The following paper was originally published in the
Proceedings of the Eleventh Systems Administration Conference (LISA '97)
San Diego, California, October 1997

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Automating 24x7 Support Response To Telephone Requests

Peter Scott – Jet Propulsion Laboratory, California Institute of Technology

ABSTRACT

Demands for uninterrupted availability of systems that were hitherto not viewed as critical to the success of the enterprise has placed a burden upon support infrastructures for those systems. The small-to-medium help desk for a system that comprises part of an enterprise information system is under increasing pressure to provide round-the-clock support, while staffing budgets may not have caught up with the fiscal realities of covering second- or third-shift on-site personnel. While the volume of calls outside normal hours may be small, a guaranteed turnaround and response to emergencies is essential for many IT operations to gain the trust of their customers.

This paper describes a system for automated answering of the help desk telephone during non-peak hours, and for notifying on-call staff of emergencies within minutes. The system uses two voice-capable modems on a well-maintained computer for presenting to the caller a typical phone menu hierarchical menu which may include an option to record a message about some type of emergency. In this event, the system will notify the staff who are on call at that time for that type of emergency, contacting them by means they have specified in advance, such as voice telephone, numeric or alphanumeric pager, or other means. The system is currently undergoing testing in the Enterprise Information System File Service at JPL.

Motivation

After years of preaching the advantages of current technologies such as distributed systems, management acceptance comes with the price of making those technologies ubiquitous, easy-to-use, and reliable. The Enterprise Information System (EIS) Project at JPL was established in 1996 in an ambitious plan to establish a unified and coherent strategy and implementation of computing infrastructure to serve the laboratory's reengineering plans. Early adoption and roll-out of AFS (a distributed filesystem from Transarc), and a current migration path to DCE/DFS are some of the components of that implementation. Making this technology accessible and even transparent to a laboratory population highly diverse in its computer expertise and extremely heterogeneous in its platform usage demands a capable and responsive help mechanism.

The File Service element of EIS, having rolled out its technology fairly soon, set up an e-mail alias and a mnemonic telephone number for support. The nature of space missions and international collaboration means that JPL is open 24 hours a day, every day of the year. If flight projects whose personnel might work during off-hours were to embrace the new technology (a key aim for the reduction of overall costs and achievement of stretch goals), these projects and their personnel needed to know that their emergencies during off-hours would receive prompt attention. However, the start-up nature of the File Service meant that the number of customers was too small to justify funding for continuous on-site assistance.

In order to break this Catch-22, it was determined that on-call support would suffice, since response time could be fairly rapid. This led to a "pass the pager" scheme for the on-call person of the week (or month). This has a number of disadvantages:

- Only one person can be on call (the phone system will not forward a number to more than one pager at a time).
- Switching to another person requires that they meet in the same location at the same time as the desired handoff.
- If the pager is not working or out of range, the user does not know that their call has not been received.
- Granularity of on-call scheduling is extremely coarse, on the order of one day. If someone were able to answer calls from 6pm to 2am but no later, and someone else were able to answer calls from 2am to 6am but no earlier, for instance, one or both of them would have to compromise.
- If the support person's schedule changes after they leave work, handing off to another person can be difficult.
- Completeness verification and reporting of the on-call schedule is decoupled from the page forwarding itself and therefore subject to inaccuracy.

It is not acceptable to require customers to memorize multiple numbers, or use different numbers at different times, or look up the current phone number over the web (since the very reason they may be calling is to report that the network is down!).

Solution

What is needed is a mechanism for connecting the calls coming in to a single number with the multiple different and time-dependent methods of contacting on-call staff. A large enough organization could afford to staff the helpline with a person around the clock who need know no more than who to reach for a particular problem at any given time, and how. Some organizations cannot accommodate the budget necessary to pay for second- and third-shift personnel, and automation suggested itself in this case.

We proposed and constructed a system for automated answering of the help desk telephone during non-peak hours, and for notifying on-call staff of emergencies within minutes or seconds. No on-site personnel are needed provided that the on-call staff at any time are within a suitable distance for emergencies that cannot be resolved from wherever they are at the time. The scheduling of the on-call staff can be done by the staff themselves, and reports generated from the on-call schedule highlight any gaps in coverage. Nothing in the design of the system is inherently specific to the particular project it serves.

