AN ANALYSIS OF INFORMATION SYSTEMS DESIGN METHODOLOGIES

INFORMATICA 3/1988

Krista Rizman
Ivan Rozman
Anton Zorman
Tehniška fakulteta Maribor

UDK 681.3.:519.863

ABSTRACT
To improve development and use of information systems methodologies, these have to be discussed and studied from many aspects. We have analysed JSD, ISAC, SA-SD and the Warnier/Orr methodology. The contents of this paper is not a description of studied methodologies. It is the description of our findings and the results of evaluating the practical value of the methodologies in relative terms by comparing them. We characterize the methodologies according to their life cycle coverage, their representation shemes, learnability, their behavior in the real time environment and automated tools by which they are supported. Our main point here lies in demonstrating that each of the four methodologies is relative powerful in some circumstances and for some systems.

1 INTRODUCTION

The use of computers has spread to all areas of labour and life and the way of use has changed. In the first period of use, computers were used only for calculation. Almost all data processings were numerical. The main point of use has moved from calculation to storing and searching data - information. There has been a great progress of new approaches, methodologies and tools for developing of information systems respectivly application by the last decade. The information system is a system with the following tasks: creating, collecting, processing and distributing informations. Information systems could be developed only if they in some way can improve some activities of information process.

The progress of computer technology, especialy the fall of prices, caused the mass use of computers. The development of applications has become a bottleneck. This is a reason why the languages of the fourth generation (application = functional languages: LISP, PROLOG, query languages...) and a great number of computer aided tools for system analysis and design have been produced.

There are a great number of graphic tools and methodologies that can be used for analysing and design of information systems. The problem is to choose one from this set of methodologies, that will be the most efficient, easy for use and will give the best results. There are many questions: 'Which methodology to choose?', 'Which is the best?', ' Which is the most general?', 'What are advantages of each of them?'

Which methodology to use? This is a difficult question because it depends on your needs, on the type, the extension and the complexity of information systems you want to develope, on your experience, style of thinking and probably on your knowledge of principles of different methodologies.

We restrict our study on methodologies that are shown in table 1, mainly because we think they are the most efficient and widespread used for developing information systems. This paper presents the results of study the different methodologies with the purpose to answer some questions above.

Table 1: In the following we give the methodologies covered in this report along with developers

Methodology mnemonic	Full name of methodology	Developers
SDM	System Development Methodology	G.F.Hice, W.S.Turner
SA-SD	Structured Analysis and Structured Design	De Marco Yourdon
ISAC	Information Systems Work and Analysis of Changes	Mats Lundeberg
JSD	Jackson System Development	Michael Jackson
	Warnier/Orr-method	Warnier, Orr

It must be pointed out that our analysis draws on the referenced literature. This is important because methodologies are not finished products. They do not have precise characteristics. They are improved through time. Methodologies in a practical use can be adapted to chainging environments and circumstances. On the other hand, the authors often emphasize ascpects considered as most original, while aspects regarded as more usual are left out.

Several points of view can be used to analyse a methodology and all of them must be considered in a complete analysis.

2 LIFE CYCLE COVERAGE

By analysing the mentioned methodologies, we find out that they have much in common. They have a great diversity in form, in original principles and in name of each phase, but they agree with basic elements that need to be defined. A lot of them cover almost all phases of a life cycle(table 2).

The life cycle is an important concept in discussing methodologies. Our view is that an efficient methodology must support a process of activity that covers the entire life cycle. It does little good to have a methodology for design if there is not a procedure for specifing requirements and functions which are used for the design and the system that must be built. There are numerous life cycle models in use and many of them separate analysis and functional specification activities./PORC83/ We merge both of them into one phase (analysis) because the discussed methodologies do not distinguish between these two steps. For both phases almost all of them (except the ISAC methodology) support the same graphical diagrams and other tools and in both the users are most included.

In this paper each methodology is presented through the description of the following life cycle steps:

- analysis,
- design,
- implementation,
- validation,
- evolution.

