M.I.T. LABORATORY FOR COMPUTER SCIENCE
Computer Systems Research Division

March 21, 1980
Request for Comments No. 184

Programs for Distributed Computing: Designs of Calendars

by

Irene Greif

Abstract

This report on the calendar application focuses on the functionality of the system and includes a section on a subproblem having to do with the distribution of the calendar data.

1. Introduction

We began our design of calendar systems with the intention of focussing on implementation issues, particularly those of communication, sharing of data, and the use of forms as the standard interface among modules. In [Greif, 1979] we outlined some possible design alternatives that were intended to provide a variety of implementation projects from which we could gather information about these issues. Several such calendar implementations are now underway.

During later design phases we have become more interested in the functionality of the calendar and its user interface. To some extent the use of forms as the communication medium shapes the human interface as well as the process-to-process interface. Also, some aspects of the functionality, most notably the notion of "tentative meeting" have arisen from analysis of the distributed implementation.[1] Section two is a report on the functionality of the calendars we are building. Section three is about the distributed implementation.

2. The Functionality of the Calendar

There are two kinds of calendars that we have been designing -- personal calendars and public "resource scheduling" calendars.

2.1 The Personal Calendar

The personal calendar can be used for keeping track of appointments, meetings, holidays, etc. The calendar can be displayed in several ways showing either a summary of the week, a list of appointments on a day, or a diagram of the day showing blocks of free and reserved time. The main operations are "appt" to

1. Leslie Lamport inspired this line of thought when he insisted on knowing what my calendar did, and started to help me specify the meaning of a call for a meeting.

make an appointment, "cancel" to cancel one, and various display commands. A description of the commands will be typed out in response to "?."

One can attempt to make an appointment at any time. If there is a conflict with another appointment, the calendar reports this fact. If not, the appointment will be made. Appointments are recorded at a particular time with a few keywords to indicate the purpose.

2.2 A Calendar for a Conference Room

The Conference Room Calendar is similar to the personal calendar in that time slots can be reserved and cancelled. This program is meant to support the reserving of time in one of our conference rooms in the laboratory. The room is generally used for seminars and may involve the coordination of several people and resources. Since a seminar generally has a host who is responsible for the reservation, the host's name is listed in the calendar display as the keyword for the appointment. In addition, there is a form on file for each appointment. The form contains information about the seminar such as the speaker's name, the title of his talk and whether there will be refreshments. These forms can be active, in which case they may trigger communication with other calendars (such as the calendar for the person who sets up the coffee pot in time for scheduled refreshments).[1]

1. Simple single process versions of both the personal calendar and the 512 calendar have been implemented. The personal calendar can be run on XX by copying <greif>pcal.exe and <greif>_xfile.pcal and then typing pcal at exec. For the 512 calendar, copy <greif>512.exe and <greif>_xfile.512 and type 512. No guarantees are made about the performance or consistency with the description in this paper. The programs will leave some files named "your-name.cal.1", "512.cal.1", and "512.forms.1" in your directory.

2.3 Coordination Between Personal Calendars

A personal calendar can try to call a meeting. The desired length of the meeting, a set of possible times and a list of participants must be specified in the request. The calendar system will try to find a time at which the meeting can be held and will then notify all participants.

For meetings that are called very far in advance of the time at which they will be held, the meeting can be considered to be tentatively scheduled. A scheduler will keep track of several possible times at which the meeting can be held. A second meeting is considered to conflict only if scheduling it (and therefore removing its time slot from the set of times tentatively reserved for the original meeting) would reduce the set of possible times to less than one. If the second meeting is scheduled, the set of available times for the first meeting is simply reduced. Shortly before the date of the meeting a single time is chosen for the meeting. This can occur either at a "commit" time specified in the call for the meeting or by an explicit request to commit. A caller could specify that he wants a meeting the week of March 10th and that it should be definitely scheduled by March 3rd. Thus the caller can be sure that the meeting will appear on his calendar with sufficient advance notice for planning. If the meeting is committed to a single time too soon, it is quite likely that some participant will have to cancel in order to meet a higher priority commitment that arises later. This would require rescheduling, rather than the simple reduction in the set of tentative times.

2.4 Coordination Between Personal Calendars and Resource Calendars

In the laboratory now, when someone needs the conference room, either that person or his secretary calls the person in charge of the room and asks for a time. Such a request can be initiated by a person at his personal calendar if, for example, as part of arranging a meeting one could request a room for the meeting. The personal calendar and the resource calendar could then negotiate for an available time slot. We expect that most appointments on the conference calendar will be made by other calendars so that the person-in-charge need not be involved, except perhaps as reviewer of the calendar's decisions.

2.5 Making Requests -- The User Interface

As mentioned above, in the case of seminars we keep a file of forms containing information which does not necessarily appear on the calendar display. For uniformity we implement all interactions with the calendar in terms of forms so that even a reservation for the personal calendar is considered to be accomplished by filling in a form -- a "request for reservation" form. The fields are date, start time, end time and keywords. A cancellation form requires only a date and start time. Requests for displaying parts of the calendar are made by filling in the date.