Requirements

The requirements we imposed on the system were as follows (KEY: **R**: Required for initial system; **RF**: Required for some future system; **D**: Desirable for initial system; **DF**: Desirable for some future system):

Schedule

- **R** – Maintains a user-editable schedule of who is on call at what times.
- **D** – User-friendly and quick to use interface for specifying when you are on call.
- **R** – Records incoming calls and makes outgoing calls to people on the schedule.
- **R** – System allows for more than one person to be on call at a time.
- **R** – System can provide a map of coverage showing who is on call for what reasons during specified upcoming period.
- **RF** – System allows users to be on-call at different times for different reasons.
- **RF** – System allows users to specify different actions to be taken when they are called for different reasons.
- **RF** – System supports multiple schedules specifying different sets of people to be called in response to caller selecting different options.

Input

- **D** – System will take a message even if caller does not press any touch tones.
- **R** – Ability to hook into programs that monitor system health and automatically call when malfunction detected.
- **D** – System can be triggered by e-mail.
- **DF** – System has API for triggering from Perl scripts.

Output Methods

- **R** – User can specify how they are to be contacted: voice phone, numeric pager, alphanumeric pager, e-mail.
- **R** – System can make multiple outgoing calls to different numbers and devices to contact the same person.
- **DF** – System can take alternate action in the event that a phone number on outgoing call does not answer.
- **D** – System allows user to specify multiple methods for contacting them.
- **R** – System allows custom audio to be played according to user specification in schedule.
- **D** – System allows users to record their own custom audio.

Paging – Alphanumeric

- **R** – User can specify the display message in the case of alphanumeric pager.
- **RF** – System can make outgoing calls to human-operated paging service.

Paging – Voice

- **DF** – System allows user to specify and/or record audio to be played to human-operated paging service.
- **RF** – A user-selectable option when calling out is to play the incoming message to the answerer.
- **DF** – A user-selectable option is to play only the first N seconds of the incoming message to the answerer.

E-Mail

- **D** – System can send e-mail as part of a response cycle.
- **D** – E-mail part of response is configurable by user.
- **D** – System can send e-mail for every request, configurable by administrator, in addition to any e-mail sent out to people on call.

Menu System

- **D** – Menu system provides a tree which can be traversed with touch tones, providing information and taking calls.
- **D** – Menu output can be preempted by caller pressing a valid touch tone.
- **RF** – System allows reason for call to be a function of where in voice menu system caller selected an option to record message.

Commercial Products

The first time we conducted a search for COTS products to perform this task, we were unable to find any costing under $100,000 - we did not investigate systems costing over $100,000 which might have had these capabilities. We found several inexpensive packages offering phone menu capabilities for personal computers, but not providing notification to voice, pager, and email according to a schedule.

A more recent search conducted for this paper turned up the Tel Alert product from Telamon, Inc., [6] which performs the phone menu functions in addition to status monitoring and contacting by pager, and has some scheduling capability; although it requires proprietary hardware it is worth evaluation before implementing an in-house solution. VoiceGuide from Katalina Technologies [7] performs most required functions (without the scheduling sophistication), plus many others (such as caller ID detection), at a very low price, but on Windows only. It could be used to call out to an external program which implemented the scheduling algorithm (perhaps remotely on a Unix box), although if the helpline application requires a high degree of robustness, most people would not choose to run it on a Windows machine.

Among freeware products, tpage [5] allows scheduling of people to be paged but not with the same flexibility and readability that this system provides, and only contacts pagers, not voice phones.

Construction

The key to the system is a voice-capable modem such as the ZyXEL Omni 288. A daemon waits for an incoming call and when one arrives, presents a typical hierarchical phone menu to the caller (“Press 1 if you know the NAME of...”). In fact, the daemon is simply playing a series of sound files that have been placed in a particular directory and given names corresponding to the keys the user should press to select them. Subdirectories containing more sound files are used to represent deeper levels of the menu, and the

