decision-making.

Assumption Monitors test for situation changes that violate assumptions upon which a given plan relies.

Assumption monitors are particularly valuable in that they enable the detection of potential problems with a plan ahead of time, rather than waiting for the problems to surface during plan execution.

CPEF supports user definition of a wide range of monitors. Additionally, certain kinds of monitors are generated automatically based on the content of a plan. In particular, we have developed and implemented a technique for the automated generation of assumption monitors from HTN plan derivation structures.

The algorithm for extracting assumption monitors involves a traversal of the HTN plan derivation structures, collecting from each node those operator applicability conditions that are *dynamic* (e.g., troop movements, but not geographic conditions) and are not included in the effects of some preceding node in the plan (i.e., they must be satisfied in the initial world). The latter type of filtering eliminates large numbers of conditions that should not be monitored, since they are to be established by actions within the plan. Prespecified domain models identify a *response* to be performed when the applicability conditions are violated. Currently, there are three categories of responses: *alerts* for the user, *plan repairs*, and invocation of *standard procedures*. Additionally, the domain models indicate conditions for which assumption monitors are definable, thus filtering conditions whose violation is not significant. For example, weather conditions may fluctuate over time; their status can be disregarded until entry into a critical time window preceding key actions.

Failures and Repairs

Within CPEF, the Plan Manager determines when to initiate modifications to a plan. In contrast to many current systems, individual failures do not necessarily lead to plan repair. Rather, the Plan Manager supports a variety of models for interpreting failures and responding.

Generalized Failure Models

Within the AI community, models for detecting and recovering from plan execution failures have generally been limited to *precondition failures* and *action failures*. A *precondition failure* arises when associated preconditions for an action are not satisfied at the time the action is to be executed. An *action failure* results when the execution of an action does not attain its intended effects. These two types of failures, while important, cover only a small portion of the space of possible failures. In our initial CPEF system, we have taken a first step towards a more general framework that includes the following two types.

Unattributable Failures Failures are called *unattributable* if no individual action has failed or no assumption is violated, yet some assessment (human or automated) has deemed the current plan inadequate. For example, a commander may declare that a planned breach of enemy IADS has failed, despite the success of each constituent mission. Such a situation can arise either because the planning operators do not model the real world with sufficient fidelity, or simply because the commander has a conservative nature (e.g., he requires a high guarantee of neutralization before he is willing to fly subsequent missions through a sector).

Aggregate Failures In many situations, a single failure need not be cause for alarm. Indeed, good human planners often build redundancies into their plans to improve robustness. As a concrete illustration, Air Campaign plans often include extra missions above and beyond what is required to satisfy the objectives at hand in order to improve the likelihood of success.

Detection of unattributable and aggregate failures requires information beyond what is stored in plan dependency structures. Within CPEF currently, unattributable failures are identified by human assessors, while a simple theory of aggregate failures has been defined for Air Campaign plans.

Plan Repairs

Our application domain (like many others) requires the use of *conservative* repairs (Nebel & Koehler 1995) that minimize changes to the original plan. Plans should evolve gradually, with small changes in the world or current goals resulting in proportionally small changes to the plan. Minimization of changes is important to ensure the continuity of the plan, and because of the potentially high costs of redirecting execution entities. To date, our focus has been on conservative repair based on analysis of plan dependency structures (Wilkins 1985; Kambhampati & Hendler 1992). In particular, we rely on the core methods defined previously within SIPE–2, along with several extensions that support more flexible forms of plan repair.

Generally speaking, plan repair based on dependency structure analysis involves identifying a set of *root nodes* that are the source of failures. Each such root has an associated *wedge* of lower-level tasks, which are removed from the plan. New subplans are then generated for each root node, if possible; otherwise, the process repeats for the parent of that node, terminating when the generation process succeeds. To support the repair of unattributable failures, we have implemented a method that enables users to identify arbitrary root nodes whose wedges are to be replaced.

A second extension to the methods in SIPE–2 enables replanning for *task generator* nodes. A task generator node differs from standard tasks within a plan in that it spawns a set of instances of a *task template* rather than a single task instance. The set of instances is determined by a special *creation condition*: an instance of the task template is created for each set of bindings that satisfies the creation condition. Generator nodes provide a powerful representational capability that is critical for planning in many realistic domains. To support plan repair, generated tasks whose creation conditions are violated are identified and removed from a plan. Furthermore, situation changes that result in the satisfaction of additional instances of the creation conditions lead to insertion of corresponding generated tasks into the plan.

