Adjustable Autonomy for a Plan Management Agent

Martha E. Pollack[†,*]
Ioannis Tsamardinos[*]
[†]Department of Computer Science
[*]Intelligent Systems Program and
University of Pittsburgh
Pittsburgh, PA 15260 USA
{pollack,tsamard}@cs.pitt.edu

John F. Horty
Institute for Advanced Computer Studies
and Department of Philosophy
University of Maryland
College Park, MD 20742
horty@umiacs.umd.edu

Abstract

The Plan Management Agent (PMA) is an intelligent software system that is intended to aid a user in managing a potentially large and complex set of plans. Currently under development, PMA applies AI technology for modeling and reasoning about plans and processes to the development of automated support for work activities. We have developed and implemented algorithms for reasoning about richly expressive plans, which include explicit temporal constraints, temporal uncertainty, and observation actions and conditional branches. We have also developed and implemented an approach to computing the cost of a new plan in the context of existing commitments. The current version of PMA has a low level of autonomy: it makes suggestions to its user, but it does not directly act on her behalf. In this paper, we first describe the PMA system, and then briefly raise some design questions we will need to address as we increase the level of PMA's autonomy, and have it vary with the situation.

Introduction

The Plan Management Agent (PMA) is an intelligent software system that is intended to aid a user in managing a potentially large and complex set of plans. Previous research in Artificial Intelligence—especially in the subfields of plan generation and plan execution systems—has led to the development of rich representations for modeling structured processes, as well as algorithms for reasoning about such processes. Systems employing these representations and algorithms have been used in a variety of applications, including equipment malfunction handling (the PRS system) (Georgeff & Ingrand 1989), air traffic control (dMars) (Rao & Georgeff 1995), robot navigation (RAPs) (Firby 1996), and autonomous spacecraft (ESL and the Remote Agent) (Muscettola *et al.* 1998). In PMA, we are drawing on and extending the techniques of planning and execution systems, applying them to a new domain: personal plan management. The key idea behind PMA is to apply AI technology for modeling and reasoning about plans and processes to the development of automated support of work activities.

Currently under development, PMA includes representations and algorithms we have developed for reasoning about richly expressive plans, which include explicit temporal constraints, temporal uncertainty, and observation actions and conditional branches. We have also developed and implemented an approach to computing the cost of a new plan in the context of existing commitments. The current version of PMA has a low level of autonomy: it makes suggestions to its user, but it does not directly act on her behalf. For example, it identifies conflicts among plans and suggests possible resolutions of those conflicts, but it does not automatically accept any of the suggested resolutions. PMA's usefulness will increase when it is given a greater degree of autonomy. It should have the authority to make some kinds of commitments and perform some kinds of actions on behalf of its user. However, its autonomy must be controllable and adjustable, and may vary with the situation. We envision cases in which PMA needs to perform deliberation to determine the degree of autonomy appropriate in some situation: specifically, it may reason about the potential costs and benefits of its making a decision on behalf of its user. In this paper, we first describe the PMA system, and then briefly raise some design questions we will need to address in making PMA an adjustably autonomous system.

Background

As mentioned above, PMA is an intelligent assistant, providing tools to help its user manage plans and commitments. It is thus related to two major classes of software systems: personal electronic calendar systems and workflow systems.

Commercially available electronic calendar systems, published by major software companies, essentially provide glass interfaces to written calendars. They typically have advanced GUIs, and provide linkages to contact databases and email; some also provide capabilities for meeting scheduling, for example, automatically generating email to notify meeting participants, and automatically selecting a time for the meeting based on the participants' responses. However, these systems suffer from a highly impoverished representation for activities: they can only model simple events and recurring simple

events. Simple events are blocks of time with a single property—"free" or "busy"; a "free" activity is allowed overlap with other "free" activities, but a "busy" activity cannot overlap with other activities. Recurring simple events are simple events that recur at regular intervals, e.g., every Tuesday from 4-5pm. Labels and textual information can be attached to each event, but these are not used in any sophisticated way by the system; they are stored only for the human user's information.

