 Közlemények 31/1984. 77-94

SYSTEM DESIGN TECHNOLOGY

A. TÓTH

INORGA, Košice, Czechoslovakia

1. INTRODUCTION

Design and implementation of computerized information and control systems is a complicated process.

This process involves a very wide range of managerial and technical activities such as: problem analysis, user requirements specification, functional design, data structure design, computer-equipment selection, program development, user training, system testing, etc.

The way from the idea (i.e. requirements specification) to the final product (i.e. information or control system) often called as project life-cycle unfortunatelly is not straightforward:

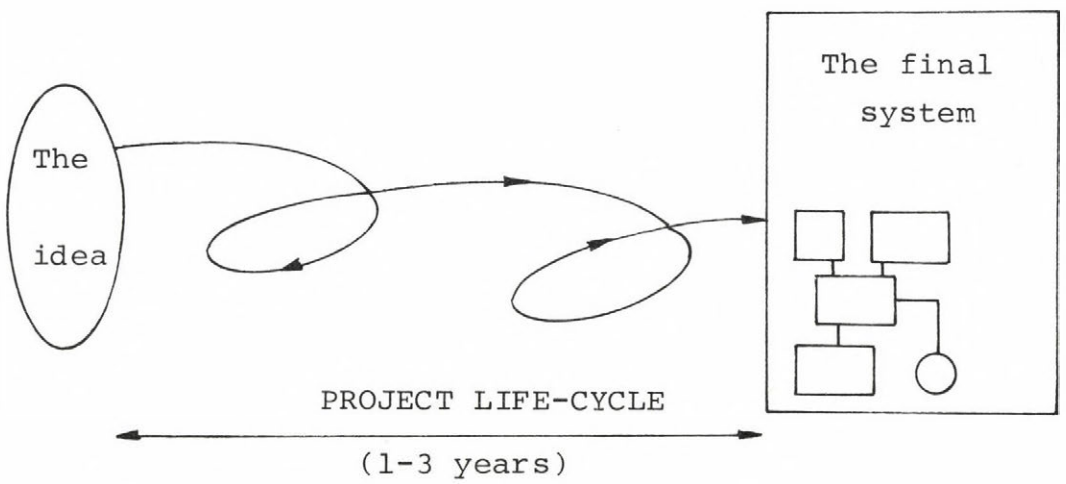


Fig.1: System development process

The succes of our projects depends on many facts. Let us mention only the most crucial ones of them:

*1 Does the developed system satisfy the functional requirements specified at the project initiation?
*2 Has the project met the original budget?
*3 Was it implemented within the given time schedule?

In the case of computerized projects the answers to these well-known questions steem from the following 'details':

What is the manpower and its professional level allocated to the project?
* To what extent is the company management and the potential users involved in the development process?
* The level of the project management itself?
* Which from the available methods and techniques are used for a particular system development activity?
* The level of project documentation?
* The level of user training?
* Do the end-users know how to use and operate the system developed for them?
* etc.

Really there are too many things we should take into account and combine them properly into a goal oriented development process. Unfortunatelly should only...

The usual case is that some of these details are neglected due to shortage of time or lack of project development experience:

we often use unefficient and/or incompatible methods in design and programming; the results of the work are incompletely docomented and certain details are known only by some 'key professionals' who can quit in the middle of the job; the project progress is not systematically evaluted so nobody knows what has been done and has to be done; after 1-2 months the users are already 'out of game' and are waiting for hopeful results; eventually, if these usually delayed results missmatch the user needs the project has to be reworked again.

Besides wasting of money and time we have to realize a dep-

ressive consequence of this ad-hoc system development approach:

There is no garantes that these mistakes will not happen again in the next project. Until there is no structured and recorded knowledge neither about the system developed nor about the development methods used this job remains

'a secret art of some experienced masters'.

In such situation young or less experienced fellows have very little chance to learn alone from the case studies of imlemented projects. They have to work in the shadow of their masters for long time as observers if they wanto to acquire this system development knowledge

This is a serious problem also in the developing countries where there is generally a lack of experienced computer professionals.

SDT (System Desing and Development Technology) offers an opportunity to resolve this problem.