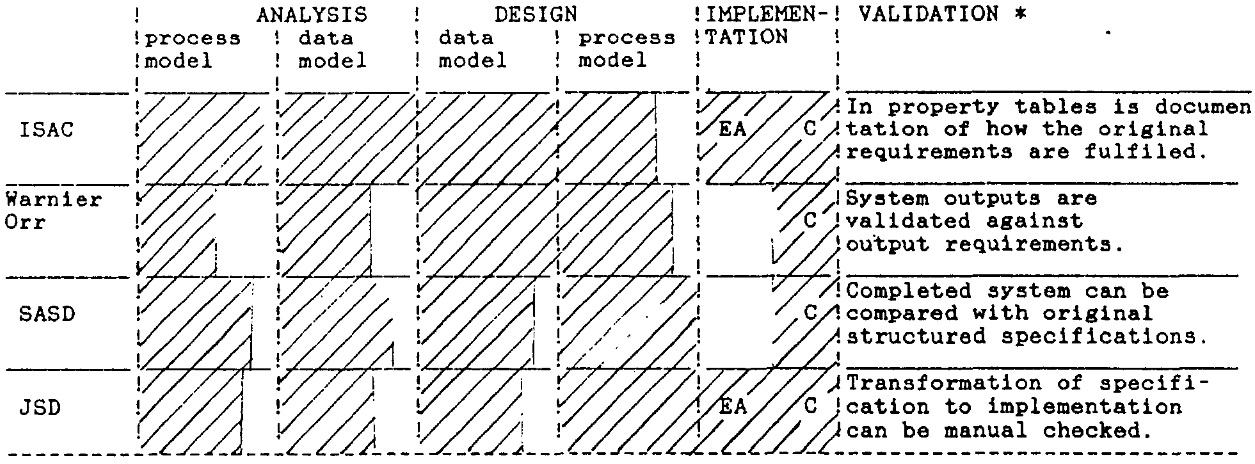
2.1 Analysis

Analysis is the first step of any information systems development activity. The result of this step is besides the requirements definition also a functional specification - description of system functions and an answer to the question : 'What should the system do?'. Functional specifications are used during the design phase as a checkpoint against which to validate the design.
The successful analysis includes communications with the users. The analysis must be able to bound a problem and to identify those areas where the information system is useful and practical.

A particularly effective method of analysis is modeling. Models represent the problem and the real world in a formal form. Models used for analysis are graphical diagrams, graphs and tables.

Because of the complexity of problems and systems, methodologies must support a problem decomposition which can be procedural or data flow or data abstraction.

All discussed methodologies are performed through an analysis of the data, either data structures or data flows! The data orientation is sensible because data are more stable than processes. SA-SD and ISAC analyse data flows, but the Warnier methodology analyses the data



	ANALYSIS		DESIGN		IMPLEMEN-TATION	VALIDATION *
	process model	data model	data model	process model		
ISAC	▨	▨	▨	▨	▨ EA C	In property tables is documentation of how the original requirements are fulfiled.
Warnier Orr	▨	▨	▨	▨	C ▨	System outputs are validated against output requirements.
SASD	▨	▨	▨	▨	C ▨	Completed system can be compared with original structured specifications.
JSD	▨	▨	▨	▨	▨ EA C	Transformation of specification to implementation can be manual checked.

▨ - represents how detail is a particular phase covered

EA - methodology contains a special phase with the purpose to choose specific equipment and then to adapat the equipment independent solution to this choice

c - coding

* - how the completed system is validated against the original requirements

Table 2: Life cycle coverage

structures of the outputs. JSD analyses the entity/action structure of the real world. (Figure 3)

```
                methodology
                    /\
                   /  \
                  /    \
                 /      \
                /        \
are performed through   are performed through
an analysis of data     an analysis of data
    structures                flows

  JSD, Warnier               SA-SD, ISAC
```

Figure 3: A simple division of the information systems design methodologies

Our view is that the data flow methodologies are more understandable than the methodologies which base on the data structure. A data flow diagram (DFD) is a better tool for understanding and more perfect representation of any information system than data structure diagrams. It represents both flows and actions. (Figure 5 shows an example of DFD.) It is a network diagram that can be easy used for bounding the system. First of all, we draw the diagram with only one action and with input and output data flows. They are then decomposed. We think that so the users and the analysts can easily, systematically and by degrees, with the help of DFD, make the requirement definitions and the functional specifications.