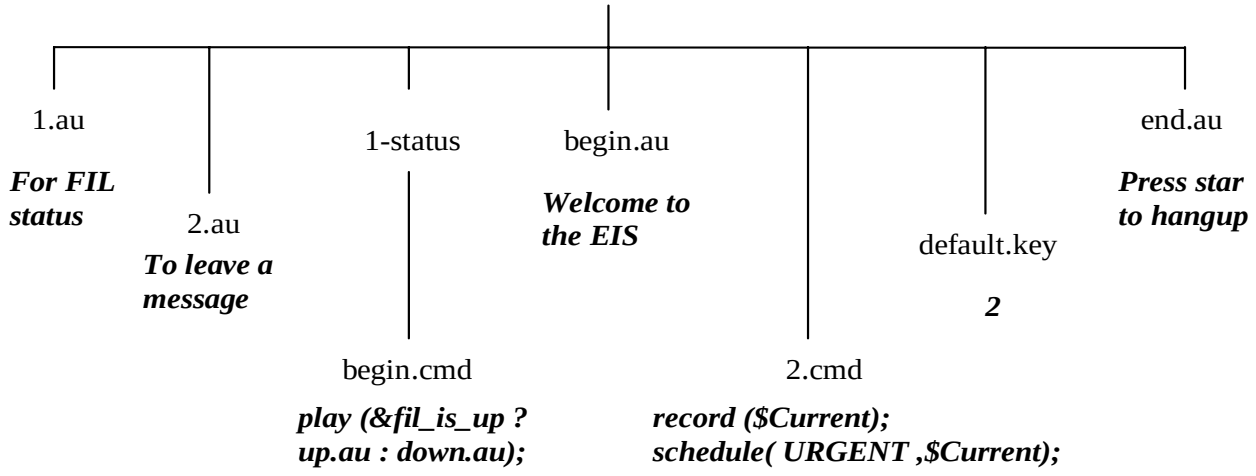

Figure 1: Menu directory hierarchy.

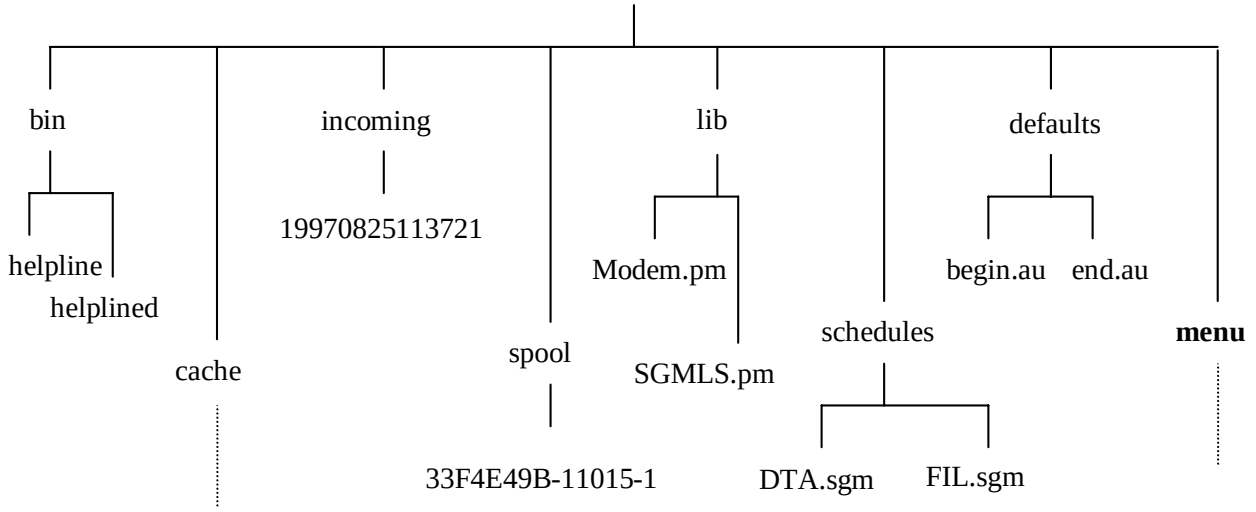

Figure 2: Tool hierarchy.

traversal follows some simple rules for checking the existence of files or directories with names matching particular expressions.

The directory hierarchy of the this menu looks something like Figure 1. The algorithm for processing this tree is very simple; see Figure 3. The directory tree for the phone menu is part of the hierarchy for the tool, which looks something like Figure 2 (not all files displayed).

An advantage of this approach to constructing a phone menu system is that it is so easy to create additional levels that an entire support staff can be given write access, allowing them to create new sound files and menu levels if they wish. This distributes the job of maintaining the phone menu system among the people with the domain expertise appropriate to each particular part of the hierarchy.