2. THE SDT APPROACH

In first aproximation SDT is a structured set of proved methods combined into an activity-network for system development process.

Immediatelly one can object each project must have a specific development strategy (and activity network) so there is no general approach. This objection is true until we percieve the systems development as a heuristic process.
However, after a deeper analysis, we can 'discover' many common technical and managerial features among seemingly quite different, types of projects. For example:

* the rules of activity planning and project progress monitoring
* how to prepare a functional or data structure specification
* how to document a program algorithm, etc.

All such project independent features are incluede in the sc called SDT-skeleton. This a 250 page guide we can consider as a common model for system development. On the case study of a production control project it illustrates the usage of SDT approach. The main components of this modular document are:

AS - System development activity structure
AL - Activity list
AD - Detailed activity descriptions
AN - Activity network
WM - Optional set of working methods
WMT - methods for technical activities
WMP - methods for project management activities
DS - Project documentation standards
DSP - project progress documentation standards
DSF - final system documentation standards
DSS - sample progress and final documentation
CS - Computer-aided design facility
GL - Guidelines for SDT implementation

This skeleton is a structured knowledge-base for project planning:

* The activity list (AL) specifies WHAT has to be done
* The detailed descriptions (AD) supported by an optional set of working methods (WM) specify HOW to carry out each activity. Documentation standards (DS) specify HOW to document the result (output) of each activity and the final system developed
* The activity network (AN) determines WHEN to carry out an activity

The computer-aided design fasility (CS) is an optional support for SDT-users. This software-packege can interactively maintain all development documentation in a common project data base.

Now let as have a brief overview what is inside the SDT-skeleton:

2.1 AL - Activity list

This is a three-level hierarchical list of activities:

- development phases
- subphases and
- project steps

We have picked out some parts from the sample production control project activity list:

Project phases:

First level:

A - Problem analysis
B - Functional design
C - Data structure design
D - Process structure design
E - System implementation

subphases of B:

2nd level:

BOP - Standards training for phase B
B1 - Functional structure specification
B2 - Input/output specification
B3 - Functional feasibility test
. . . . .
B9P - Project progress monitoring

3rd level:

step for B1:

B11 - Determine time horizons
B12 - Specify functions for horizon
B13 - Describe the functions
(AD)
. . . . .

steps for B9P:

B9P1 - Check the completness of B
B9P2 - Evaluate progress in B
B9P3 - Prepare schedule for phase C
. . . . . .

Comments: - codes containing P denote project management activites.
- an average project-life-cycle consists of about 120 project steps.
- the description of B12 we can find in section AD (Fig.2 on the next page)

2.2 AD - Activity descriptions

For each activity there is a standardized activity description sheet:

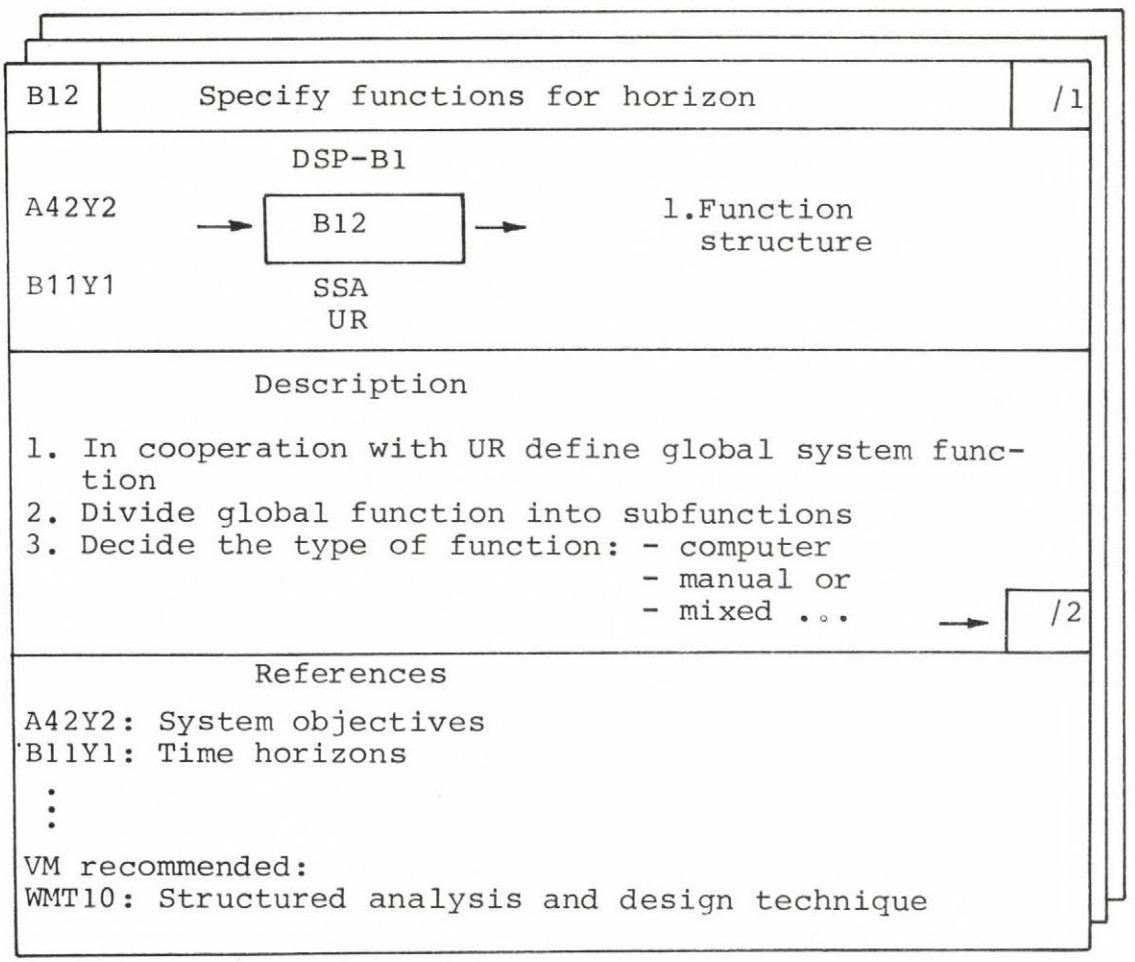


Fig.2: Activity sheets

Explanation:

* This step should carry out a senior systems analyst (SSA) in cooperation with a user representative (UR)
  They will need input documents A32Y1, B11Y1 and their result will be identified by B12Y2 (Function structure)
  This output should be docemnted according to documentation standard DSP-B1 (described in DSP component)
  The description part of the sheet can continue on the next page.

* Progress document coding: A42Y2 is the 2nd output(Y) of step 2 subphase 4 phase A.

2.3 AN - Activity network

Depending on the scale of the project this is a 2-3 level time-based precedence graph representing the activities and their scheduled duration:

