A GENERAL PURPOSE SOFTWARE SYSTEM FOR APPLICATIONS IN DIFFERENT FIELDS OF PHYSICS

by

Poul B. Ring
Danish Defence Research Establishment
Copenhagen, Denmark

ABSTRACT The paper presents the principle of a software system (FTSS) developed by the Danish Defence Research Establishment (DDRE) for applications in different fields of physics. Software is made up by independent modules exchanging data by standarized disc files. Special attention is paid to the production methods of new modules of the extensible system. Since the basic structure of all modules is the same a schedule for production of new modules can be established. Finally as an example an implementation of a ray-trace model is briefly described.

INTRODUCTION

The paper presents the principle of a software system (FTSS) developed by the Danish Defence Research Establishment (DDRE). The system has been in use since 1973.

The purpose was to make a standarized software system for a number of user groups working in different fields of physics. The intention was that common problems should be solved by a basic package of common software modules, but it should further be possible for the groups to make their own dedicated extensions. Furthermore the system had to be easy to extend for new demands.

These requirements led to the design of a software system with independent modules exchanging data by standardized disc files. Each module performs a specific task and new modules are added to the system depending on new demands. The system is interactive, but batch-facilities are also included.

For the time being (Aug. 1979) the system consists of about 100 modules in fields such as: Timeseries acquisition and analysis, simulation of differential equations, image processing, pattern recognition and a number of simulation models such as sound propagation models, sensor simulations etc.

The system is implemented on a HP-2100 computer using the monitor DOS-III. At present the system is implemented in the environment of the RTE-IV monitor. The software is implemented in Fortran IV.

1. FILE STRUCTURE

Data are stored in about ten types of standarized disc files, e.g. timeseries, spectrum, histogram, digital filter, image and record files. The last type of a file represents different types of tables of complex structures. These files are suited to simulation models.

The overall structure of the files is a division in a descriptor and a data section (See fig. 1). A descriptor contains information, name of module which created the file etc. An example of a descriptor for a timeseries file is shown on fig. 2.

2. STRUCTURE OF PROGRAMS

Software is divided into modules placed on three levels. (See fig. 3). The only module on level one is named MAIN. It is the only legal entrance and exit to the FTSS system modules. The most important task of MAIN is to initialize a system data area containing information about the state of the system, e.g. execution in interactive or batch mode, bookcounting of accessed files etc.

All processing modules are placed on level two. On level three a display module and an editor module (editing of descriptors) for each type of the FTSS-files are found.

When FTSS is entered by MAIN there is random access between the modules. The significance of the leves is: If you return to a module from a higher numbered level the module will be in the state as when it was left. In other cases a module is initialized when called.

All modules use only FTSS disc files when accessing data on the disc. To each type of file a collection of accessing subroutines is available.

3. THE COMMAND SELECTOR CONCEPT

Communication between a user and a module is performed by a command selector, which is a scheme containing a head-line and a number of indexed lines.

The head-line identifies the module. The indexed lines contain values of parameters and processing possibilities. The first step is to select the parameter-lines by typing their index and afterward change or insert the values of the parameters

as desired. Parameters are numbers, text strings or file names. Each time a parameter is changed the terminal is erased and the command-selector is displayed with the new value. (Duration one sec.). Processing of data is performed by typing the index of one of the processing-lines, which are marked by a *. Examples of command-selectors are shown on fig. 4.

Each module has one and only one command-selector.

4. DESIGNING NEW MODULES

Several years experience has shown that the standardization of the communication by means of command-selectors is an advantage for the users of the software because the modules seemes very alike. But also for the designer and programmer the architecture gives advantages.
The main reason is that a fixed schedule is used in production of new modules. Following steps are performed.

- Design phase.
  - Seperate problem in modules.
  - For each module write down the command-selector.
  - Design (or copy) processing algorithms.

- Implementation phase.
  - Use the module generator program. This is a program having input as a brief description of a command-selector.
    Output is a Fortran program containing the communication part of the new module.
  - Implement the processing algorithms and merge them (as subroutines) into the generated program.

- Test phase.
  - Test each subroutine and repeat the examples for the whole module. (The classical colution).

In the design phase the command selector concept is a convenient tool in the discussions between designers and users having a demand for new software, because a set of command-selectors gives an informative description of user requirements.

In the implementation phase the use of a module generator reduces the production time because Fortran code-lines describing the communication is automatically generated. Another advantage is that the basic structure of all modules is the same (see fig. 5). Therefore a schedule for the implementation phase (see fig. 6) can be used.