In addition, responses can be generated on the fly according to information derived at call time. The menu can call scripts that can execute arbitrary Perl commands, for instance, to ascertain the status of various institutional servers, and modify the playback content accordingly. The daemon runs each script with a Perl `eval` command so that a subroutine library is imported into the namespace of the script enabling it to perform certain phone menu-specific operations

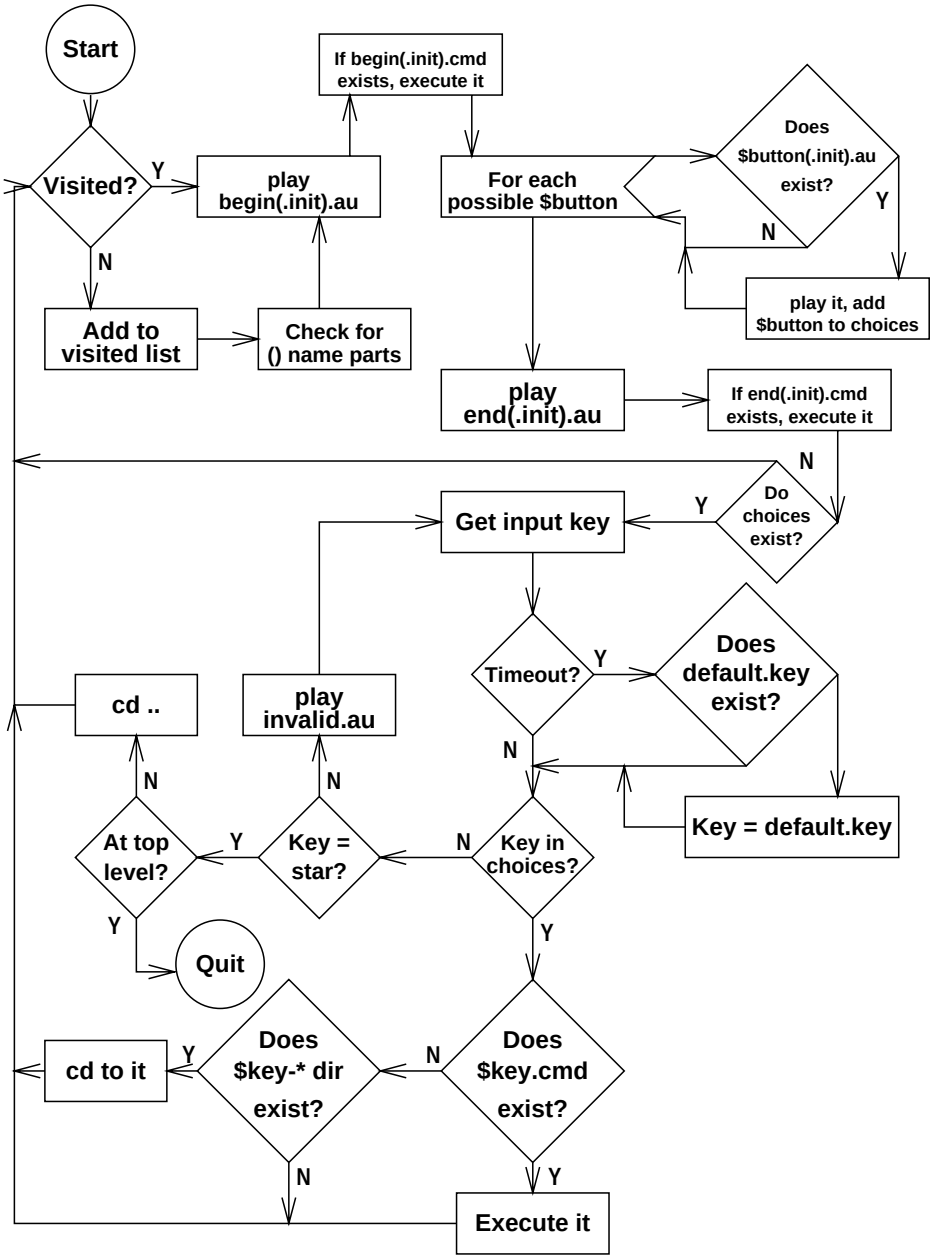

Figure 3: Tree processing algorithm.

easily. The key point here is not that a phone menu system can respond with dynamic information, but that the design of this implementation allows it to be done so easily using only standard tools (simple Perl, and a microphone hooked up to a recording program).