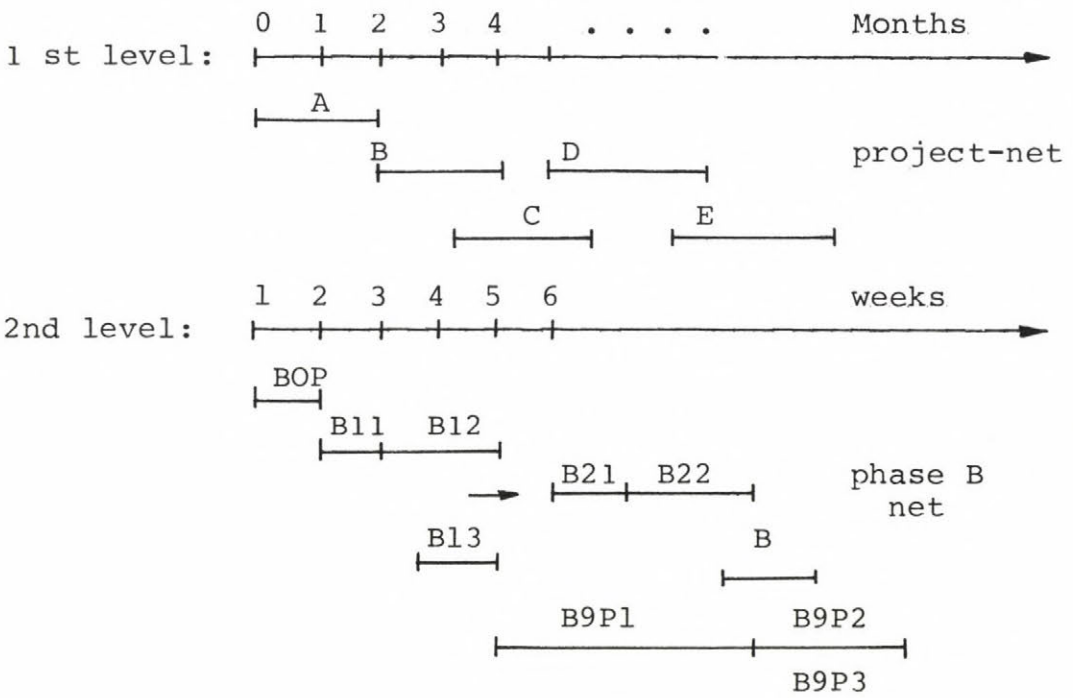


Fig.3: AN-samples

2.4 WM - Optional set of working methods

Is a collection of proved technical and project management methods and techniques. For example:

for technical activities:

WMT01: Information flow diagrams the ISAC method
WMTO2: System an program flowcharts
. . . .
WMT10: Structured analysis and design technique
WMT11: The HIPO method
..... etc.,

for managarial activities:

WMP01: Job assignment and evaluation
WMP02: Bar-charts
WMP03: Time-based networks
WMP04: Project man-power planning

These methods we can find also in a wide range of publications. But here in WM one can find them together (the most relevant ones) and already properly 'chained' to project activities: on the activity-sheets there are recommendations for WM selection.

2.5 DS - Project documentation standards

As it has been already mentioned in the Problem statement the quality of the documentation is a key condition for project success.

In SDT the documentation has two main functions:

* a medium for exact inter-team communications during system development
* a comprehensive information source for understanding and operating the system developed

For that a documentation standard is a common prescription for;

- how to record the result of work and
- how to percieve its content by other persons.

The project progress standards (DSP) are structured according to activities (AL,AD) the final system documentation standards (DSF) according to system structure.

Now let us see what is the standard for output B12Y1 included in DSP-B1 (see activity sheet, page 6):

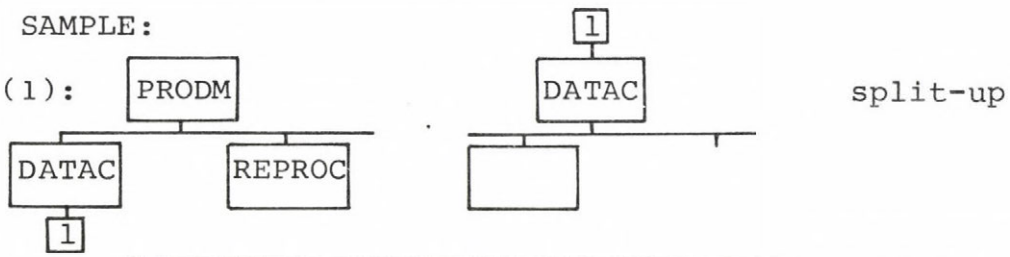
In Fig. 4A there is a prescription for B12Y1 output specification. This is a conventional standard for manual-mode design. If we decide for computer-aided design mode (see pages 10-13) then this prescription should be converted to rules of a specification language - shown on Fig.4B. In this case, however there is no need to draw hierarchical diagrams: the computer will draw and insert it into your progress and/or final documentation.

.....

DOCUMENT: B12Y1 - Function structure
COMPONENTS:
(1) Hierarchical diagram
(2) Function structure description

SAMPLE:

(1):



(2):

IDENT	NAME	TYPE	...
PRODM	Production monitoring	MIXED	
DATAC	Shop-floor data collection	MIXED	
REPROC	Report processing	COMP	
...	....		

EXPLANATION:
(1): ..... Identifier not longer than 8 letters
.....
(2): .....