Workflow systems (Georgakopoulos, Hornick, & Sheth 1995; Nutt 1996; Mahling, Craven, & Croft 1995) constitute another class of systems aimed at helping users manage their routine activities. In contrast to the personal calendar systems, workflow systems employ richly structured representations of activities (or processes), and they use these representations to ensure that information and tasks flow to the appropriate people in an organization in a timely fashion. Modern workflow systems support "document management, imaging, application launching, and/or human coordination, collaboration, and co-decision" (Georgakopoulos, Hornick, & Sheth 1995, p. 121). On the other hand, they tend to have limited capabilities for handling uncertainty, for replanning when a problem is detected, and for reasoning about the relative value of alternative ways to perform a given task. PMA is being designed to include just these sorts of reasoning capabilities, some of which have been previously developed in work on plan generation and execution, and some of which we are developing within the PMA project.

PMA is a joint project of research groups at the University of Pittsburgh and the University of Maryland. Other efforts to apply AI plan generation and execution technology to develop workflow-style systems include the IWCM project at SRI International (Berry & Myers 1998) and the Enterprise Project at the Artificial Intelligence Applications Institute (Drabble, Lydiard, & Tate 1998; Stader & Jarvis 1998). Like PMA, the IWCM project is relatively new, and comparisons between the projects are not yet possible. Although some of the functionality of PMA overlaps with that of the Enterprise project, the approaches taken in the two efforts are rather different. For example, the Enterprise Project relies on a different model of plan generation (the O-Plan approach) and a different representation for activities (the <I-N-OVA> model (Tate 1996)) than we are using in PMA, and the techniques we are using for tasks such as determining the compatibility of plans and assessing the cost of a new plan are also quite different. As PMA is further developed, a more thorough comparison of it and the Enterprise Project will become valuable.

An Example PMA Interaction

To illustrate the behavior of PMA, we describe a sample interaction with it. PMA has knowledge of the structured activities—the "plans" or "procedures"—that its user typically performs. A PMA for an academic user would have a library containing descriptions (templates) of activities such as holding a meeting, teaching a class, attending a conference, overseeing the editing of a paper, and so on, while a PMA for use in aa physician's office would know the steps involved in carrying out diagnostic procedures, preparing a patient for surgery, and handling insurance forms. The activity of preparing a patient for surgery might include, say, organizing a preliminary battery of tests, assembling and scheduling the surgical team, booking the operating room, etc. And of course, many of these tasks would themselves decompose into structured activities: carrying out a single test might involve scheduling that test, tracking the lab work, entering the results into the patient's record, and calling the patient for follow-up work if necessary.

Imagine that a physician (or nurse) specifies the goal of performing a particular diagnostic test on a patient. PMA immediately posts commitments to various tasks pertaining to that goal in an internal knowledge base—the "schedule". It also updates the graphical display that includes a calendar and to-do list. In this example, the posted commitments might include scheduling the test, obtaining the necessary background information before the test date, reminding the patient 48 hours before the test date, and so on. Once the user indicates that the test has been scheduled for a certain date—December 15, say—the temporal information associated with the related procedures will be updated accordingly; for example, a calendar entry will then appear reminding the user to notify the patient on December 13. Furthermore, if this test is just one of a battery of tests, and the scheduled December 15 date places it too near another test with which it might interfere, PMA will notice this conflict and notify the user, suggesting an alternative schedule that avoids the conflict. It may also suggest to the user that an operating room should be scheduled now, even though the actual deadline for the reservation has not yet occurred, because there is limited flexibility in the schedule to handle the situation should the operating rooms become unavailable at the desired time.

This scenario illustrates the main capabilities that we are building into PMA:

- The PMA user can commit to activities that have rich temporal and causal structure. She does not need to specify separate commitments to each component of the activity.
- The PMA user can make partial commitments: for instance, she can commit to performing a particular activity without yet specifying the exact time at which it will occur, or she can specify that she wants to commit to a particular goal, without yet specifying exactly which plan she will use to achieve that goal.
- When the user extends her commitments (e.g., by specifying a particular time or a particular plan for a goal), PMA propagates the new commitment to all affected parts of the activity. In the example above,

when the user specifies that the test should be scheduled for Dec. 15, a patient reminder is automatically scheduled for Dec. 13.

