PRONTO - A SYSTEM FOR MAINTENANCE OF DUTY ROSTERS OF SHIFT PERSONNEL

Erik W. Tilanus, Anton A. Bäcker, Henk Esselman
KLM Royal Dutch Airlines

Introduction

Air transport involves a lot of shift work. Several departments of an airline, especially those at the airport, serve 24 hours a day; many others from 06.00 in the morning till 24.00 h. In contrast to a factory, the workload in these departments is not stable during the day but varies widely due to the flight schedules at certain time of day.
Matching the amount of staff available at any moment to the momentary workload is a must. Unsufficient staff results in flight delays, and overstaffing wastes money. This matching is done by assigning staff to several different rosters, with different shift times.
Consequently, roster maintenance is complex and labour intensive.

Duty Roster Maintenance

Basically a roster consists of a survey of all shifts of all people of a department during a week. (Fig.1)

SEQ.NR	SUN	MON	TUE	WED	THU	FRI	SAT
1	3	2	1	COM	2	2	1
2	OFF	COM	COM	3	3	3	2
3	OFF	3	3	2	1	1	OFF
4	1	1	2	1	COM	COM	3
5	3	2	1	COM	2	2	1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

LEGENDA: SHIFT 1: 06.00-14.30
SHIFT 2: 09.00-17.30
SHIFT 3: 14.00-22.30
COM WEEKEND COMPENSATION

figure 1: the roster

The duty roster is the next step in the process: names are assigned to each line of the roster. Each week all the names are shifted one line up and the first name becomes the last. (Fig.2)
Once people are assigned to a specific line of the roster they start coming with change requests: they want holidays, they don't want an evening shift on their birthday etc.
All these leave requests, shift changes etc. should be administered and the resulting available manpower checked against the minimum requirements as imposed by the flightschedules. The quality standards for this process are very high, since this is the process that controls the actual workforce present and provides the basis for shift pay.

WEEK JAN,10-JAN,16

NAME	SUN	MON	TUE	WED	THU	FRI	SAT
JANSEN	3	2	1	COM	2	2	1
BUIS	OFF	COM	COM	3	3	3	2
ZEVENUM	OFF	3	3	2	1	1	OFF
KRAAY	1	1	2	1	COM	COM	3
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

WEEK JAN,17-JAN,23

NAME	SUN	MON	TUE	WED	THU	FRI	SAT
BUIS	3	2	1	COM	2	2	1
ZEVENUM	OFF	COM	COM	3	3	3	2
KRAAY	OFF	3	3	2	1	1	OFF
OKKERS	1	1	2	1	COM	COM	3
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

figure 2: the duty-roster

Figure 3 presents the overall process in conjunction with neighbouring processes. Automation of the duty roster maintenance has been discussed for several years. The process seems well defined and therefore suited for automation.

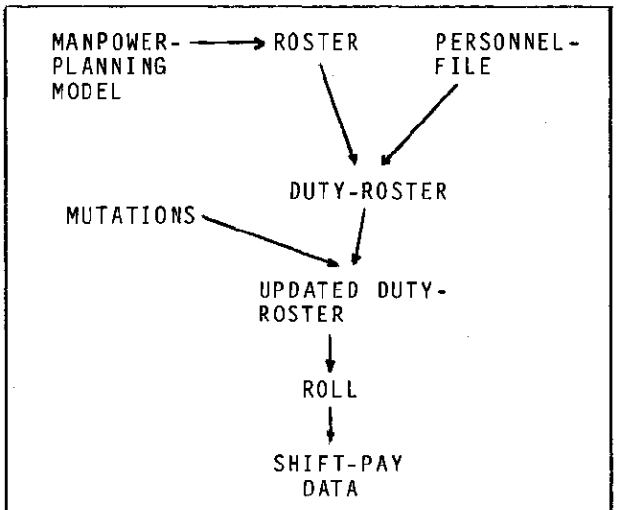

figure 3: the roster maint. process

The advantages of automation are obvious. There is some direct cost saving by reducing the work involved in the process, but especially the indirect saving potentials are very attractive. An appropriate registration of everything happening between setting up the duty-

roster and the actual service is a good basis for optimization of manpower planning and roster development phases. However, several complicating factors exist: there are many different rosters and each department has its own "roster folklore": there are many ways to construct and publish a roster within the constraints of labour agreements and union guidelines.
All these variations must be taken into account.

Project Setup

The decision to develop a system for roster maintenance was finally reached in August 1981. The system was named PRONTO, an acronym for Ploegen Rooster ONderhoud TOepassing - Duty roster maintenance application. It was supposed to be a terminal oriented on-line system and was to be ready by November 1981. The available elapsed time made it clearly impossible to develop an IMS-based system. Besides, the many uncertainties, mostly resulting from the "folklore", made it attractive to adopt a prototyping approach. Therefore it was agreed to use TSO/SPF-II for the terminal-computer communication and dialog and SAS for the application development and file management. The release of SAS-FSP occurred just in time to simplify the data-entry part of the system.