The user need not know what information is required for a particular request -- the form itself will prompt him once he specifies the kind of request he wishes to make.[1] This is a kind of active form in which the filling in of one field (the type of request field) triggers the asking of further questions -- the form only asks for the fields that contain information relevant to the particular request. Changes to entries can be made by entering edit mode. In this mode the system asks which field is to be modified and for each field, displays the old contents of the field and accepts a new entry. There is currently a distinction made between fields that can always be changed (e.g. title of a talk) and those which require rescheduling (e.g., start time). To change the latter one must enter "reschedule" mode.

Sometimes filling in one form will trigger filling in an auxiliary form. This could arise, for example, when calling a meeting that has a room requirement. The room requirement implies that the seminar request form must be filled in and sent to the conference room calendar.

1. The current implementation of pcal does not prompt for all fields in "appt" requests.

Most request forms are disposed of as soon as the request is fulfilled. This is not true of the seminar request forms, since that information may have to be referred to (and possibly even modified) later. These forms are kept on file according to date and time of the seminar. It should also be possible to file incomplete forms that are not yet associated with a definite time. For example, if a request conflicts with scheduled meetings, but an alternative time cannot be suggested immediately, the user might store the form and retrieve it later, avoiding retyping all of the information in order to resubmit the request. A user interface with more sophisticated forms manipulation facilities is being designed by a bachelor's thesis student [Wylen, 1980].

The next stage of design for a user interface will include consideration of the coordination among the personal calendar and other personal resources such as lists of things to do. Probably there should be a uniform interface to all of the user's resources, so that the calendar and the various other personal resources can all be considered to be parts of a single system. In the context of an office support system the user might expect to enter the calendar system immediately upon login so that all of these resources might be available directly. Alternatively, the calendar, or various other parts of the time management system might be subsystems which the user enters only as desired. Urgent messages from any of them might be presented to the user at login time. Deadlines and reminders from various lists might show up on the calendar, but details would be available only if the user decided to enter his calendar system. Appointments made by the calendar program itself might be flagged until seen and confirmed by the user. This is similar to the current arrangement on the TOPS-20 where one gets messages about new mail in EXEC, but can only read the mail by entering mm.

3. Calendars in a Distributed System

Facilities for coordinating a set of calendars are of use in either a centralized or a distributed system. If the system is to be distributed, its implementation will certainly differ from the implementation of a centralized version. We are assuming that in order to coordinate with another calendar a request must be sent to that calendar. That is, there is no central data base that contains information on all calendars and that can be accessed directly by any calendar.[1] Thus from the point of view of anyone who needs information about several people the data is stored in a distributed data base.

Operations other than calls for meetings may depend on data at more than one node. For example, when there are tentative meetings (as described in section 2.3) then while a meeting is "uncommitted" the status of certain time slots on the personal calendars of the participants may depend on the status of the tentative meeting. Thus even if the personal calendars store their data locally they may have to communicate with the tentative meeting in order to find out whether a particular time slot is free. This can cause noticeable delays if a user is at the terminal trying to schedule an appointment in real time. It also raises a question as to how to display the calendar -- should all tentative times for various meetings be shown or should the display show a possible schedule based on information available locally?

Other questions arise:

- How do these data dependencies relate to the dependencies which arise in supporting modular atomic transactions [Reed, 1979]? Are such dependencies at the application level likely to occur in many applications? If so, how can we support their implementation in a programming language for distributed applications?

1. This does not preclude storing all information on a central storage device such as a Distributed Data Storage System [Svobodova, 1980]. Access to any process' data, e.g. a calendar's data, would still be under that process' control.

- Should the caller of the meeting act as the source of information about the tentative meeting? If the tentative meetings are distributed in this way how will scheduling be done if one person is invited to several meetings? Should a central scheduler be invoked to manage meetings? (This latter approach is being explored by a UROP student.)

- Should chains of tentative meetings be schedulable? (E.g., can I schedule meeting A conditionally depending on the final timing of meeting B?) This may save the time of checking with the tentative meeting about a particular time slot. But then how will the system help me in backing out of meetings when conflicts are later confirmed?

4. Conclusion

This report on the progress in the calendar application has focused on the functionality as seen by the end user. Most of the user interface functions have been implemented in a single user setting. Some simple communicating calendars have been built. "Tentative meetings" have not been implemented but initial design efforts have already generated a number of additional design issues such as those mentioned in the last section.

The recent focus on functionality and planned work on the interface may lead to a more concerted effort to build a calendar service available in the laboratory in advance of a truly distributed environment. To this end, suggested extensions to the functions of the calendar are welcome. The distributed version (or versions) are being developed not as a service but as a source of research questions.

5. References

Gifford, D. K., "Weighted Voting for Replicated Data," *Proceedings of the Seventh Symposium on Operating Systems Principles*, Pacific Grove, California, December, 1979, pp. 150-162.

Greif, I., "Programs for Distributed Computing: An Interim Report of an Experiment," M.I.T. Laboratory for Computer Science. Computer Systems Group RFC 180, December, 1979, pp. 1-13.

Reed, D. P., "Implementing Atomic Actions on Decentralized Data," *Preprints for the Seventh Symposium on Operating Systems Principles*, Pacific Grove, California, December, 1979, pp. 66-74.

Svobodova, L., "Design of a Distributed Data Storage System," M.I.T. Laboratory for Computer Science. Computer Systems Group RFC 182, January, 1980.

Wylen, E., "A Personal Calendar: The Human Computer Interface." Bachelor's Thesis, Expected June, 1980.