- Whenever the user attempts to form a new commitment, PMA performs temporal and causal reasoning to determine whether it is consistent with the user's previous commitments. If PMA determines that certain additional constraints are required to ensure consistency, it notifies the user of those additional constraints, which we call "forced constraints". If PMA determines that there is a conflict between the new commitment and prior commitments, it suggests ways to resolve the conflict.
- PMA can assess the cost of executing a plan in the context of existing commitments, and notify the user if the cost fails to exceed some specified threshhold.
- As time passes, PMA monitors the execution of the user's activities, and reminds the user when deadlines are approaching. It also reasons about the tightness of the schedule: for instance, if there is little slack at some future periods, PMA may suggest taking early action.

To date, we have implemented the first five capabilities, but only for non-hierarchical activities.[1] The extension of these capabilities to hierarchical activities, and the implementation of execution monitoring, are part of our ongoing effort.

System Description

An initial version of PMA has been constructed and implemented on a Pentium Platform, using Allegro Common Lisp for Windows. Figure 1 illustrates the user's view of PMA.

The user interacts with PMA through three main components: a calendar, a to-do list, and an activity management window. The calendar lists all the fully scheduled tasks to which the user has committed; by fully scheduled, we mean tasks that have been assigned a specific time of occurrence. (Note that we use the term "activity" to refer to complete processes, such as performing a diagnostic test, and use the term "task" to refer to individual steps in an activity, such as "sending the patient a reminder message".) The to-do list shows all tasks that the user has committed to, but has not yet fully scheduled. Finally, the activity management window allows the user to add a commitment to a new activity, view the details of an existing commitment, or delete an existing commitment. The details of an activity can also be viewed by selecting a task in that activity from the calendar or the to-do list. In either case, the result is a pop-up window providing a description of the activity and details about constraints on it; the user can then modify those constraints if desired.

[1] As of this writing, the cost-assessment calculation has been successfully implemented, but not yet incorporated into PMA. We plan to incorporate it in the very near future.

1. Earliest Start Time for Task T
2. Latest Start Time for Task T
3. Earliest End Time for Task T
4. Latest End Time for Task T
5. Minimum Duration for Task T
6. Maximum Duration for Task T
7. Minimum Period of Separation between Tasks T_1 and T_2
8. Maximum Period of Separation between Tasks T_1 and T_2

Figure 2: Temporal Constraints for a Task

PMA has two primary knowledge bases. The first is an activity library, containing templates for domain activities. The second is the user's schedule, a record of the activities to which the user has committed, along with all constraints on those activities. The user's schedule thus includes fully or partially instantiated activity templates, along with supplemental constraints.

Activity templates are encoded using a standard AI planning language augmented to allow conditional branches and explicit temporal constraints. The latter allow the user to specify constraints on the start and end times and/or the duration of tasks. More specifically, the user can specify any or all of the constraints in Figure 2. Of course, these constraints may interact—for example, if all of constraints (1)-(4) are specified from a task T, then items (5) and (6) for T can be derived. We present the user with the full range of possibilities because it is sometimes more natural to specify some conditions than others. PMA checks that all input constraints, including temporal ones, are consistent with one another.

Whenever a new constraint is added to an activity, or a new activity is added to the user's schedule, PMA performs a consistency check. For this purpose, we have developed a new algorithm, described in detail in (Pollack, Tsamardinos, & Horty 1999). The algorithm has three stages. In the first, it employs a Conditional Simple Temporal Network (CSTN) to identify conflicts among plans. CSTNs extend Simple Temporal Networks (Meiri 1992) to include branching nodes. After conflicts have been identified, the second stage of our algorithm uses an approach developed by Yang (1997) to suggest a potential resolution of the identified conflicts. Yang's approach is not guaranteed to generate resolutions that are consistent with all the temporal constraints allowed in our constraint language, so the third stage of the algorithm involves re-using the CSTN to check for consistency. If the proposed resolution is determined to be inconsistent, then the algorithm backtracks and an alternative solution is proposed.