System description

The user who logs on to PRONTO will receive a selection menu on his screen from which he or she may select the desired function. There are 12 functions available for entering personnel and roster data, entering change or leave requests, display of assignments etc. (Fig.4)

```
-----PRONTO SELECTION MENU-----
YOUR CHOICE ===⇒

1 - CREATE/UPDATE PERSONNEL DATA
2 - CREATE/UPDATE SHIFT CODES
3 - CREATE/UPDATE ROSTER
4 - ENTER MUTATIONS
5 - DISPLAY DUTY-ROSTER
6 - DISPLAY ROLL
7 - PRINT DUTY-ROSTER
8 - PRINT ROLL
9 - PRINT SHIFT CODES
C - NEXT MENU PAGE
X - LOGOFF
```

figure 4: SPF - primary selection

After specifying the required function the system responds with another TSO/SPF data frame for entering the parameters for the selected function, such as a date and a rostercode for a duty roster of a specific week. Behind all this are "skeleton SAS programs". The "file tailoring services" of SPF will resolve the symbolic parameters in the selected skeleton program. Programs that produce output on the screen are run in the foreground using the CALL command, with SYSIN assigned to the skeleton file. Programs that produce printed output are run in background, by including JCL statements in the skeleton and executing them via a SUBMIT command. The create and update file programs (functions 1-4) contain a PROC FSEDIT for on-line display and update of the dataset.
All intermediate output is suppressed by specifying the NOSOURCE and NONOTES options and assigning the SYSOUT-file to DUMMY. The only result that remains visible on the screen is that the curser advances one line down for each DATA or PROC step. This could have been avoided, but because of the poor responsetimes it is left this way: it tells the users that the system is still working for them.
The final results of the program are written to an OS dataset and presented on the screen by means of PROC FSPRINT.

File structure

The file structure is in fact very straightforward. (fig.5)
There is no permanent file for the duty roster. The weekly duty roster is contructed each time it is requested. The permanent SAS files are:

- 2 personnel files: one sorted by name and one by roster sequence number. This saves extra PROC SORT's during the execution.
- A roster file: contains the roster pattern in coded form e.g. shift 1, shift 2 etc.
- A roster format file: translates the shift code into actual shift start and finish times.
- A mutation file: contains all the changes compared to the standard duty roster.

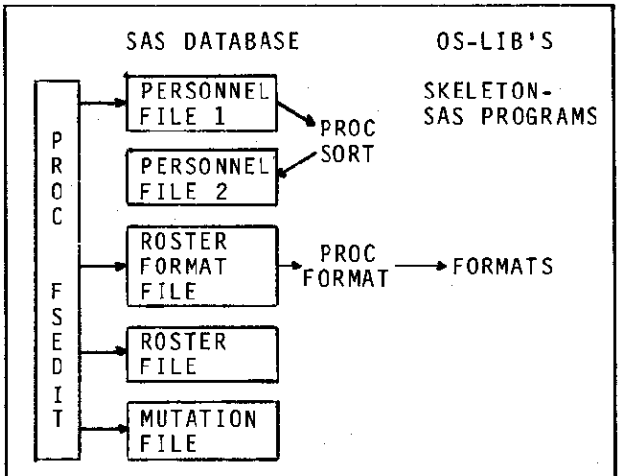

figure 5: the file structure

These files are maintained using PROC FSEDIT. The whole dialog to start FSEDIT (e.g. which dataset, which FSEDIT function) takes place behind the scenes. The right information is passed to SAS via the skeleton program and the user will directly get the edit screen on the terminal (fig.6). The second personnel file is automatically generated after finalization of the updates of the first one. The roster format file is only an intermediate result: Storing values (roster-shift codes) and their explanation in a SAS data set makes it easy to maintain, but what is needed here are stored, ready to use, formats. After updating the roster format file, the file is therefore copied to a temporary OS dataset, in PROC FORMAT source statement form. Thereafter SAS is run with this file as input and the stored formats are ready.

```
 EDIT SAS DATA SET:WORK.MUT1   | SCREEN 1
COMMAND ===>                  |---------
                              | OBS NEW

       ENTER MUTATIONS
DEPT.:SPL/PH        ROSTER:47

MUT.CODE: __      LE(AVE) SI(CK) SW(AP)
MUT.TYPE: _       + = OWN REQUEST
          _       # = KLM REQUEST
START DATE: _____   DD MM YY
END DATE  : _____   DD MM YY

PERS.NO.: _____
NAME    : ______________
PERS.NO.: _____  SWAP ONLY
NAME    : ______________
```