One operation a caller may wish to perform is recording a call for help. The phone menu interface for handling this is very simple: at the node where the corresponding sound file says, "If you are experiencing trouble with..." the associated script contains:

```
$record_file = nextname();
record ($record_file);
schedule (<situation>,
    $record_file, "FIL");
```

```
<Day Name="Weekdays">
 <DayRange>
  <DayFrom>   <WeekDay Day="Mondays">  </DayFrom>
  <DayTo>   <WeekDay Day="Fridays">  </DayTo>
 </DayRange>
</Day>
<Day Name="Holidays">
 <DayRange>  <OneDay Name="NewYearsDay"> </DayRange>
 <DayRange>  <OneDay Name="MemorialDay"> </DayRange>
 <DayRange>  <OneDay Name="IndependenceHoliday"> </DayRange>
 <DayRange>  <OneDay Name="Thanksgiving"> </DayRange>
 <DayRange>  <OneDay Name="Christmas"> </DayRange>
</Day>
<Day Name="WorkWeek">
 <DayRange>  <OneDay Name="Weekdays"> </DayRange>
 <Except>  <OneDay Name="Holidays"> </Except>
</Day>

<Situations>
 <Situation Type="AFSDown">
 <Situation Type="WWWDown">
 <Situation Type="Urgent">
</Situations>

<Profile Name="George1" Priority="CallOnlyIfAlone">
 <Command Number="44321" Type="Voice">  <PlayFile>
 </Command>
 <Condition Type="DoOnFailure">
 <Command Number="97907560" Type="Voice">
  <PlayFile Name="~george/audio/sounds/pageme-intro.au">
  <PlayFile>
 </Command>
</Profile>

<Schedule>
 <ProfileName Name="Glenn1">
 <Situation="AFSDown"> <Situation="Urgent">
 <DateTimeRange AllDay="Yes">   <OneDay Name="GlennWork">
 </DateTimeRange>
 <DateTimeRange>  <OneDay Name="WeekEnds">
  <TimeFrom Hour="8">  <TimeTo Hour="18" Minute="30">
 </DateTimeRange>
 <DateTimeRange>  <OneDay Name="GlennHolidays">
  <TimeFrom Hour="9">  <TimeTo Hour="17">
 </DateTimeRange>
</Schedule>
```

Listing 1: Scheduling examples.

where <situation> is a mnemonic for the type of emergency being reported (any number of situations is possible). The first line acquires a unique name for the incoming sound file, the second guides the caller through a phone menu recording menu which allows them to review, cancel, or send a message, and the third notifies the scheduler of an occurrence of that particular type of emergency.

The spool directory contains files queued for processing by `helplined`, the daemon which handles outbound communications. A spool file looks like:

```
SITUATION: AFSDOWN
PROFILE: PETER
CONDITION: NULLCONDITION
COMMAND_TYPE: VOICE
COMMAND_NUMBER: 42246
COMMAND_PIN:
COMMAND_ADDRESS:
COMMAND_PLAYFILES: ~peter/audio/
    sounds/attention.au ..
    ./incoming/19970825113721
COMMAND_MESSAGE:
```

On-call Scheduling

The on-call notifier and scheduler is an independent component which could be implemented entirely separately from the phone menu system; it is not necessary for emergency notifications to come from the phone menu system, since any process which executes the schedule() subroutine can make such a notification. Thus it is possible to cause on-call notification in the event of automatically-detected or triggered events such as a UPS alarm signalling power failure, a network health monitor alarm, or SNMP events.

When an emergency notification occurs, the scheduler parses a schedule file to see who is on-call at that time for that type of situation. The schedule file contains statements describing four entities:

- **Date Ranges** - assigning identifiers to a set of dates;
- **Situations** - a list of valid situation identifiers for the schedule;
- **Profiles** - assigning identifiers to methods of reaching a particular person;
- **Schedules** - associating certain situations with certain profiles during specified times over certain date ranges

When it has determined which profiles are active for the given situation at the current time, the scheduler runs the profile to contact the on-call support person by pager, telephone, or whatever other method is specified in their profile.

The schedule file is written in SGML; a DTD was created to describe schedules. This provides two advantages: firstly, that the input and editing of the file can be done through an SGML editor which guides the user through the creation of a syntactically valid entry, and can be customized to provide a high degree of user-friendliness; thus, entries can be made by the on-call support personnel themselves. Secondly, that enforcing of the required syntax by the SGML editor leaves almost no syntax checking required of the application. Whenever an emergency notification occurs, the schedule file is read through the nSGMLS [3] parser with the SGMLS.pm module [2] and rejected if a syntax error is found (of course, the file is also validated every time it is edited).

The SGML schedule in Listing 1 contains examples of each of the four statements listed above. The date range definitions of the individual holidays and the personal date ranges have been elided to save space.