The replanning capabilities within CPEF represent a start towards more flexible plan adaptation mechanisms. More work is required to produce the kind of general plan repair framework envisioned for the final CPEF system. Methods grounded in the analysis of dependency structures produce a plan that is proven correct with respect to the underlying domain model; here, correctness means that simulated execution of the plan will result in a world state where the original goals are satisfied. Even though this notion of correctness is somewhat weak (since unexpected events generally will occur, and the domain knowledge itself may be faulty), guarantees of correctness can be computationally expensive to secure. For this reason, a continuous planning system should ideally provide a spectrum of plan repair mechanisms ranging from the correct but costly minimal-perturbation, dependency structure methods to transformational approaches that employ domain-specific transformation rules (in the spirit of (Ambite & Knoblock 1997)), possibly trading correctness for efficiency.

Simulation Environment

The JFACC application domain precludes evaluation of CPEF in an actual operational setting. Furthermore, no appropriate simulation environment exists in which to conduct experiments. For these reasons, we have developed a PRS-based simulation environment called *SImulated FLexible EXecution* (SIMFLEX) to enable testing, evaluation, and demonstration of the continuous planning and execution capabilities of CPEF.

SIMFLEX provides a 'zero-fidelity' simulation capability that neither requires elaborate domain-specific action models, nor tracks state information in the simulated world. The only required input to SIMFLEX is a syntactic description of the possible actions and conditions for a given domain (i.e., a specification of the action names and world-state conditions and their associated argument lists).

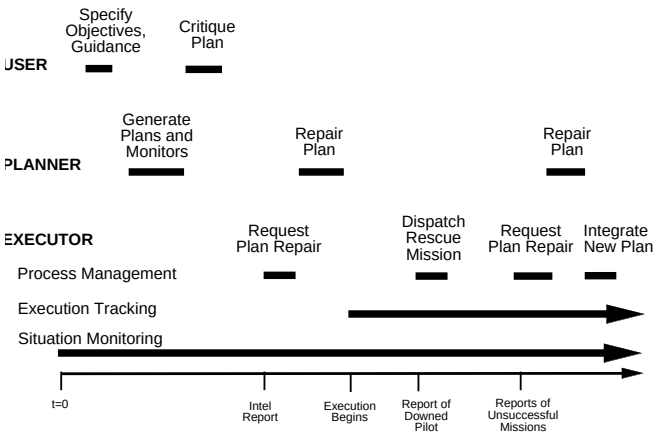

Figure 2: Demonstration Timeline

For each possible action, SIMFLEX generates an Act for simulating the execution of the action. These Acts are parameterized, thus enabling runtime-modifiable outcomes for the actions in the plan. To simulate a given plan, SIMFLEX traverses through its nodes, invoking the generated Acts using the standard task refinement methods within PRS. By building upon the infrastructure of PRS in this manner, development of the simulation environment required relatively little effort.

User-specifiable distributions determine the rates at which actions or condition tests succeed, fail, or yield no information about their execution, as well as the duration of actions. Additionally, users can specify overriding success and duration rates for individual actions, conditions, and tasks. As such, SIMFLEX provides a flexible, tailorable environment in which to perform extensive evaluation of continuous planning capabilities.

CPEF Application

CPEF has been applied to an Air Campaign Planning (ACP) domain (Lee 1998), with emphasis on achieving air superiority within a designated region. The plans in this domain are derived through hierarchical refinement of objectives for defensive and offensive air superiority, terminating at the level of the Air Tasking Order (ATO). The final plans, which require less than a minute to generate, contain several thousand nodes and can span up to 25 refinement levels.

Extensive documentation for the demonstration system is available at the URL `http://www.ai.sri.com/~cpef/jfacc.html`, including operating instructions, detailed descriptions of the capabilities of the system, and sample plans. Here, we provide a brief overview of key concepts and contributions.

Figure 2 provides a high-level overview of one run of the system. The timeline along the bottom shows exogenous events that lie outside of the system's control. Above that, activities are organized along three functional roles: the User, the Planner (encompassing plan generation and plan repair), and the Plan Manager. The system is designed so that the User plays an active role in the plan development and execution processes. The term Planner is used generically to refer to a number of planning-related activities: plan generation, plan analysis, and plan repair. The executor is active at all times, providing three main threads of activity in parallel: *situation monitoring*, *execution tracking*, and *process management*.