Fig.4A: Project progress documentation standard sample - (conventional design)

```
FMODUL : CONCEPT 'Functional module'
           ATTRIBUTES:
fname  :     TEXT40 'Module name'
       :     MTYPE  'Module type'
           LEGAL OWNERS: HORIZON, FMODUL
fm     :   FORMAT:
             <ident>:MODUL<fname>TYPE<ftype>
           SAMPLE:
             -- ...
             -- (see screein image on Fig.6)
```

Fig.4B: Project progress documentation standard sample - (computer-aided design mode)

2.6 CS - Computer-aided design facility

The common feature of any conventional system development it produces a great amount of paperwork which has to be shared simultaneously among several professional groups:

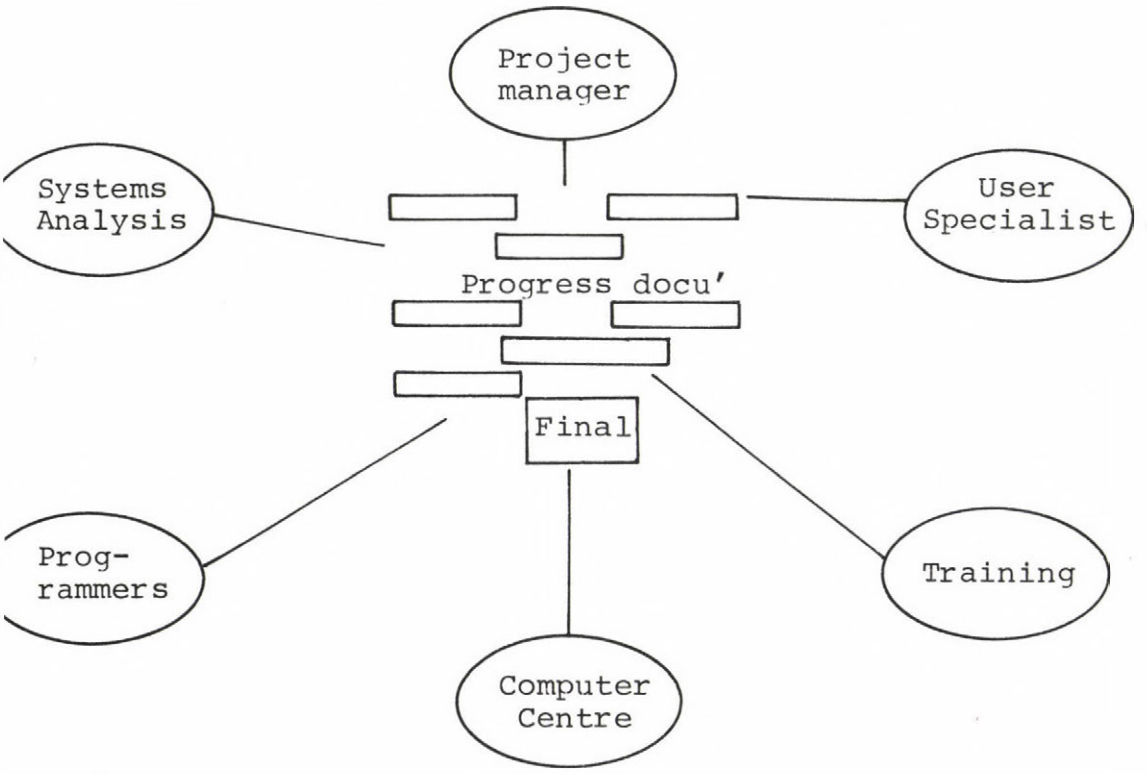


Fig.5: Conventional system design

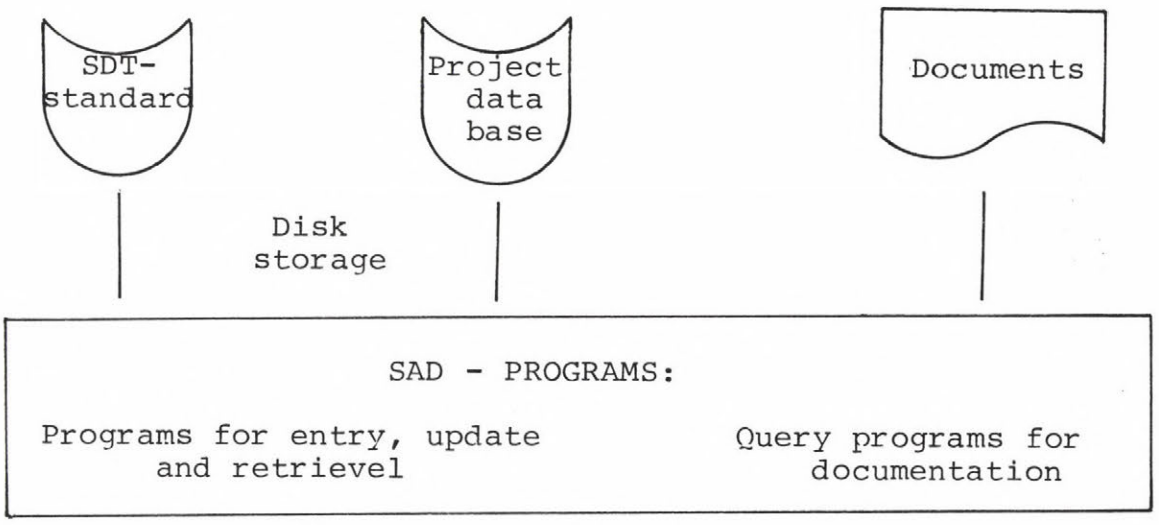
Disadvantages of this situation are well-known:

cost and time consuming clerical work; to modify reports or drafts is very difficult; the recorded results are often inconsistent; we have to search for some details in many documents; the final documentation has to be compiled from progress drafts manually and in many versions for: computer operators, users, training, etc.

The solution: in SDT you can utilize the computer to execute this clerical work!

The program support which provides these services is called:

SAD (Softvare for Advanced Design)



SAD-GUIDE
SDT-INEX

Fig.6:
Computer-aided design facility

Using SAD the outputs of project activities can be simultaneously entered and/or up-dated through on-line display terminals into a common project data base. In Fig.6 a part of B12Y1 activity output entry is illustrated.