The SA-SD methodology is very useful for working in a team because of the logical functional decomposition and well-known input and output data of any decomposed action.

More complex and detailed analysing process can be done by ISAC. Besides the information (message) flows we can also describe the real flows (persons and objects). The picture of current and future information system is complete.
ISAC is very strong in the early phases of the system life cycle: the change analysis, the activity analysis and the information analysis. The ISAC approach is extensive and comprehensive!
In ISAC interest groups are studied thoroughly and described both with the problems they have. This is a part of the change analysis. The ISAC approach is widespread used in the Scandinavian countries.

The weak point of ISAC is that the graphical notation for the information analysis is redundant because all information contained in information graphs (I-graphs) is derived from activity graphs (A- graphs) from the change or the activity analysis. (Figure 6 shows an example of A-graph!)
The weak point is also that it is necessary to illustrate two identical information sets in the same I-graph because of the hierarchical way of description. Two sets are needed when one information set is output set of a graph and at the same time is a precedent to other set in the same graph.
I-graphs show information sets and precedence relations among information sets, but C-graphs (component relation graphs) describe the contents and the structure of the information set. The methodology does not provide that the same information set will have only one C-graph. It depends on when the particular C-graph is created.

We think that many of definitions and work of the information analyse could be overcome by use more powerful information model such as extended entity-attribute diagram to replace both information graphs and component graphs.

Weakness of JSD is the first step of methodology (entity-action step), by which we analyse data and actions of the real world. It seems to us that in this step the methodology does not provide such a graphical tool which can help the users and the analysts to edit, colect and represent specifications (especialy all entities).
Graphical notations are useful in showing the interrelationships between the system components, which enable easy communications between the users and the analysts and so help both of them to build the complete information system.
JSD does not provide such a graphical tool. Jackson suggests a simple process to make a wide list of entities and actions: nouns which appear in the users description of the real world are identified as potential entities, but verbs as possible actions. The users many times forget to mention some parts of the reality because they do not have resource for sistematic description of often very complex sistems. Unfortunately, the complete list of entities and actions is request and condition for success of entire development. JSD does not support graphical presentations of links between entities of the entire system, from which can the users and the analysts quickly find out the missing entities.

The JSD methodology is little oposite to the other information systems development methodologies. They tend to more exact analysis with purpose to build a better system with less price to meet the needs of its users and to reduce costs of correcting.
The second tendency is reducing of returning to previous steps. Of course, there is an iteration, but we all want to reduce it as much as possible. Modern methodologies and computer aided tools include mechanism to reduce it to minimum.

In the **Warnier methodology** the first step is to determine which are required outputs. The answer is quite obvious when the system is not too big. Else we have to subdivide the real system into several smaller. Many times the subdivision will be done according to the organisation of the firm. Analyst may help formulating questions and so help to create a list of required outputs. The methodology does not provide any graphical tools to help in this first step.

All the methodologies except JSD apply the hierarchical decomposition. Of course, levels of detail are related to complexity and vagueness. The vagueness concerns the early phases of life cycle, when the functional and the data system may be fuzzy and there is no clear idea how the system will work. In this context a crude information analysis is quite good alternative. The possibility of the crude information analysis is embedded in the ISAC methodology. We start to build new system with the crude information analysis in the change analysis and then we end with the detailed analysis in the information analysis phase. The crude information analysis is also supported by SA-SD, which enables simple execution of the functional decomposition.
Our view is that the most detailed analysis is provided by ISAC, then follow SA-SD, JSD and the Warnier methodology.

2.2 Design

is the process of determining the architecture of the system components, the algorithms to be used and the logical data structures. This phase is an answer to the question: 'How will the system perform the functions defined in the previous phase?' . An output of design activities can be used by the programmer to implement the system. We must emphasize that coupling and coheison are the simple judge of whether a design strategy produces good designs or bad designs.

A developer must be able not to continue only forward to the next phase of the life cycle, but also back to a previous phase. The need for this is the fact that work must be changed and any necessary corrections can be made. It is important to note that information lost at a particular phase is generally lost forever with a bad result on the system. For example, if a requirement are not documented, it will not appear in the functional specification and it will cause the failure of the system.