In the maintenance phase the similarity of the modules makes it easier to correct a module, esp. in case the module was implemented by a former colleague.

5. BATCH

If an analysis is to be performed by many datasets it is clumsy to run the system interactively. Therefore FTSS has been designed to store communication sequences in FTSS batch files. These files can be executed with no interference. A sequence may use several modules and an arbitrary number of FTSS batch files can be stored.

6. IMPLEMENTATION OF A RAY-TRACE MODEL

As a dedicated software package an implementation of a ray-trace model is briefly described. (See fig. 7). The model consists of 7 independent modules exchanging data by disc files.

The model is based on analytical formulas by a set of functions fitting a measured sound speed profile and a description of bottom and surface by their reflection coefficients.

The modules perform the following tasks :

- Interactive fitting of Epstein profiles to a measured sound speed profile.
- Calculation of reflection coefficients of bottom and surface for a number of selected frequencies.
- By four modules the ray-pattern can be examined in details along a vertical or horizontal line.
- Calculation of the energy distribution of the sound propagation field along a vertical or horizontal line. In case a 2-D display of the energy field is wanted, the calculated energies are stored in an image-file which can be displayed or further processed by the FTSS image package.

The structure of the ray-trace model is typical of models implemented in FTSS, I.e. to split the model components, and let component data be generated by seperate modules and be stored in individual disc files.

CONCLUSIONS

The principles of a software system with applications in different fields of physics was presented. It is made up by independent modules exchanging data by standarized disc files, which makes the system very easy to extend. All modules have the same basic structure, a standarized communication between user and software and the same schedule is used in production of new modules.

For the user the advantage is a system easy to operate since all modules seems very alike. For the software staff the advantages are a system very excellent to work with in both the production and maintenance phases. After six years experience of working with the system it has proved that structured programming at all levels is very powerfull leading to a fast production of reliable software.

REFERENCES

1. P.B.Ring, J.K.Nielsen, O.S.Madsen :
A software package for timeseries analysis and acquisition.
(In Danish.) DDRE report 1976/29.

2. P.B.Ring, N.O.Jensen :
Digital circuit simulation.
DDRE report 1978/46

3. O.C.Mortensen, P.B.Ring :
A software package for fundamental image processing.
DDRE report 1978/42

4. A.V.Møller, P.B.Ring, O.C.Mortensen :
A database software package for a multichannel sensor system.
DDRE report 1978/47

5. P.B.Ring :
Software packages for description and rendering of geometric objects.
DDRE report 1978/48

6. M.Caspersen, P.B.Ring :
A simulation model for an infrared sensor in a scenario.
DDRE report 1979/27

7. H.Preuss, A.Nielsen, M.Andersen :
Description of geometric objects by their contours.
DDRE report 1979/30

8. T.Strarup, V.Jeppesen :
A ray-trace sound propagation model.
To be published.

9. T.Strarup, A.Pedersen :
A mono pole source array sound propagation model.
To be published.

10. G.Hvedstrup Jensen :
A software package for pattern recognition and image analysis.
To be published.

DISCUSSION

T. Totten Aren't disc files slow?

P.B. Ring It depends very much on the case; in the field of image processing it has been a problem, therefore, we extend the "core" to 256 K.

D. Nairn How many man-years were required in developing the system?

P.B. Ring Up to now we have invested approximately twenty man-years in modelling and programming the system.

R. Seynaeve What will be the future evolution of your system?

P.B. Ring The next step is to switch from the HP-DOS III monitor to the HP-RTE IV monitor and to add an array processor.

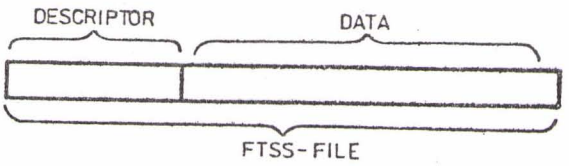

Fig. 1

TIMESERIES DESCRIPTOR

A	PROJECTNAME	EGB
B	NAME OF FILE	EKS1
C	TYPE OF FILE	TIMESERIES
D	FORMAT OF FILE	FL POINT
E	TIME OF CREATION	13/10-1975 : 10.42
F	CREATED BY PROGRAM	TSMAT
G	PROTECTED	YES
H	UNIT OF DATAELEMENT	MV
I	SCALE FACTOR	0
J	SAMPLINGFREQUENCY IN HZ	1
K	SAMPLINGSPERIOD IN SEC	1
L	NO. OF BLOCKS	10
M	BLOCKSIZE	512
N	LENGTH OF TIMESERIES IN SEC	5119
O	ADDITIONAL INFORMATION	3.1244E+02
P		0
Q		0
R		0
S		0
T		0