As a side-effect of consistency checking, constraints on a single task or activity are propagated to other tasks

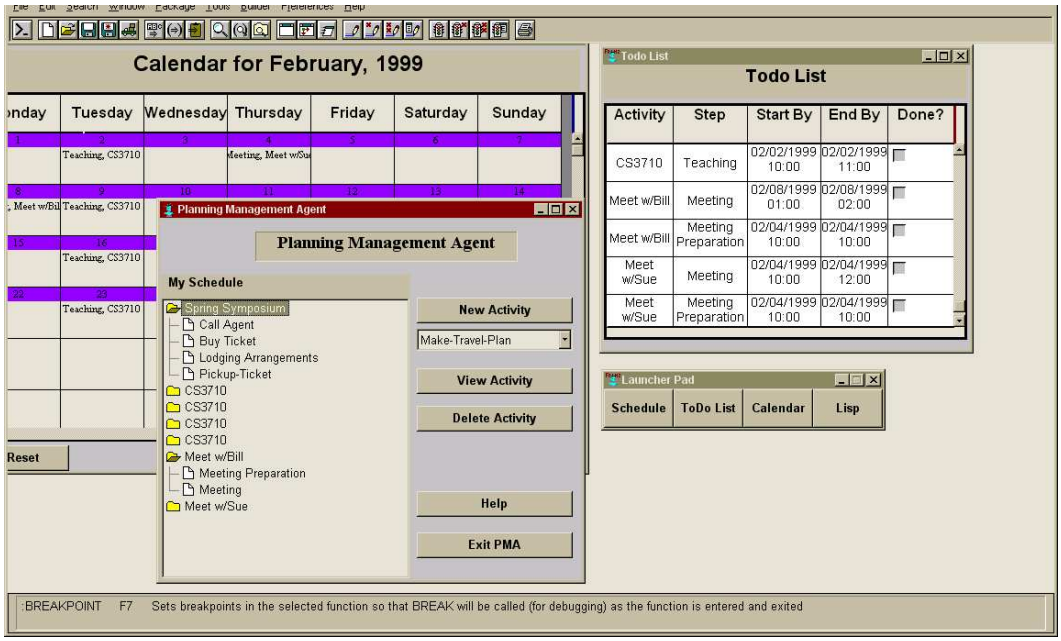

Figure 1: PMA: The User's View

and activities in the user's schedule. Additionally, when a conflict is identified, our algorithm finds candidate ways to resolve the conflict. These are suggested to the user, who can select among them, or can retract the new constraint or activity entirely.

An additional reasoning task performed by PMA involves assessing the cost of a new potential commitment. By "cost" we mean all types of resource usage, including the user's effort, and not just monetary cost. In general, because activities may interact with one another, the cost of one activity may vary, depending upon the other activities to which the user has already committed. We have developed and implemented an approach to evaluating the cost of a potential option in the context of existing commitments; details can be found in (Horty & Pollack 1998). If activities have known values associated with them, then cost assessment can be used to determine whether an activity is worth pursuing. Our cost-assessment algorithm uses cost estimate intervals: as soon as the new activity's value is determined to lie outside the cost-estimate interval, it can in principle immediately be accepted (if its value exceeds the cost interval) or rejected (if the value is below the cost interval). PMA will point this out to the user, but will not accept or reject an option directly.

A final component of PMA, which we have yet to implement, is an execution monitor, which will track the execution of activities to which the user has committed. The execution monitor will warn the user when her tasks are about to become due, reasoning about the appropriate time at which to issue the reminder, taking account the temporal constraints associated with the activity in question. Additionally, we hope to include probabilistic reasoning capabilities in the execution monitor, so that it can warn the user when the likelihood of successly completing some activity drops below a certain threshhold.

Adjustable Autonomy Issues

As mentioned earlier, PMA currently has a very low level of autonomy. Although it can detect conflicts among plans, and can suggest potential resolutions, it never adopts a resolution directly, but instead leaves it to the user to accept one of its suggestions. Similarly, although it can compute an estimated cost for a new activity and determine whether the activity has a value that is greater (or less) than its cost, it never directly adopts or rejects a new commitment for its user. Although it will be able to determine when the likelihood of succeeding at an activity has dropped below a certain level, it will not, in the initial version, modify the set of commitments to increase the probability of success.