The SGML is not designed for human parsing nor editing, and fortunately, neither is necessary. There exist a number of products which read the DTD and provided structured editing of the source, guaranteeing that the syntax is valid. A freeware example is the SGML module for EMACS. Some Commercial Off-The-Shelf (COTS) products also allow ‘‘style sheets’’ to be constructed which hide the tags completely from the user, providing WYSIWYG editing like modern HTML editors (e.g., Arbortext’s Guided Editor). When formatted for human-readable output, the statements in the schedule can be more easily illustrated in some more examples:

Date Ranges

```
DAY ChristmasEve: 12/24
DAY Christmas: FROM ChristmasEve
          TO BoxingDay
DAY BobWork: WorkWeek EXCEPT
          BobVacation
```

The first example assigns a specific date (which can be qualified with a year if necessary) to the identifier `ChristmasEve`, the second example assigns a range of already-assigned dates to another identifier, and the third assigns a range of dates excluding a predefined subrange. The first two lines would have been input by an administrator to define certain well-known dates; since the year is they will not need to change fixed-date definitions such as Independence Day from year to year. The third line would have been input by Bob, a member of the support staff, defining a date range corresponding to his default work week: the administrator-defined workweek (weekdays except for holidays), minus his personal vacation (defined elsewhere).

Profiles

```
PROFILE Bob1: VOICE 5557458
   (MESSAGE       ~bob/sounds/
   oncall1.au, $current),
IF FAILED PAGE 5553453 (PIN
    3894634)
PROFILE Jack3: ONLYIFALONE
```

```
IXOPAGE 5550532
(MESSAGE "I've fallen and I
   can't get up")
PROFILE Jim2: PAGE 5550974 (PIN
      63459023)
   THEN EMAIL jim@tarkus (MESSAGE
      "It's dead, Jim")
```

The first example tries to call Bob at his home, playing first an introductory sound file recorded by Bob which can both explain things to his wife if she answers, and is timed to go past the outgoing message on his answering machine if they're not in, and then plays the message let by the caller; if there is no answer at that number, it calls Bob's numeric pager. The second example calls Jack on his alphanumeric pager, but only if he's the only one on call at the time. The third example calls Jim on his numeric pager, and then, regardless of the success or failure of the page, sends him an email message.

Schedules

```
SCHEDULE Bob1, Jim2 FOR PWRDOWN
    20:00 - 05:00 DURING WorkWeek
SCHEDULE Jack3 FOR NETDOWN 09:00 -
    18:00, 20:00 - 22:00 DURING
    Weekends
SCHEDULE John FOR ANYTHING 07:00
    - 01:00 DURING AllWeek
```

The first example attaches the profiles `Bob1` and `Jim2` to a late-night/early morning slot during the work week (previously defined as being `AllWeek EXCEPT Holidays`, where `Holidays` was defined as the union of the defined holidays), in the case of power failures. The second example attaches the profile `Jack3` to two time slots on weekends for network failures. The third example attaches John's profile to a punishing schedule in case of any situation.

We can now do two things with this schedule file: we can produce a report showing the on-call