Fig. 2

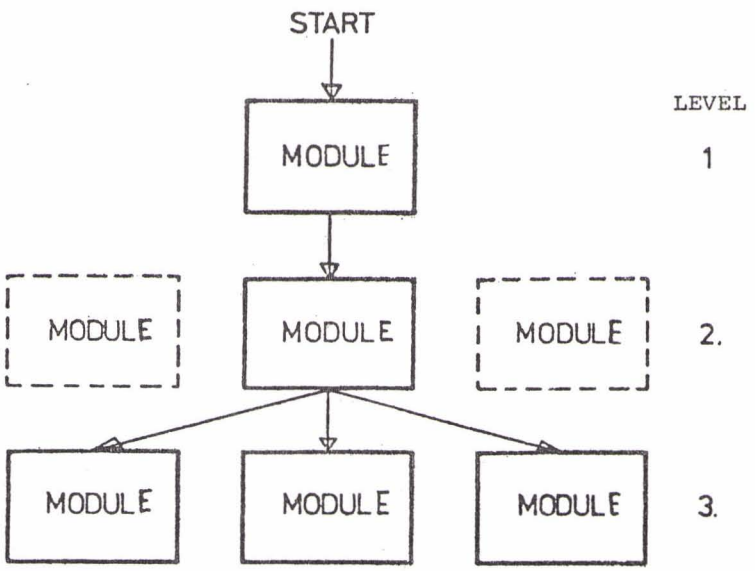

Fig. 3

```
                    *************************
                    * BEAM EMITTING MODULE *
                    *************************

*****************************************************************************
* A BEAM EMITTING FILE....................* ............................ *
* B SOUND-SPEED PROFILE ..................* ............................ *
* C RECIEVING DISTANCE....................* ..........................0 *
* D NUMBER OF BEAMS TO BE EMITTED ........* ........................421 *
* E NUMBER OF BEAMS TO BE DISPLAYED ......* .........................21 *
* F BEAMDIRECTION ........................* .........................90 *
* G BEAMWIDTH ............................* .........................30 *
* H VERTICAL POSITION OF SOURCE ..........* ..........................0 *
* I PLOTMEDIUM............................* .....................SCREEN *
* J * CALCULATE BEAMS ....................* ............................ *
* K * Z-0 PLOT............................* ............................ *
* L * DZ-D0 PLOT..........................* ............................ *
* M * Z-T PLOT ...........................* ............................ *
* N * NEW MODULE..........................* ............................ *
* O * RETURN..............................* ............................ *
*****************************************************************************
```

```
          *****************************************************
          * INTERACTIVE GENERATION OF REFLECTION COEFFICIENT *
          *****************************************************

*****************************************************************************
* A REFLECTION COEFFICIENT FILE ..........* ............................ *
* B SOUND-SPEED PROFILE ..................* ............................ *
* C SELECT FREQUENCIES....................* ............................ *
* D LIST FREQUENCIES .....................* ............................ *
* E SELECT BOTTOM/SURFACE ................* ............................ *
* F SELECT BOTTOM/SURFACE PARAMETERS......* ............................ *
* G LIST BOTTOM/SURFACE PARAMETERS........* ............................ *
* H * CREATE REFL. COEF. FOR RAYTRACE ....* ............................ *
* I * DISPLAY REFL. COEF FOR RAYTRACE ....* ............................ *
* J * 3-DIM. DISPLAY OF REFL COEF ........* ............................ *
* K * NEW MODULE..........................* ............................ *
* L * RETURN..............................* ............................ *
*****************************************************************************
```