The SDT standards here are stored also on disk storage so the SAD automatically can verify the correctness of the specifications entered before storing them into the project data base. This ensures the contents and structure of the project data base is consistent with the standards. These standards (denoted as BS2 on the screen, Fig.6) can be selected and then entered interactively as a set of expression rules at the project initiation phase. Really be means of these rules we formulate our specification language tailored to the 'dialect' of particular project phases.

For early design phases (A,B) we can select natural language-like constructs and for the final ones computer-oriented language contructs. Here is an example for a real-time program specification (phase D):

```
ocdc    :PROG'On-line converter data collection'
        :. PROGRAM INPUTS:
cbuf    :. . BUFFER'Host control buffer'
chspec  :. . . GROUP'Channel specification'
        :. PROGRAM OUTPUTS:
mbuf    :. . BUFFER'Measured values'
chnum   :. . . DATA'Channel number' PICT octal
valuem  :. . . DATA'Value measured' PICT floar
        :. COMMON DATA:
        :. . EVENT flag35 = 'Cbuf queued for satellite'
        :PROGRAM ALGORITHM:
ocdc1   :. RECEIVE cbuf
        :. . WAIT FOR flag35
        :. . CLEAR flag35
ocdc2   :. DOUNTIL'All channels processed'
        :. . SELECT chspec
        :. . . CASE 10
        :. . . . DO'Measure temperature'
        :. . . CASE 15
        :. . . . DO'Measure 02-flow rate'
        :. SEND mbur
        :END ocdc
```

Fig.7: Program specification sample

The design of any specification language by SAD-language definition facility is a relatively simple job:

- we define sentence types, (e.g.Program heading, etc.)
- some keywords, (PROG,BUFFER, etc.)
- attributes types (text,data,evenet,number, etc.)
- and finally specifiy feasible relations of these sentences (e.g. the RECEIVE sentence can be only part of PROGRAM ALGORITHM or a DO-type sentence)

So, you can define your own project-oriented design language!