figure 6: FSP screen

Application Logic

Behind each possible selection from the menu is a SAS program. These programs are all more or less similar, except those that are merely for data entry. The first processing step when a duty roster or a roll is requested is always the matching of the right name to the right roster line. The first name to appear on the roster is calculated using a simple algorithm based upon the roster sequence number in the personnel file and the SAS week number of the requested week (INTCK function against 0). The personnel file is then rearranged with this name as first observation. The actual coupling with the roster is then a matter of merging. At the end of that DATA step there is a dataset containing the duty roster of the specified week. This file is then updated by the mutation file. The mutation file contains all changes for the whole season, so validity must be checked for each week. When a roll is requested, the proper day of the week is then abstracted from the updated duty roster. The presentation on the screen or on paper is a matter of simple PUT statements. The original assignments are saved across the processing and printed at the bottom to indicate the changes that occurred. This facilitates the necessary tracing if anything went wrong. The complete PRONTO application is about 600 SAS statements, 30 DATA steps and 15 PROC steps.

Experience

The selected approach turned out to be very successful. The combination of TSO/SPF-II, SAS and SAS/FSP appears to be a suitable prototyping toolbox. It required only 2 weeks work for two persons, one working on SPF and one on SAS, to create the first working prototype. The presentation of this prototype to the users happened to be a relevation for both users and developers. The users were surprised to see a recognizable system in such a short period. The developers were surprised by the disucssions that followed. That was the moment that the forementioned folkore came fully into the open. The prototyping cycle had to be repeated 3 times. The third prototype came so close to what was wanted that this one has been implemented after further refinement. The development of the first phase of the operational system took 5 manmonth to develop, including user manpower. A rough estimate for a similar development under IMS, using PL1 as programming language, has been made for the sake of comparison. It yielded 1.5-2 manyears! So the advantage of using SAS, from a development point of view, is obvious. The bill for this will be presented in the operational environment. While an average IMS terminal requires 0.5-0.7% CPU in the KLM hardware environment (IBM 3032) the PRONTO terminals require 1.5% CPU with peaks of about 2%. It has been estimated that about 0.9-1.2% of this is contributed by TSO/SPF-II. The SAS file management is powerful enough for this type of application: the files are limited to a few hundred observations and thanks to the full-track blocking it makes little difference that these files are processed sequentially. The space overhead due to header and history sections, however, account for 25 to 40% of the total required space. Response times are worse than normally to be expected from IMS. Depending on the type of transaction, 10-20 seconds is not unusual. Much of it may be caused by the fact that the system is overloaded and that TSO has not the highest dispatching priority. Whether the control transfer between DATA steps and PROC steps also contributes to these response times is not clear. Within FSEDIT the response times are much better: next or previous

observation is normally 1-3 sec. To explore this more deeply, LOCATE experiments were carried out, on a data set of about 24,000 observations and 15 variables. A LOCATE of an observation near the end of the dataset resulted in 5-10 seconds response time.

Concluding remarks

The TSO/SPF-II facility combined with SAS and SAS/FSP form a powerful set of tools to construct on-line systems of limited size. During the development of PRONTO we encountered several possibilities for future improvements to these tools, e.g.:

- From the point of CPU-saving it would be attractive to have the possibility to store compiled SAS code. The present method involves a compilation during each transaction.
- When using PROC FSEDIT in applications such as PRONTO it is desirable to have more authorization levels in the FSEDIT functions, e.g. certain users should not be able to change screen layouts or PF-key definitions. It should also be possible to define certain fields as output only.
- The method used to create the roster decoding formats is rather complex. A possibility to read a SAS dataset as value definitions in PROC FORMAT would simplify this rather considerably.

SAS is at the moment used as standard end user programming facility within KLM. Some 60 people from many different departments use it. Most of the work with SAS is data-analysis or report generation in relation to data coming from our major EDP systems. The advent of SAS/FSP opens up a new area of applicability for SAS. It is something many users were waiting for. The development of PRONTO demonstrates this new dimension, although we expect that the amount of technical knowledge required to develop such systems, prohibits development by the users themselves for the time being. It is expected that the first set-up has to be carried out by EDP-people, while the user department may be capable of maintaining the system. Less sophisticated applications of FSP are certainly within the reach of users. This has been proved already several times since the completion of PRONTO.