All the methodologies except JSD are very strong based on the levels of abstraction and on the hierarchical decomposition, where there is always the problem of wheter the model at the lower level satisfies the specification fixed at the upper levels. This problem can partly be dispatch by detailed transformation rules from an upper level to a lower one and by automated tools.
SA-SD supports two transformation rules: a transform analysis and an analysis of activities for producing a structure diagram from data flow diagrams. We must tell that structured diagrams can not be made only by the transaction and the transform analysis but it requires some judgement and common sense on the part of the designer.

This strategy is considered in the Warnier methodology well but in ISAC only particular. ISAC makes levels of abstraction quite visible, but there is not a visible connections between the analysis phase and the design phase. Perhaps it is the reason in use of other method (Jackson Structured Programming) for the design.

2.3 Implementation

is the production of executable code. Coding transforms algorithms into functions or procedures and logical data structures into phisical data structures. It must be pointed out that good coding cannot make up for poor analysis or design. Good coding cannot make bad information systems good! This phase is an answer to the question: 'How can we run this system on machine available to us?' .
The ISAC and the JSD methodology enable design which is not confused with implementation. The delimitation between design and implementation is in the ISAC and the JSD methodology very rigorous. Not before the latest phase we include the use of existing hardware and programming languages for realization developed system.
This is an advantage of both methods, because the design system is more transferable and more portable. We can use it with little changes on different hardware because only the last phase must be changed.

The choice of the hardware and the software needed and technical asspects of the implementation the Warnier mathodology and SA-SD do not solve.

2.4 Validation

is the process of determining that a system correctly performs those functions described in the functional specification. It is often a step performed as a part of each phase where we must verify that the phase correctly carries out the intent of the previous step. The validation of code may be done either through testing or formal proof of correctnes.
The methodology must support determination of system correctness through the life cycle. Methodologies usualy enable correspondence between the results of one stage of development and the previous stage.
Table 2 shows how the whole system can be validated against the original requirements.

2.5 Evolution

Systems go through many versions during their lifetimes. The development methodology can help in evolution phase by providing system documentation and, of course, a well structured software system that is easy modified by people making the system changes. The great emphasis to a well structure of a program is given in the SA-SD methodology. The factors contributing to interactions between systems components (modules) and the coheison of individual systems components are very well described. /Your79/ We tent to the greater coheison of individual modules in the system and the lower coupling between modules.

3 INTERMEDIATE WORK PRODUCTS

By methodology we mean a number of coherent work steps including rules for types of documentation that are produced during these work steps. The documentation must be a natural part of work and not something that you do afterward! Table 4 shows the steps of all the four discussed methodologies and the products that are produced at each step! Figures 5,6,7 and 8 show the working procedure of all the four methodologies. Each working procedure is presented by the diagram, which is particularly signicifant for each of the four methodologies.

We have already emhasized that graphical tools in ISAC seem to us redundant because the contents in I- graphs is the same as in A - graphs. But the purpose of using both graphs is different. A- graphs give an overview of the connections between the information system and its environment. I-graphs show the information contents in detail. ISAC enables adding details in a systematic way, but by help of the different graphs.

We think that the data flow diagrams (SA-SD methodology) have an advantage, because it can be used for connections of the information system with the environment by sorces/sinks and for adding details by the functional decomposition and multilevel diagrams. For all this we have to use more graphical notations of the ISAC methodology.

There is an assumption in ISAC that careful and detailed decomposition of the user activities will largely procedure the information requirements. From ISAC point of view work methods are more important than description tehniques. We do not quite agree with this because we emphasize that description tehniques must be used as the basis for understanding the problem and for communication between the users and team.