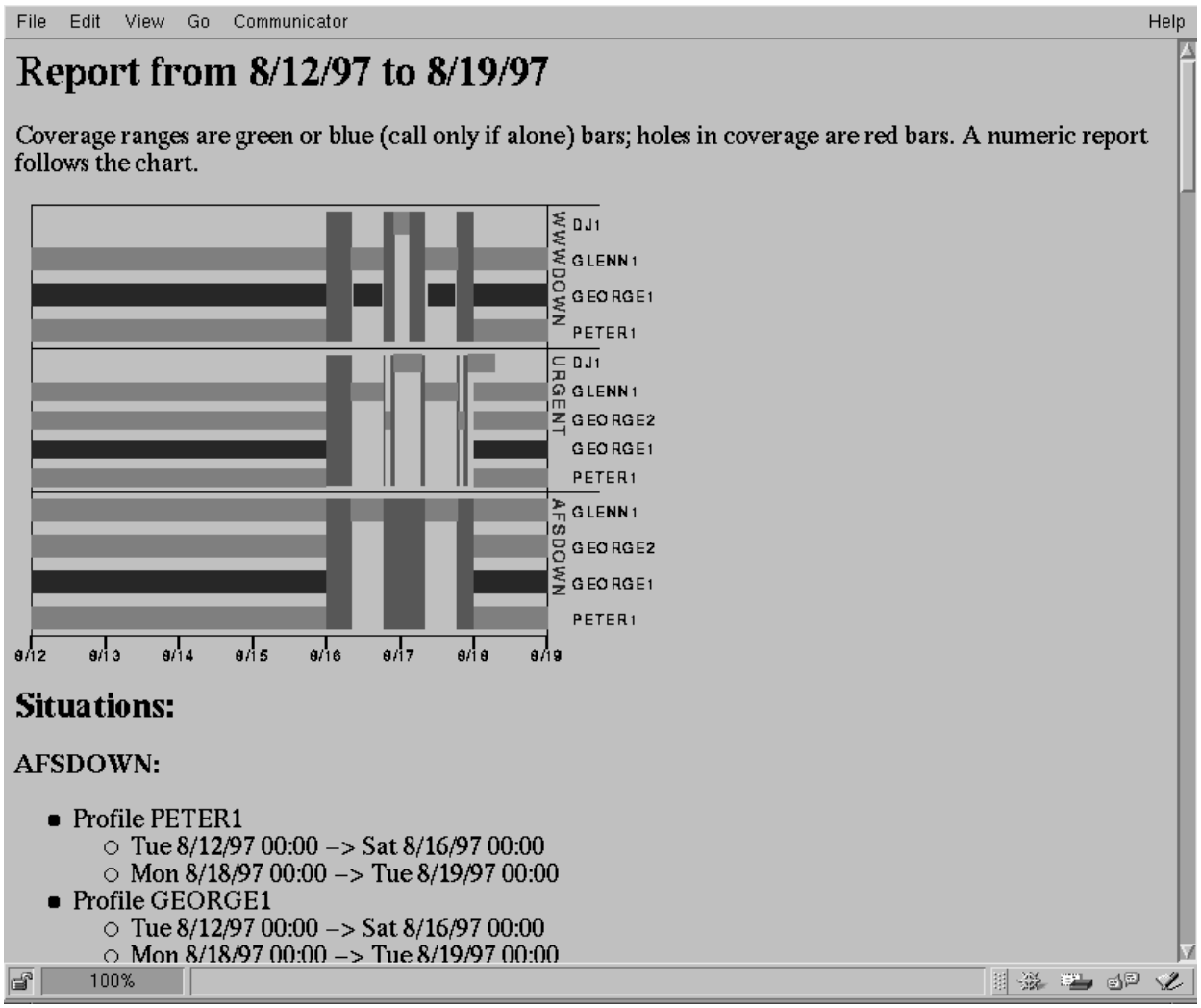

Figure 4: Web Page.

```
:            Toggling DTR to reset modem
:            Configuring port parameters.. (/dev/cua/a / 38400 bps)
:            Initializing Modem
Modem Cmd:   Command Mode?                   ATE1V1 -> OK
configure:   Modem was in commmand mode.  Now in command mode.
Modem Cmd:   Initialization String           AT&FS2=255S0=0S13.2=1N0&H4 -> OK
Modem Cmd:   Switch to VOICE mode            AT+FCLASS=8 -> OK
Modem Cmd:   Default settings for voice mod AT+VIP -> OK
Modem Cmd:   Turn up DTMF sensitivity        AT+VDD=3,8 -> OK
Modem Cmd:   Enable XON/XOFF Flow Control    AT+FLO=1 -> OK
:            Modem Initialized
Modem Cmd:   Ping                            AT -> OK
main_loop:   Waiting for a call..
main_loop:   MODEM: [
main_loop:   Waiting for a call..
main_loop:   RING detected!
Modem Cmd:   Answering VOICE call            AT+VLS=2 -> VCON
handle_call: Call received.
handle_call: Voice::init complete.
menu:        Menu /afs/jp1/group/ets/src/helpline/menu
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/menu/begin.init.au.zyxel
Modem Cmd:   Start Playing                   AT+VTX -> CONNECT
play:        Playing voice file .../cache/.../menu/begin.init.au.zyxel
play:        Transmission Buffer Underrun
Modem Cmd:   Stop Playing                    <DLE><ETX> -> OK
say_choices: Running script begin.init.cmd
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/menu/1.au.zyxel
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/menu/2.au.zyxel
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/menu/star.au.zyxel
Modem Cmd:   Start Playing                   AT+VTX -> CONNECT
play:        .../cache//afs/jp1/group/ets/src/helpline/menu/1.au.zyxel
             .../cache//afs/jp1/group/ets/src/helpline/menu/2.au
             .../cache//afs/jp1/group/ets/src/helpline/ menu/star.au.zyxel
Modem Cmd:   Stop Recording                  <DLE>! -> OK
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/default/record/1.au.zyxel
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/default/record/2.au.zyxel
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/default/record/3.au.zyxel
say_choices: Using .../cache//afs/jp1/group/ets/src/helpline/default/record/pound.au.zyxel
Modem Cmd:   Command Mode?                   ATE1V1 -> OK
play:        Modem was in commmand mode.  Now in command mode.
Modem Cmd:   Start Playing                   AT+VTX -> CONNECT
play:        Playing voice file .../cache/.../default/record/1.au.zyxel
play:        Playing voice file .../cache/.../default/record/2.au.zyxel
play:        Playing voice file .../cache/.../default/record/3.au.zyxel
play:        Playing voice file .../cache/.../default/record/pound.au.zyxel
Modem Cmd:   Stop Playing                    <DLE><ETX> -> OK
process_keys Waiting 5 seconds..
MODEM:wait:        DTMF digit: "#"
menu:        Ready to process keypress: #
menu:        Valid choice.
process_keys Running script pound.cmd
run_script:  send message
33F4E5A6-11173-1 ( PETER )
```