For any case the DSP kit of SDT skeleton, contains a sample language for each project phase of a production control project. Addition to this standard kit there are also available further language sets from a wide spectrum of previous SDT applications. From these sources new SDT users can easily compile their own language or directly take an existing one.

Now let us summarize the benefits of this computer-aided design approach for:

* project management staff:
  - using S (Show), T (Trace), G(Graph) commands we can obtain an up-to-date project progress documentation on arbitrary object and level of details.
  - using the SAD query programs we can obtain immediatelly final or semifinal system documentation from the project data base; we can specify the format and contents of documentation required for different user levels.
* system developers:
  - the result of my work I can directly and exactly specify in A (Auto-guide) mode where the computer step by step asks for the details of my activity output
  - if I want to add or extent a specification already stored, using R (Refer), F (Format) or I (Insert) commands I can easily do it.

- if I want to change, rearrange or delete some parts of my outputs I use the U (Update), C (Change), M (Move) or D (Delete) commands.
- if I need some details on results of my colleanges I can retrieve them by S,T,G commands.
- if I forget some dialog command I type only H (Help)

* user representatives:

- we have got always up-to-data and comprehensive documentation for systems operation and for 'end-users' training
- if some system change happens we only run the document processing programs and we obtain a new version within a couple of minutes.

The above statements are typical responses of SDT/SAD users. Really comparing with the conventional style of develpment this

new facility can reduce the project costs and time up to 40% and significantly increases the quality of documentation.

For this reason this facility addition to SDT is highly recommended for project staff having access to a computer with terminal network. SAD is now running on IBM mainframes on PDP11/family and on some personal computers. To provide easy portability it was developed in FORTRAN IV programming language (the personal version in BASIC) with sensible core-memory requirements (56 KB).

2.7 GL - Guidelines for SDT implementation

SDT implementation involves a set of preparatory activities providing application of the SDT skeleton for a given project. Generally we have to complie a new skeleton let say SDT-x which should incorporate all the pecularities of our new project. Such a project dependent part is for example the Activity network (AN).

As it is shown in Fig.8 for each SDT-x skeleton component we have to decide how to create it:

*1 take it from an existing skeleton without modification?
*2 take if from an existing skeleton with modification?
*3 create a completely new one?

An existing skeleton can be either the master SDT or some dedicated SDT-1, SDT-2,... skeletons from previous SDT-driven company projects.

Existing skeletons:

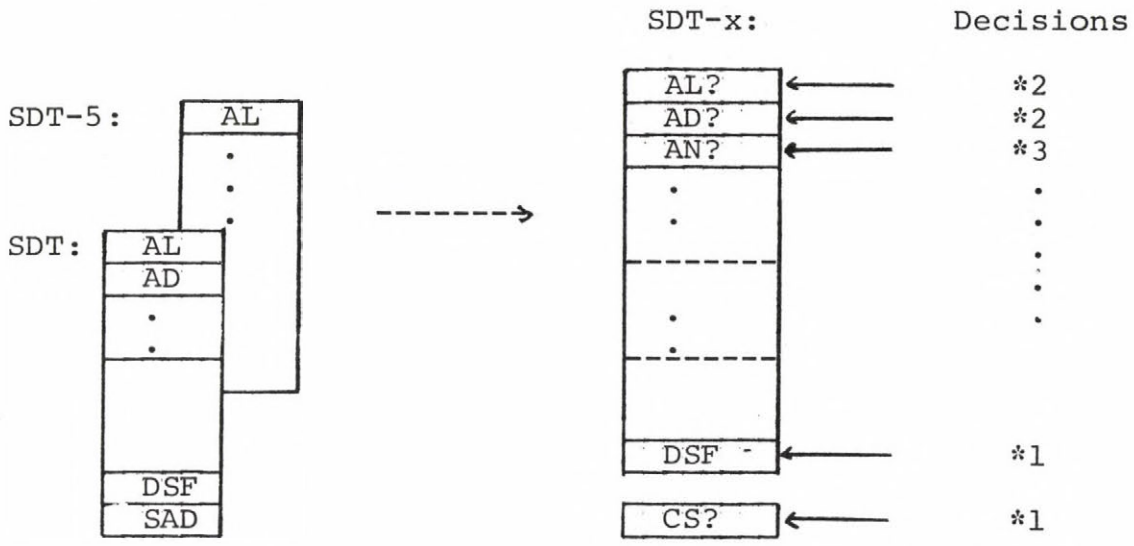


Fig.8: SDT implementation (example)

For example if we are going to prepare SDT-x for a Blast Furnace process control project we potentionally should utilize the SDT master skeleton and an (SDT-5) skeleton of Rolling Mill production control project implemented before.