TABLE 4: Table shows steps of system development of all the four discussed methodologies and the products that are produced at each step:

ISAC:

working procedure: different analysis and design areas each of them is devided into more than 3 steps and substeps!)	workproducts (documentation)
1. CHANGE ANALYSIS:analysis of problems and needs and the current state. We define and produce alternative changes and choose the best!	A-diagrams are used for hierarchical decomposition current activities. We can use also: problem groups tables, text pages, property tables, table of objectives!
2. ACTIVITY ANALYSIS: we continue with the decomposition of activities. We more detail define information flows and information subsystems!	A-graphs (Figure 6), property tables
3. INFORMATION ANALYSIS: we analyse relationship between information sets and the structure of information sets.	I-graphs:hierachical graphs of information flows, textual description C-graphs
4. DATA SYSTEM DESIGN	D-graphs D and P strctures (JSP)
5. EQUIPMENT ADAPTATION	E- graphs

JSD:

working procedure	work products (documentation)
1. ENTITY/ACTION STEP: We define entities and actions by help of user specifications (actions are verbs, entities are nouns).	entity action list
2. ENTITY STRUCTURE STEP	structure hierarchical diagram (Figure 8)
3. INITIAL MODEL STEP: An entity is defined as a proces which is with data flow or state vector conected with entity of the real world or with other process in a model. Processes are detailed described with pseudocode.	System Specification Diagram Jackson pseudokod
4. FUNCTION STEP	System Specification Diagram
5. SYSTEM TIMING STEP	note of temporal requirements
6. IMPLEMENTATION STEP	System Implementation Diagram

SA-SD:

working procedure	workproducts (documentation)
1. ANALYSIS of system actions, information flows, data bases	data flow diagrams (DFD)-(Figure 5) data dictionary, decision tables, psedocode
2. DESIGN : with help of transform analysis and transaction analysis we produce from data flow diagram hierarchical data structure!	structure diagram psedocode

WARNIER:

working procedure	workproducts (documentation)
subdivision the big system into several smaller, each of them have its own data procesing system.	note of subsystems
the list of the required output and the description all of them	Warnier diagram (Figure 7)
the organisation of all the data needed to obtain the output, the design of a logical base.	Warnier diagram
the definition of transactions needed to update the data.	
the definition of logical programs	Warnier diagram