PMA's usefulness will increase when it is given a greater degree of autonomy. It should be able to make some kinds of commitments and perform some kinds of actions on behalf of its user, and to defer to the user in other situations. For example, we might want a PMA for an faculty user might to schedule routine meetings with students, and if the user specified a conflicting appointment, we might want PMA to automatically change the times of the student meetings (notifying the student of course!). On the other hand, we would not want PMA to change the times of some other meetings, e.g., a meeting with the Dean, without first obtaining the approval of its user. A key question is how PMA can be made to recognize which commitments are strictly

under the control of the user, and which ones can be directly modified by PMA. Can control be specified in terms of the types of activities and/or their arguments, or will the control status of each activity token need to be specified individually, by the user?

In addition to modifying the user's schedule—adding, deleting, and changing commitments—we might also want PMA to perform certain tasks on behalf of its user. Again, using the example of a PMA for a faculty member, suppose that the user is also an editor for a journal. We may want PMA to directly perform some of the tasks in the "edit paper" activity, for instance, automatically updating the user's tracking sheet as reviews for the meeting as received. Again, the question arises as to which tasks PMA should have authority to perform, and how this information can be encoded.

In general, PMA's authority may vary with certain features of the current situation. For instance, when a user is on vacation, she may want to grant PMA authority to modify some (types of) commitments and/or perform some (types of) tasks that she might otherwise directly oversee. In general, PMA may need to perform deliberation to determine the degree of autonomy it should adopt. More specifically, it may need to reason about the potential costs and benefits of its making a decision on behalf of its user, and only assume autonomy when the potential benefits are high and the potential are low. We speculate that some form this calculation will be central in all adjustably autonomous agents.

Acknowledgements This research has been supported by the National Science Foundations grants IRI-9619579 and IRI-9619562 and by the Air Force Office of Scientific Research contract F49620-98-1-0436.

References

Berry, P. M., and Myers, K. L. 1998. Adaptive process management: An AI perspective. In *Proceedings of the Workshop Towards Adaptive Workflow System*. Available from http://www.ai.sri.com/berry/.

Drabble, B.; Lydiard, T.; and Tate, A. 1998. Workflow support in the air campaign planning process. In *Proceedings of the Workshop on Interactive and Collaborative Planning, AIPS98*.

Firby, R. J. 1996. Modularity issues in reactive planning. In *Proceedings of the Third International Conference on AI Planning Systems*, 78–85.

Georgakopoulos, D.; Hornick, M.; and Sheth, A. 1995. An overview of workflow management: From process modeling to workflow autonomation infastructure. *Distributed and Parallel Databases* 3:119–153.

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*, 972–978.

Horty, J. F., and Pollack, M. E. 1998. Option evaluation in context. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98)*, 249–262. San Francisco: Morgan Kaufmann.

Mahling, D.; Craven, N.; and Croft, W. B. 1995. From office automation to intelligent workflow systems. *IEEE Expert* 10(3).

Meiri, I. 1992. *Temporal Reasoning: A Constraint-Based Approach*. Ph.D. Dissertation, UCLA.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote agent: To boldly go where no AI system has gone before. *Artificial Intellience* 103:5–47.

Nutt, G. J. 1996. The evoluation towards flexible workflow systems. *Distributed Systems Engineering* 276–294.

Pollack, M. E.; Tsamardinos, I.; and Horty, J. F. 1999. Merging plans with quantitative temporal branches, temporally extended actions, and conditional branches. Submitted for publication; available from the author's Web page.

Rao, A. S., and Georgeff, M. P. 1995. BDI-agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems*.

Stader, J., and Jarvis, P. 1998. Intelligent support for enterprise modelling. Technical Report AIAI-TR-220, AIAI, Edinburgh, Scotland.

Tate, A. 1996. Representing plans as a set of constraints – the <I-N-OVA> model. In *Proceedings of the Third International Conference on AI Planning Systems (AIPS96)*, 221–228. AAAI Press.

Yang, Q. 1997. *Intelligent Planning: A Decomposition and Abstraction Based Approach*. New York: Springer.