APPENDIX 1: CLIST FOR PRONTO

This appendix contains highlights of the CLIST used to run PRONTO. It starts with establishing the environment.

```
PROC 0 OPTIONS('SYSIN=INP NONOTES)
ISPEXEC VGET (ZUSER,ZTEMPF)
   SET &USER = &ZUSER
   SET &SFX  = &SUBSTR(5:7,&ZUSER)
ISPEXEC TBOPEN D8TAB&SFX WRITE
CONTROL NOMSG
FREE F(SYSOUT WORK INP SASLIB FT20F001)
CONTROL MSG
ALLOC F(SYSOUT) DUMMY
ALLOC F(WORK) TRACKS SPACE(20 10) NEW
ALLOC F(INP) DA('&ZTEMPF.') SHR
ALLOC F(SASLIB)
      DA('&USER..PRONTO.FMTLIB') SHR
ALLOC F(FT20F001) TRACKS SPACE(20 20) NEW
ALLOC F(LI8) DA('&USER..PRONTO.SAS') OLD
* Display the selection menu.
DISP:ISPEXEC DISPLAY PANEL(PRONT001)
SET &RC = &LASTCC
IF &RC GE 8 THEN DO
RETURN:DEALC ALL
   ISPEXEC TBCLOSE D8TAB&SFX REPLCOPY
   EXIT CODE(0)
   END
IF &RC = 0 THEN DO
   IF &OPT = X THEN GOTO RETURN
* Next follows the preperation of data
  frames for the required function.
AGAIN:IF &OPT = 1 OR  &OPT = 3 THEN DO
      ISPEXEC DISPLAY PANEL(PRONT011)
           .
           .
   IF &OPT = 5 OR &OPT = 7 THEN DO
      ISPEXEC DISPLAY PANEL(PRONT015)
      IF &LASTCC = 8 THEN GOTO DISP
      END
           .
           .
* Check on validity of personel
  and roster files.
      SET &ROSID = PERS&ROSNR
      ISPEXEC TBEXIST D8TAB&SFX
      SET &RC = &LASTCC
      IF &RC = 8 THEN DO
         WRITE NO PERS.ROSTER &ROSID
         GOTO AGAIN
         END
           .
           .
CONT: ISPEXEC TBSAVE D8TAB&SFX REPLCOPY
* Resolve symbolic parameters by means of
  file tailoring service.
      ISPEXEC FTOPEN TEMP
      ISPEXEC FTINCL PRONT01&OPT
      ISPEXEC FTCLOSE
* For functions with printed output:
  deallocate files and submit background
  job else execute SAS in foreground.
   IF &OPT = 7 OR &OPT = 8 OR &OPT = 9
   THEN DO
      DEALC ALL
      SUBMIT '&ZTEMPF'
       END
   ELSE CALL 'DO.SASLIB(SAS)' '&OPTIONS'
GOTO DISP
```

APPENDIX 2: DATA PANEL EXAMPLE

The next lines of coding give an example of SPF-DATA frame coding. The chosen example is from the display or print duty roster functions.

```
% PRONTO------- &KEUZE ------------------
%
%
%
+       DEPT.SUBCODE    %===> _KSTPL+
+
+       ROSTERNUMBER    %===> _NR+
+
+       BEGIN DATUM     %===> _DATUM    +
+
+
%
)INIT
  .HELP = TPRONTO
   IF (&OPT = 5)
      &KEUZE = 'DISPLAY DUTYROSTER'
   IF (&OPT = 7)
      &KEUZE = 'PRINT DUTYROSTER'
)PROC
  &NEW = N
  VER (&KSTPL,NONBLANK,MSG=PRN017E)
  VER (&KSTPL,RANGE,10000,99999,
       MSG=PRN013E)
  &KP1 = TRUNC (&KSTPL,4)
  VER (&NR,NONBLANK,MSG=PRN017E)
  VER (&NR,RANGE,1,99,MSG=PRN014E)
  VER (&DATUM,NONBLANK,MSG=PRN017E)
  VER (&DATUM,PICT,NN-NN-NN,MSG=PRN015E)
)END
```

APPENDIX 3: SKELETON PROGRAM

The coding contained in this appendix represent the skeleton program of the display or print duty-roster functions. The complete SAS-source is not included.

```
OPTIONS PS=22;
MACRO ARR1
      ARRAY CODB  (J) CB01-CB07;
      ARRAY CODBR (J) CBR1-CBR7; %
MACRO MDATE
      DATUM_ = '&DATUM';
      DATE_ = INPUT(DATUM_,DDMMYY8.);
      AWK = INTCK('WEEK',1,DATE_-1); %
MACRO SELECT
      ROSNR = KP2 * 100 + ROS ;
      IF VWK NE AWK OR ROSNR NE &ROSNR
      THEN DELETE ; %
MACRO PERSFILE PERS&ROSNR %
MACRO PRSNFILE PRSN&ROSNR %
MACRO BASRFILE BASR&ROSNR %
PROC PRINTTO UNIT=20 NEW;
  DATA WEEK1;
  MERGE LIB.BASRFILE LIB.PERSFILE;BY NO;
  ARR1
    .
    .
    .
PROC FSPRINT CMD ;
```