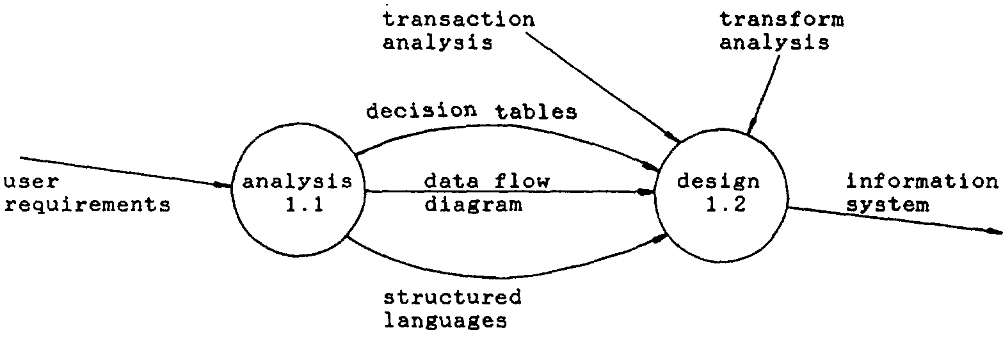


Figure 5: DFD of the SA-SD working procedure

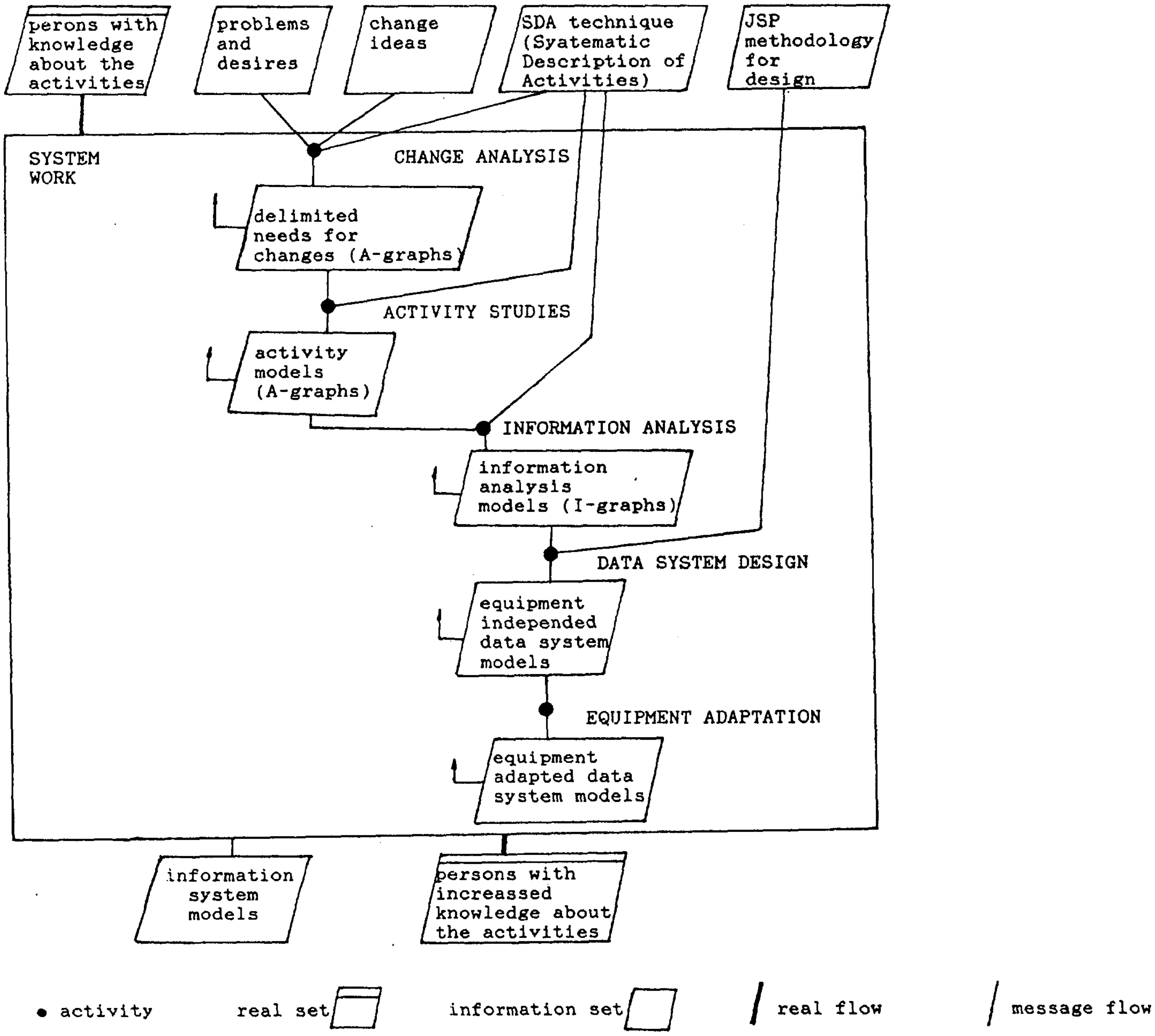

perons with knowledge about the activities
problems and desires
change ideas
SDA technique (Syatematic Description of Activities)
JSP methodology for design
SYSTEM WORK
CHANGE ANALYSIS
delimited needs for changes (A-graphs)
ACTIVITY STUDIES
activity models (A-graphs)
INFORMATION ANALYSIS
information analysis models (I-graphs)
DATA SYSTEM DESIGN
equipment independed data system models
EQUIPMENT ADAPTATION
equipment adapted data system models
information system models
persons with increassed knowledge about the activities
● activity
real set
information set
real flow
message flow