Listing 2: Logfile excerpts.

coverage for any period, and we can determine who is on call for a given situation at a given time. The second capability is used when an emergency call comes in or the scheduler is otherwise triggered; the first can be used by the facility manager for planning purposes and to demonstrate to senior management proven 24x7 coverage. The same module which determines which profiles are active at the current moment has routines for producing a report between two dates and rendering it in HTML; see Figure 4 for example.

The horizontal bars on the `GEORGE1` tracks are blue; the other horizontal bars are green. The vertical bars are red, indicating gaps in coverage when no profile is active. This report shows that there are three situations defined in the schedule file, labelled `AFS-DOWN`, `WWWDOWN`, and `URGENT`.

Implementation Details

The system is implemented using custom Perl 5 code for answering the phone and calling out. We experimented heavily with vgetty from Gert Doering's mgetty+sendfax package [1] and liked the scripting capabilities, but ultimately were confined by problems with data synchronization and capabilities for extension in the version we had at the time. Our system nevertheless suffers from synchronization problems where sometimes keypresses are not detected or the modem goes into an unintended state, but we believe these are tractable problems. The schedule parsing is done with custom Perl 5 code using David Megginsons's SGMLS.pm SGML-parsing module [2] and James Clark's nsgmls SGML parser [3]. Turning the schedule file into a report or a list of profiles on-call at the time was made much easier with Perl 5's object-oriented features. Where sound files do not exist for messages to be played to the caller, voice synthesis is done with rsynth [4], by Nick Ing-Simmons (usually used for error messages).

A logging capability allows the operator to view the progress of the system. Parts of a log of a session recording a call for help are reproduced in Listing 2.

Experience in Use

The most difficult part was communicating with the modem; we have not yet figured out how to handle data overruns or why touch tones are sometimes missed. We heard at the Perl Conference that a module for modem control is in the works and that may solve this problem. The tests for failure of an outgoing call are incomplete; we would prefer more sophisticated checking for whether a call was successful.

Conclusions

The helpline system demonstrates that the tools exist to construct an apparently complex system in a domain hitherto considered the province of COTS-ware using freeware components. The use of SGML to specify the grammar of an input file provides a significant gain in user-friendly editing and syntax checking.

Acknowledgments

Josh Wilmes, of RPI, developed the modem-answering code while working as a co-op student at JPL.

The work described was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Author Information

Peter Scott left Cambridge University with an MA in CS in 1983 and has been working at the Jet Propulsion Laboratory in Pasadena, California, since 1986. For the last two years he has been lead development engineer on the File Service Element of the Enterprise Information System. He can be reached at M/S 301-342, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109 and at <pjscott@euclid.JPL.NASA.GOV> .

References

[1] mgetty+sendfax: http://www.leo.org/~doering/mgetty/index.html .

[2] SGMLS.pm: http://www.uottawa.ca/~dmeggins/SGMLSpm/sgmlspm.html .

[3] NSGMLS: An SGML System Conforming to International Standard ISO 8879 – Standard Generalized Markup Language. http://www.jclark.com/sp/nsgmls.htm .

[4] Rsynth: ftp://svr-ftp.eng.cam.ac.uk/comp.speech/sources/rsynth-2.0.tar.gz .

[5] tpage: ftp://gumby.dsd.trw.com/pub/tpage.tar.Z .

[6] Tel Alert: http://www.telamon.com/prodinfo/telalert.html .

[7] VoiceGuide: http://www.ozemail.com.au/~katalina/vgidx.htm .