Fig. 4

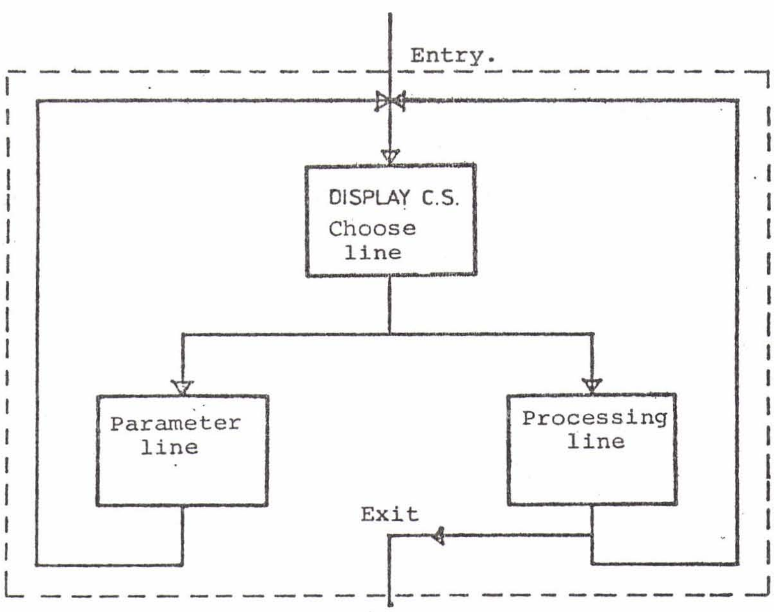

Fig. 5

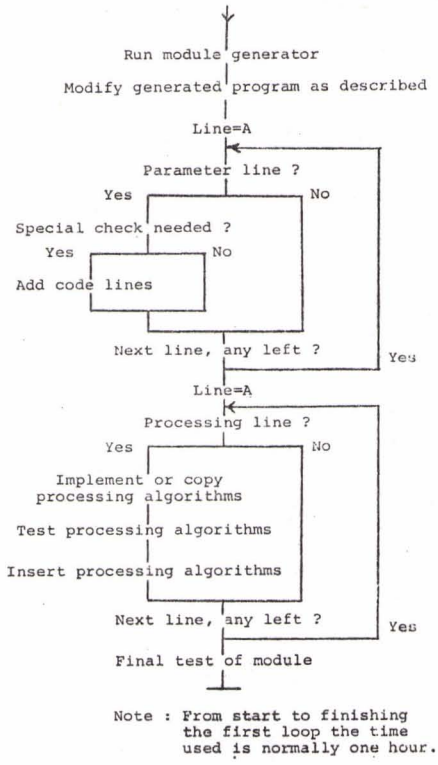
Run module generator
Modify generated program as described
Line=A
Parameter line ?
Yes
No
Special check needed ?
Yes
No
Add code lines
Next line, any left ?
Yes
Line=A
Processing line ?
Yes
No
Implement or copy processing algorithms
Test processing algorithms
Insert processing algorithms
Next line, any left ?
Yes
Final test of module
Note : From start to finishing the first loop the time used is normally one hour.

Fig. 6

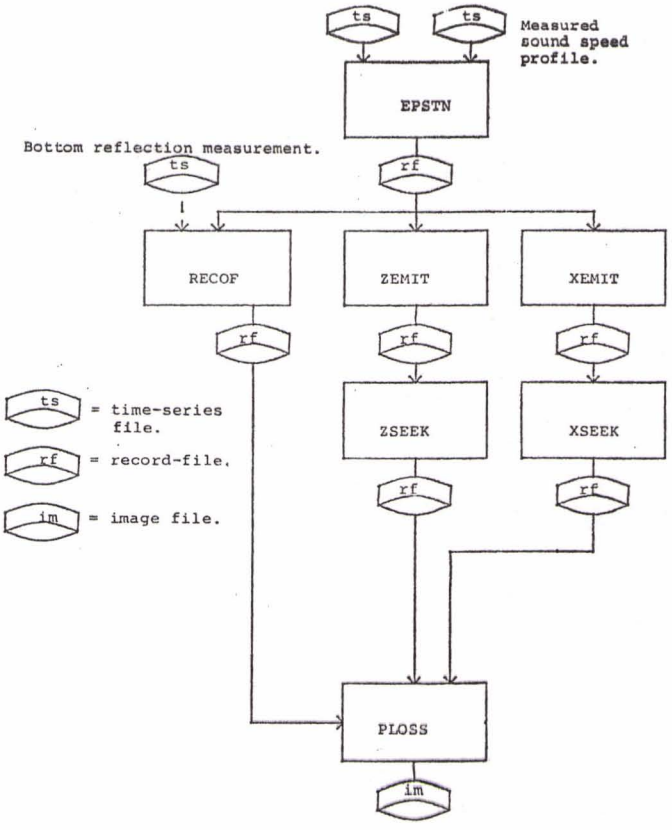
ts
ts
Measured sound speed profile.
EPSTN
Bottom reflection measurement.
ts
rf
RECOF
ZEMIT
XEMIT
rf
rf
rf
ts = time-series file.
rf = record-file.
im = image file.
ZSEEK
XSEEK
rf
rf
PLOSS
im

Fig. 7