Figure 6: A- graph of the ISAC working procedure

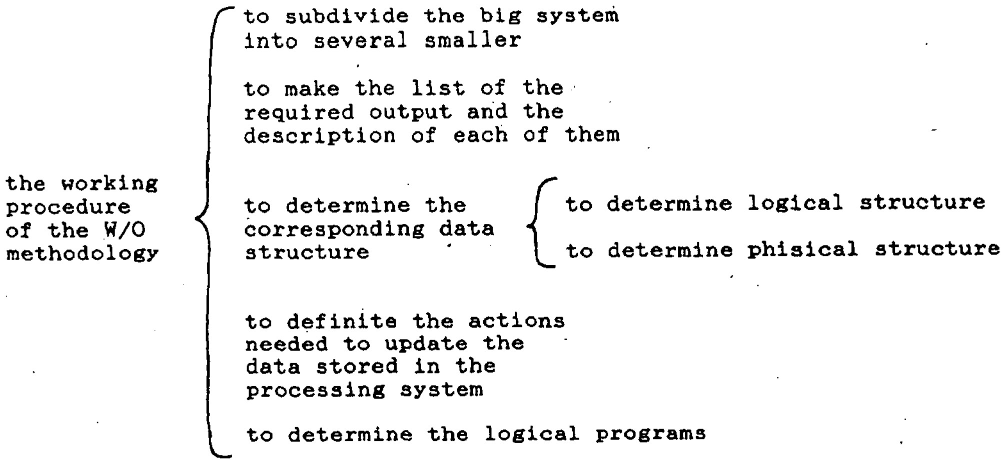

the working procedure of the W/O methodology
to subdivide the big system into several smaller
to make the list of the required output and the description of each of them
to determine the corresponding data structure
to determine logical structure
to determine phisical structure
to definite the actions needed to update the data stored in the processing system
to determine the logical programs

Figure 7: Warnier graph of the Warnier working procedure

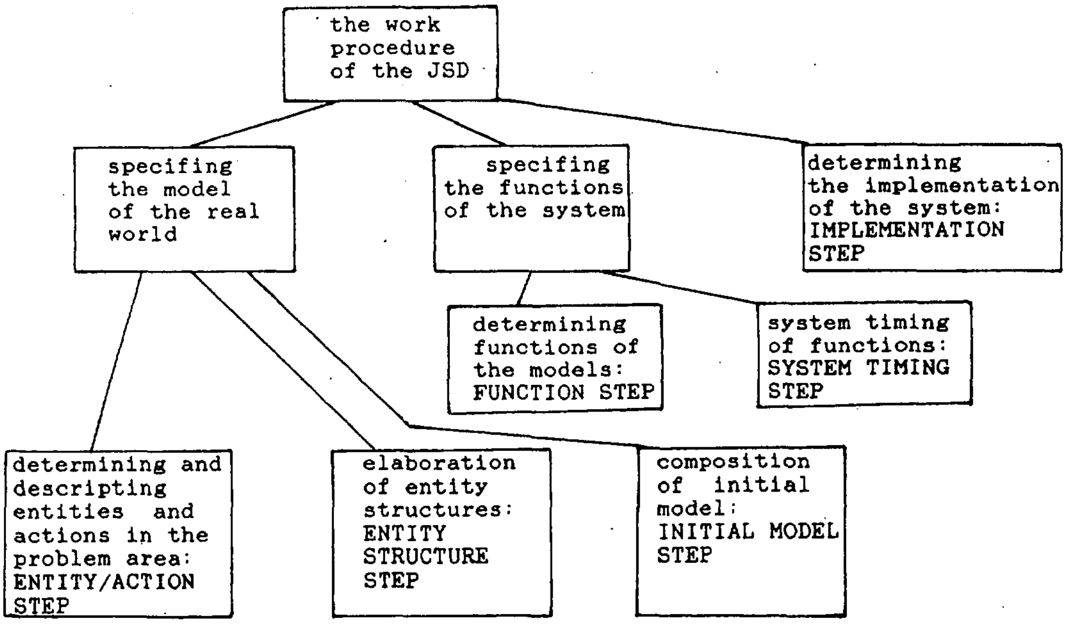

the work procedure of the JSD
specifing the model of the real world
specifing the functions of the system
determining the implementation of the system: IMPLEMENTATION STEP
determining functions of the models: FUNCTION STEP
system timing of functions: SYSTEM TIMING STEP
determining and descripting entities and actions in the problem area: ENTITY/ACTION STEP
elaboration of entity structures: ENTITY STRUCTURE STEP
composition of initial model: INITIAL MODEL STEP

Figure 8: Structure diagram of the JSD working procedure

ISAC is relatively complex due to several levels of abstraction and several graphical notations.

It seems clear that understandability is reduced by the relative complexity of descriptions.
In all development phases of the Warnier methodology we can use only one graphical tool-Warnier diagram (Figure 7). We can describe the data and structure process by only three basic components of stuctured programming (sequence, iteration and selection).