In ideal case, if we keep our skeletons on computer storage a new SDT-x guide we can compile easily using a text editor.

Despite the SDT implementation is principally and technically a comprehensive process assistance of an SDT-specialist at least for the first implementation is recommemded.

Depending on project size and skeletons available takes about 0,5-3 months. The implementation team configuration recommended:

Project manager (for coordination)
Senior systems analyst
Chief programmer
SDT-specialist

3. CONCLUSIONS

The goal of this report was to provide an overview on SDT approach which gives an oporutnity to promote the current systems development practice from art to technology.

The challenge is not new: the traditional technical disciplines (as engineering or building design) have made this move decades before.

For computerized systems development this is not a moment too soon: while the capabilities of new computer hardware increase by orders of magnitude, the productivity of people creeps forward by a few percentage points a year.

The challenge is great; the reward greater.

4. FOOTNOTES

[1] SDT has been developed by Industrial Management and Automation Institute (INORGA), Czechoslovakia - Kosice during 1979-83.

[2] SDT has been implemented and verified on about 16 domestic projects and on two UNDP/UNIDO projects:

DP/CZE/77/005 (Czechoslovakia)
and DP/EGY/13/002 (EGYPT)

5. REFERENCES RECOMMENDED

[1] Teichroev D., Hershey E.A.III, PSL/PSA a computer aided Technique for Structured Documentation and Analysis of Information Processing Systems, IEEE Tranactions on SE-J, (1977).

[2] E.Knuth,F.Halász.,P.Radó: SDLA - System Descriptor and Logical Analyzer, Proc. Information Systems Design Methodolosies, North Holland 1982, pp. 143-171.

[3] Demetrovics,J., Knuth E., Radó P.: Specification meta System, IEEE on Computers, May 1982. pp.29-35.

[4] Tóth A.: SAD - Software for Computer-aided System Description on Minicomputers, IFIP TC2 Working Conference on System Description Methodolgies, Kecskemét, 1983.

[5] Békéssi A., Demetrovich J.: Contribution to the Theory of Data Base Relations, Discrete Mathematics 27 (1979) 1-10.

[6] Sobik F., Sommerfeld E.: A Graph Theoretical Approach to the Characterization of Classes of Structured Objects, Computers and Artifical Intelligence, 3 (1984), No.3., 235-247

[7] Georgescu I: A Catesorial Approach to Knowledge-based Systems, Computers and Artificial Intelligence, 3 (1984), No.2, 105-113.

Ö S S Z E F O G L A L Á S

RENDSZER TERVEZÉS TECHNOLÓGIA

Tóth Attila

A számitógépes információs és irányitási rendszerek tervezése bonyolult folyamat, amely a müszaki és irányitási tevékenységek széles skáláját ölei fel: probléma analizis, felhasználói követelményspecifikáció, funkcionális tervezés, adatszerkezet tervezés, számitógép kiválasztás, programfejlesztés, felhasználók oktatása, renszertesztelés, stb.

Az SDT bevált módszerek strukturált készlete, amelyek egymásba kapcsolódva egy olyan folytonos tevékenységhálót képeznek amely magába foglalja a rendszerfejlesztési projekt életciklusát.

ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ СИСТЕМ

Аттила Тот

Проектирование информационных и управляющих систем для ЭВМ является сложным процессом, охватывающим широкий диапазон деятельности: анализ проблем, спецификация требований пользователей, функциональное проектирование, проектирование структуры данных, выбор ЭВМ, развитие программ, обучение пользователей, проверку систем и т.п.

Система CDT является структурным набором проверенных методов, которые образуют непрерывную сеть деятельностей, охватывающую цикл жизни проекта выработки систем.