Activity begins in response to the user specifying air objectives for a given campaign, along with advice that reflects the commander's guidance for one or more courses of action to be developed. The Planner generates one or more plans to satisfy those objectives and advice, relative to its current knowledge of the operating environment. The completed plans are reviewed by the User, who can recommend changes to satisfy any outstanding concerns; the Planner then produces updated plans that incorporate the User's feedback. (Alternatively, the User could request a completely different plan through the specification of a new set of advice.) As planning proceeds, the Plan Manager monitors the environment for events that are relevant to the developing plan. Receipt of an updated weather report that invalidates parts of the plan will prompt a request that the Planner repair affected portions of the plan. Receipt of an intelligence update that invalidates parts of the plan will prompt a request that the Planner repair the affected portions of the plan.

Monitoring of the environment continues as execution of a selected plan commences. At some point during the execution, notification is received that a pilot has been downed; the system responds by instigating an appropriate activity (e.g., a Search and Rescue mission). In addition to monitoring for critical events of this type, the Plan Manager tracks progress through the execution of the plan to determine whether modifications are needed in response to the status of the execution. As an illustration, one generated CPEF plan contains a set of missions to neutralize a set of SAM sites, as a way of enabling access to a critical air sector. Receipt of reports indicating that more than a designated threshold of missions failed (i.e., an aggregate failure) triggers replanning to address the failure, leading to either the establishment of a second neutralization mission, or changes in the overall plan that eliminate the need for access to the air sector.

Conclusions

CPEF, while a work in progress, already provides many of the foundational capabilities required for continuous planning and execution in highly dynamic and complex worlds. These capabilities include an agent-based architecture, rich monitoring and repair strategies, flexible integration of plan generation and execution, and highly adaptive problem-solving capabilities. Much more is required, however, to produce a truly continuous planning and execution system. Several research problems that we intend to address in the near future as a way of moving us towards that objective are summarized below.

Open-ended Planning Continuous operation requires the ability to produce open-ended plans that grow and evolve in response to the dynamics of the environment. CPEF currently relies on traditional planning methods that create end-to-end solutions. We intend to explore methods for generating plans whose abstraction depth and temporal extent are grounded in the knowledge and constraints of the current situation. Incremental planning techniques will be required to enable the growth and evolution of open-ended plans in response to situation changes.

Options Management Contingency planning provides conditionalized plans that accommodate low-level variations in the operating environment (Pryor & Collins 1996). However, for domains where failures have substantial impact, *a priori* options should be generated to accommodate significant changes in operating assumptions. We are interested in developing methods that can prepare for such high-level dynamism, through the generation of different options that provide coverage for relevant eventualities. The key problem is to determine what constitutes adequate coverage for a given problem-solving space, given the dynamics of the environment.

User Involvement Within CPEF currently, user involvement is limited to the provision of advice for plan development, situational updates, and assessment data. The system informs the user of certain conditions (failures, critical events) and solicits limited recommendations. More generally, we would like to build a framework in which user interactions enable much broader control over system behavior, and plans and process are communicated more effectively to the user.

Acknowledgements

This research was supported by DARPA Contract F30602-97-C-0067, under the supervision of Air Force Research Lab – Rome. David Blei, Tom Lee, and David Wilkins have contributed towards the development of CPEF.

References

Ambite, J. L., and Knoblock, C. A. 1997. Planning by rewriting: Efficiently generating high-quality plans. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. AAAI Press.

Finin, T.; Weber, J.; Wiederhold, G.; Genesereth, M.; Fritzson, R.; McKay, D.; and McGuire, J. 1992. Specification of the KQML Agent-Communication Language. Technical Report EIT T R92-04, Enterprise Integration Technologies, Palo Alto, CA.

Georgeff, M. P., and Ingrand, F. F. 1989. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*.

Kambhampati, S., and Hendler, J. 1992. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence* 55(2):192–258.

Lee, T. J. 1998. The air campaign planning knowledge base. Technical report, Advanced Automation Technology Center, Menlo Park, CA.

Myers, K. L. 1993. *User's Guide for the Procedural Reasoning System*. Artificial Intelligence Center, SRI International, Menlo Park, CA.

Myers, K. L. 1996. Strategic advice for hierarchical planners. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*. Morgan Kaufmann Publishers.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence* 76:427–454.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research* 4:287–339.

Wilkins, D. E., and Myers, K. L. 1995. A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation* 5(6):731–761.

Wilkins, D. E., and Myers, K. L. 1998. A multiagent planning architecture. In *Proceedings of the Fourth International Conference on AI Planning Systems*.

Wilkins, D. E.; Myers, K. L.; Lowrance, J. D.; and Wesley, L. P. 1995. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI* 7(1):197–227.

Wilkins, D. E. 1985. Recovering from execution errors in SIPE. *Computational Intelligence* 1(1):33–45.

Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann.