A Personalized Calendar Assistant

Pauline M. Berry, Melinda Gervasio, Tomás E. Uribe, Karen Myers, and Ken Nitz

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
{surname}@ai.sri.com

Abstract

Many calendar tools have become available to organize, display, and track a user's commitments. However, most people still spend a considerable amount of time personally organizing meetings and managing the constant changes and adjustments that must be made to their schedules. Our goal is to provide the technology necessary to manage an individual's calendar. The resulting agent will let the user retain control of decisions when necessary and relinquish control to the assistant at other times. Meanwhile, the agent will be sensitive to the user's wishes and preferences.

The key elements in our approach are the creation of a *process framework* that captures possible interactions with users and other agents, *learning* technology to capture the user's preferences, and *advisability* to enable direct instruction by the user at various levels of abstraction. As the system improves its model of the user over time, reliance on user interaction will decrease.

Introduction

Management and command decision makers handle many simultaneous activities while responding to unexpected opportunities and obstacles. A key organizational tool is their calendar and an important support function is the maintenance of their calendar and the coordination of meetings and other shared tasks. Often the decision-maker's performance depends on interactions over long periods of time with people inside and outside of the organization, wherever and whenever those interactions need to take place. The objective of this work is to build a Personalized Calendar Assistant (PCalM) to coordinate these interactions and schedule events on the user's calendar. This paper describes the project vision, the current implementation and the research challenges for our future work.

PCalM is designed to build the user's trust in its capabilities so that the user will delegate significant responsibility: entirely transferring to it routine scheduling tasks.

The functional goals of PCalM are threefold:

- To increase autonomy through use: as the user becomes more confident with the agent's ability, the agent can retain more decision-making responsibility
- To have the agent learn when to involve the user in a decision as autonomy increases
- To have the system learn user preferences through a combination of passive learning and advice-taking

A process framework captures the key decision points in calendar management and allows the user to make choices or give advice within this clearly defined framework. These choices range from selecting an alternative process, such as *negotiate or relax*, to simple preferences over time slots, such as *don't schedule meetings just before lunch.* In addition, these decision points provide the structure over which the agent can learn process preferences, scheduling preferences and eventually new processes from the user.

Let's consider a scenario:

"Annie (*A*) wants to organize a meeting with her boss Bill (*B*), an external client, Clark (*C*), and one of her design engineers, Doug or David (*D1* and *D2*). She wants the meeting to happen next Monday and requires at least a -hour slot and a private space with a projector. Her Personal Calendar Assistant (PCalM) replies that the request is not feasible because although the client arrives in town next Monday his travel schedule means it will be too late to schedule a meeting. *A* relaxes the time constraint indicating that the meeting could occur anytime next week. Both *D1* and *D2* have blocked out their schedules for Thursday and Friday because a development deadline is looming on the horizon. A's boss is free only on Thursday and Friday. The PCalM agent is aware that *A* does not like to bother her boss so she initiates a negotiation cycle with *D1* and *D2*. *D1*'s PCalM quickly responds that *D1* is willing to meet anytime on Thursday, and *A*'s PCalM agent quickly accepts the offer and proposes several time slots to *A*. These candidate time slots are ordered using evaluation criteria that have been learned over time to reflect the ones that *A* prefers."

The Personal Calendar Assistant project is part of a more ambitious automated assistant called CALO. CALO is a personal assistant that is persistent, can act autonomously, and is robust. The above scenario is part of the CALO evaluation schedule and supported by the current implementation of PCalM. PCalM is an early test of the hypothesis that in order to persist, an intelligent agent must learn and adapt to the user's changing needs.

Background

It is interesting to note the recent surge in meeting systems for both research and commercial needs. Tools and standards for representing, displaying, and sharing schedule information have become common. A generally adopted standard for calendar representation is iCalendar [RFC2447]. As mentioned, these systems still leave the user with the bulk of decision-making and negotiation responsibilities.

The emphasis in the research community has been on automated meeting scheduling: how to find feasible time slots for meetings given multiple, overlapping sets of participants and constrained temporal and, perhaps, location requirements. Generally, the work in this area can be divided into two categories specified by [Ephrati et al. 1994], as *Open* and *Closed* scheduling systems. In *Open* systems each individual is an autonomous entity responsible for creating and maintaining his or her own calendar and meeting schedules, perhaps selfishly. The user may operate in an unbounded environment without constant obligation to one organization. In a *Closed* system the meeting mechanisms are imposed on each individual within the system, and a consistent and complete global calendar is maintained. Closed systems are more common because preference measures can be normalized across participants, participant availability is known at all times and the problem can be formulated as one of constraint optimization. However, not all closed systems are centralized and there has been some interesting work in distributed solutions to the closed scheduling problem [Ephrati et. al.1994] [Sandip and Durfee 1998].

Closed systems are rarely adopted in because the users seldom live in a truly closed environment and so need to retain more personal control of their calendars. Open scheduling systems pose other interesting challenges, such as privacy: an individual may not wish to share all, part, or any of his schedule, or may choose not to participate in a meeting.

PCalM

CALO exists in an open, unbounded environment and thus PCalM is an open scheduling system. PCalM's primary responsibility is to its user. Issues of privacy, authority, cross-organizational scheduling and availability of meeting participants abound. There is some exciting work in the area of meeting scheduling for Open systems [Franzin et al. 2002] and [Payne et al. 2002]. In the former, complete privacy is assumed and the availability and preferences of other users are learned across time. In the latter, RCAL, a more cooperative environment is assumed and meetings are scheduled using constraint-based algorithms.

PCalM is based on existing open scheduling algorithms and is similar in approach to the RCAL system. However, we extend the notion of collaboration in general, and more specifically to the individual user. The collaborative scheduling process is separated from the constraint reasoning algorithms to enable interaction with the user and other PCalM agents. This interaction forms the framework for learning and adjustable autonomy.

The calendar management processes are represented as context-sensitive, hierarchical procedures. These provide hooks into the user's decision process at multiple levels of abstraction. These hooks can be used to passively learn the user's preferences or to facilitate the specification of advice from the user. Through the continual use of passive learning [Scerri et al. 2001] and advice taking, PCalM constructs a dynamic preference profile containing two types of guidance:

- Preferences over schedules, either locally (e.g., the time, duration, location of an individual meeting) or globally (e.g., the density or distribution of meetings or average gap between meetings)
- Process selection and application criteria over both existing process descriptions (e.g., negotiate or relax) and new processes

Both types of information can be actively asserted using a policy specification language, building on work on advisability and adjustable autonomy [Myers and Morley 2003] or passively learned by monitoring the user's decisions.

Figure 1 illustrates the functional architecture. The short-term objective is to effectively manage the user's calendar process and preferences. In the future, more extensive studies of the interaction between agents will be conducted in collaboration with USC-ISI [Tambe and Zhang 2000].

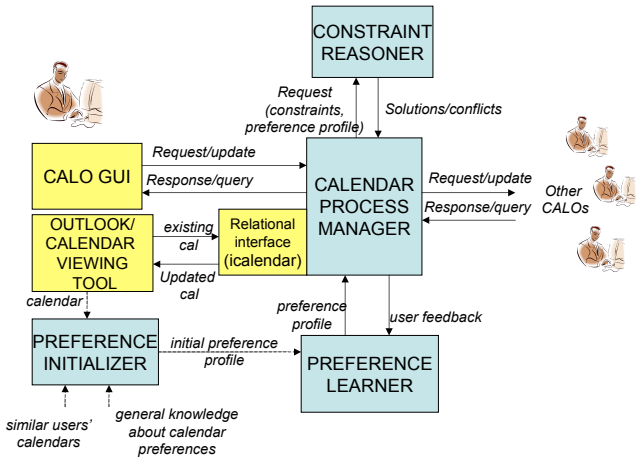

Figure. 1 Functional Architecture

PCalM is a multiagent system. The agent architecture OAA [Cheyer and Martin 2001] provides direct connectivity and is common to the overall CALO system. PCalM has several functional components:

- The **process manager**, is responsible for managing the scheduling and negotiation strategies.
- The **constraint reasoner** (reasoner), maintains consistency of scheduling commitments and provides solutions to new scheduling problems.
- The **preference learner** (learner), monitors scheduling decisions made by the user or pcalm and learns preferences over time. It also provides these preferences in the form of evaluation functions for use by the reasoner.
- The **preference initializer**, provides a basis for populating the preference profile of a new user or maintaining it as cross-CALO learning improves.
- The **calendar tool**, has initially been an instrumented form of OUTLOOK. However, the system is designed to be calendar tool-independent and is compatible with the icalendar standard [Dawson and Stenerson 1998].

Schedule Management

The process manager component is built on top of a procedural reactive controller called SPARK [Morley 2004]. SPARK is a new Belief-Desire-Intention agent framework grounded in a model of procedural reasoning [Georgeff and Ingrand 1989]. SPARK incorporates an agent-based language that allows the development of active systems that interact with a constantly changing and unpredictable world. It enables both goal-directed and reactive behavior.

The procedural framework allows hierarchical, context-dependent processes to be explicitly encoded and both goal-directed and reactive execution to be managed seamlessly. The explicit representation of processes, or *strategies*, makes it possible to track the progress of an active process, to modify ongoing processes in response to situational change, and even to adopt newly defined, or learned, processes.

The structured management of processes within this framework also provides a way to standardize operation, reducing the likelihood of errors and unexpected behaviors. This feature is extremely important if PCalM is going to gain the trust of the user over time. The explicit representation of preconditions for the applicability of a process, decision points within a process, and ease of transition to alternative refinement of strategies allows the involvement of the user in decision-making to be adjusted seamlessly. These same decision points also provide the basis for learning of user preferences as discussed in a later section.

The scheduling strategies for PCalM are designed to address the following key issues for a personalized scheduling tool

Scheduling Requests

The user or another CALO agent requests to schedule a new, or modify an existing, commitment. Even in a straightforward request from a user to delete a meeting, various strategies must be considered – or example, delete and notify other participants, request that meeting host delete user from participant list, delete the whole meeting. At each decision point there is an opportunity to involve the user. Each user may prefer a different strategy or a different level of involvement in decisions thus motivating the need for adjustable autonomy.

Information Sharing

The sharing of information about one's calendar in an open system is extremely important and impacts issues of privacy and security, such as when and how much information to request or supply. It also impacts the ability to schedule meetings efficiently. The approach of PCalM is to enable and allow users to define their own policies for sharing their calendar information. Thus, strategies must be able to adapt to a varying degree of knowledge about another's calendar and scheduling preferences.
As part of the PCalM development we are extending our shared calendar information to include user preference over possible time slots and a measure on individual commitments to represent the potential cost to delete or reschedule.

Conflict Resolution

When a conflict is detected there are two main classes for resolution: relaxation and negotiation. Relaxation and negotiation can be viewed as independent strategies for resolution or may be intertwined and interdependent. The implementation of PCalM currently refers the decision to relax or negotiate to the user. In the future, strategies will

be developed to provide functionality automatically. An interesting concept being considered is the learning of relaxation strategies.

Advice

The process manager is able to take advice from the user and conform to organizational policies. Advice is defined to be an enforceable well-specified constraint on the performance or application of an action in a given situation. [Sloman 1994] defines two types of policy: authorization and obligation. We extend this categorization to include preference.

- Authorization defines the actions that the agent is either permitted or forbidden to perform on a target.
- Obligation defines the actions that an agent must perform on a set of targets when an event occurs. Obligation actions are always triggered by events, since the agent must know when to perform the specified actions.
- Preference defines a ranking in the order or selection of an action under certain conditions.

Advice can apply to the application of strategies, the conditions under which a strategy is applicable, or the instantation of a variable. Advice may be conflicting, can be long lived and its relevance may decay over time. Advice can be used to influence the selection of procedures and strategies for problem solving and also to influence adjustable autonomy. The management of advice is an open issue for CALO, but the application of advice is central to both SPARK and PCalM.

The Constraint Reasoner

To find suitable meeting times, we use a constraint reasoning module, implemented in the ECLiPSe constraint logic programming language [Wallace et. al 1997]. This module exposes an OAA interface [Cheyer and Martin 2001] where meeting schedules for the different participants are asserted, preferences expressed, and meetings requested. One, all, or a given number of solutions can be returned. Given a cost function that expresses the preferences, the module can rank the solutions according to their cost, returning the best ones first.

The solutions could be ranked strictly according to the evaluation, but this will most likely produce a set of solutions with small variations. Instead, the system will attempt to use specific preferences to present qualitatively different solutions.

We are extending this module to support constraint relaxation, where partial or approximate solutions are returned if not all the desired constraints can be met. The user preferences will define a ranking on the constraints, indicating their relative importance. These preferences may be adjusted by the learning module after each scheduling task is completed.

Learning Preferences

As an adaptive assistant that learns to tailor its behavior to an individual user, PCalM's preference learning component is an important part of the system. Calendar Apprentice (CAP) [Mitchell et al. 1994] is another system that applies machine learning techniques to the task of automatically acquiring user scheduling preferences. However, while CAP learned to predict values for specific meeting attributes such as location and duration, PCalM instead learns an evaluation function used to rank candidate meeting schedules. PCalM is thus similar to systems such as the Adaptive Route Advisor [Fiechter and Rogers, 2000] and the Interactive Crisis Assistant [Gervasio et al. 1999], which learned evaluation functions for ranking candidate driving routes and crisis response schedules, respectively.

Formulating the scheduling assistance task as a ranking problem presents both advantages and challenges. By letting the system suggest more than one (but not too many) candidate schedules, we increase the probability that the user will find an acceptable suggestion without having to consider all alternatives. However, when a user selects a suggested schedule (or overrides all suggestions), the only feedback PCalM gets about the goodness of its ranking function is a preference of one schedule over some others. The straightforward application of machine learning approaches to learning ranking functions typically requires much richer input—for example, a ranked list of candidates or instances associated with specific numeric value assignments.

In our initial implementation of the schedule preference learner, we explored two approaches. The first was a large margin method [Fiechter and Rogers, 2000] that basically learned the tradeoffs a user was willing to make between meeting schedule attributes such as (closeness to) specified meeting date and earliness. Because we were interested in providing reasonable scheduling assistance early in the learning curve and in rapidly acquiring user preferences, we extended this approach with an active learning strategy that proposed schedules about which a user selection would provide the most information for refining the preference profile.

However, while casting preferences as tradeoffs had been intuitively appealing for problems such as providing driving route advice and recommending flight itineraries, it presented a less compelling case for meeting scheduling, where user preferences seemed less a matter of tradeoffs and more a matter of having preferences about particular attributes or combinations of attributes. That is, the problem seemed more one of learning which days and times a user preferred for which meetings rather than learning how much a user was willing to deviate from a

particular day of the week if the meeting could be kept to a particular duration. In addition, because the large margin method was a non incremental algorithm applied in an online setting, we experienced performance degradation as the system acquired more training examples.

For these reasons, we implemented an incremental Naïve Bayesian approach that learns a model of the preferred class from selected schedules (i.e., positive examples only) and used degree of membership in that class to rank candidate schedules. An important feature we wanted to accommodate was the incorporation of explicit user preferences into the learned model. To accomplish this, we convert explicit preferences about attributes or combinations of attributes (e.g., prefer Monday meetings and Wednesday afternoon meetings) into equivalent training examples for the learning algorithm.

Preliminary experiments with synthetic data show how incorporating these explicit preferences can improve the learning rate. Because of the challenges of evaluating interactive systems, we are currently designing more thorough experimental evaluation using synthetic and human users in both real and artificial settings. We plan to explore other approaches to learning ranking functions such as learning ensembles of rankers [Cohen et al. 1998] that may be useful for learning more complicated preference functions. And while we have focused thus far on learning from direct user interaction, we are also interested in leveraging patterns of behavior learned from historical data about scheduled meetings. Ultimately, we want PCalM to be learning more complex preferences such as processes for scheduling meetings of a particular type or procedures for resolving scheduling conflicts. We thus plan to investigate various approaches to procedural learning within a mixed-initiative setting, including interactive knowledge acquisition and programming by demonstration.

Future Challenge

As this project continues, we will address several key challenges,

Conflict Resolution. Although some traditional resolution strategies have been incorporated into the meeting scheduler, there is an opportunity to explore the twin problems of relaxation and negotiation to improve the resolution of scheduling conflicts. We intend to build on existing work on learning strategies for constraint relaxation in other domains to learn how and when to relax meeting scheduling constraints. Similarly, the notion of relaxation is a key component of more complex negotiation strategies.

Probabilistic Information and Cost. Some calendar events are scheduled but do not often occur,and users may double-book themselves, knowing that one or other of the events may not occur as planned. It will be important for PCalM to be able to represent the probability, or possibility, of the actual occurrence of an event as well as the cost to delete or reschedule an event. This information, combined with adequate monitoring systems, could provide more realistic calendar reasoning.

Adding Advice and Learned Knowledge or Capabilities. As advice, preferences, new strategies, and other new knowledge are added to the system there must be a concept of anytime verification to ensure that these new constructs are compatible with the existing system goals, functionality and robustness. In reality, conflicts will be created that must be resolved if the system is to run seamlessly as it adapts.

Summary

We have described ongoing work on an adaptive personalized calendar management tool called PCalM. The tool is designed to be persistent**,** gain the user's trust and, as a result, become more autonomous over time. The key elements of an explicit and adaptable process framework, learning algorithms, and adjustable autonomy provide the necessary foundations for robust reasoning, and the inclusion of preference and advice formalisms offer the flexibility required for the tool to be useful and personalized.

Acknowledgments. This work is supported by DARPA under Contract NBCHD030010.

References

Cheyer, A. and Martin, D. (2001) The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1, pp. 143-148, March.

Cohen, W. W., Schapire, R. E., and Singer, Y. (1998). Learning to order things. *Advances in Neural Information Processing Systems.*

Dawson, F., and Stenerson, D. (1998). Internet Calendaring and Scheduling Core Object Specification (Icalendar), Network Working Group RFC 2445, (http://www.ietf.org/rfc/rfc2445.txt).

Ephrati, E., Zlotkin, G., and Rosenschein, J.S. (1994). A non manipulable meeting scheduling system, *Proc. Thirteenth International Distributed Artificial Intelligence Workshop*, Seattle.

Fiechter, C.-N. and Rogers, S. (2000). Learning subjective functions with large margins. *Proc. Seventeenth International Conference on Machine Learning*, pp. 287-294.

Franzin, M. S., Freuder, E. C., Rossi, F., and Wallace, R. (2002) Multi-agent meeting scheduling with preferences: Efficiency, privacy loss and solution quality. *AAAI 2002 Workshop on Preference in AI and CP.*

Georgeff, M.P. and Ingrand, F. F. (1989). Decision-making in an embedded reasoning system, *Proc. Eleventh International Joint Conference on Artificial Intelligence.*

Gervasio, M., Iba, W., and Langley, P. (1999). Learning user evaluation functions for adaptive scheduling assistance. *Proc. Sixteenth International Conference on Machine Learning*, pp. 152-161.

Mitchell, T. M., Caruana, R., Freitag, D., McDermott, J., and Zabowski, D. (1994). Experience with a learning personal assistant. *Communications of the ACM*, vol. 37, no. 7, pp. 80-91.

Morley, D. (2004). Introduction to SPARK, Technical Report, Artificial Intelligence Center, SRI International, Menlo Park, California.

Myers, K. L. and Morley, D. N. (2003). Policy-based agent directability. *Agent Autonomy*, Kluwer Academic Publishers.

Payne, T. R., Singh, R., and Sycara, K. (2002). Rcal: A case study on semantic web agents, *Proc. of First International Conference on Autonomous Agents and Multi-agent Systems.*

Sandip, S., and Durfee, E.H. (1998). A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, vol. 7, pp. 265-298.

Scerri, P., Pynadath, D., and Tambe, M. (2001). Adjustable autonomy in real-world multi-agent environments. *Proc. International Conference on Autonomous Agents (Agents'01).*

Sloman, M. (1994). Policy driven management for distributed systems. *Plenum Press Journal of Network and Systems Management,* vol.2, no. 4, pp. 333-360.

Tambe, M. and Zhang, W. (2000). Towards flexible teamwork in persistent teams: extended report, *Journal of Autonomous Agents and Multi-agent Systems*, vol. 3, pp.159-183.

Wallace, M., Novello, S., and Schimpf, J. (1997). ECLiPSe: A Platform for Constraint Logic Programming. Technical report, IC-Parc, Imperial College, London.