4 BEHAVIOR IN THE REAL TIME ENVIRONMENT

Behavior of the methodology in the real-time environment is also very important because the information system represents a set of coherent different or equal actions. We think that JSD is the most suitable of all the four discussed methodologies for applications from the real world with the important temporal extension. In the JSD System Timing Step adequate measures are taken to ensure a correct scheduling of system processes. For this purpose, synhronisation processes are defined.

It is important to emhasize that JSD is intended for development not only for data processing, but for other applications also. Temporal dimension of information is not treated explicitly in ISAC, nor in the Warnier methodology.

5 LEARNABILITY

The methodology must be easy to learn because even within single organisation, there will be quite a great number of people who have to use the methodology as a resource and it must help all of them and not make a developer process more difficult.
It is clear that the methodologies must be communicable to other persons not only to developers. Learnability depends on the complexity of the methodology , which is probably related with covering the software development life cycle and perhaps on the depth of the understanding of information systems provided by it. We establish that among the four discussing methodologies only ISAC is more complex, the others are relatively simple. We think that ISAC is less easy to learn.

6 AUTOMATED TOOLS

It is clear that automated tools which offer series of understandable resources, brought near people, make possible the supervision of a project and immediate repairing existing failures. Automated tools give up to date version to all members of the team.

A great number of automated tools and environments have been explicitly developed to support nearly all studied methodologies. We do not know if such tools are commercialy available for ISAC methodology in a broad sense although in the ISAC group a prototype system called IA/2 was developed in the early seventies. Automated tools faciliate the work to designers and improve the productivity of both the individual developer and development team.

Tools for computer-aided software engineering provide these benefits:
- improved productivity and faster systems development (They automate design and documentation, eleminate manual drawing and redrawing and allow quick changes.)
- higher quality software (They produce universal documentacion, promote standardisation.)
- reduced maintenance (They promote easy changes and allow on-line access to design.)

7 CONCLUSION

It is clear that any information systems development methodology is better than no methodology!
We think that there is no one information systems design methodology which is best for developing all information systems.
We have represented the main findings about the methodologies. It is clear that our analysis is in many respects preliminary, because of extreme complexity of the subject.
In this section we describe importance of the methodologies from the practical point of view.

We demonstrate that each of the four methodologies is relative powerful in some situations.
We evolute the applacability of the methodologies in very crude terms by use these dimensions:
- fuzziness of the information requirements
- complexity of the data system
- complexity of the actions on the data of the system

Our view is that the relative value of ISAC are the earlier phases. It is effective and suitable for fuzzy information requirements, but it is restricted to not too much complex system because of the restrictions caused by division information and data system into relatively small and simple sub-systems without any flexible mechanism for integration.
We conclude that the Warnier methodology is less powerful in the fuzziness aspect. Then follows JSD methodology. SASD lies somewhere between JSD and ISAC which include a powerful special mechanism to manage fuzzines. (See Figure 9.a)

JSD is not quite useful for very complex data systems, but for systems with many actions, which bring entity from one stage to other, because JSD methodology provide good description technique for showing temporal sequence of actions with three basic components of structured programming (iteration, sequence, selection), but it does not provide good description of entities of entire system and relationship between them. In the implementation step we consider temporal performance of each groups of actions. It is more powerful in the description of actions then entities. (See Figure 9.b and 9.c)

We think that the Warnier methodology is suitable for developing complex data and action systems because of use only one simple and efficient Warnier/Orr diagram.

Our view is that SA-SD is very useful also for more complex data and actions systems. SA-SD methodology enables simply description of data components in the data dictionary. A structure of actions are described by a structure language or decision tables.

```
complexity of the       complexity of the  actions
 data system            on the data of the system

  high ^                               high^
       !                                   !
       + SA-SD                             !
       !                                   !
       + WARNIER                           !
       + ISAC                              + JSD
       + JSD                               + SA-SD
       !       vagueness                   + WARNIER
       !     .                             + ISAC
       !       high ^                      !
  low  !            !                 low  !
                    + ISAC
                    !
Figure 9.b          + SA-SD           Figure 9.c
                    !
                    + JSD
                    + WARNIER
                    !
                    !
           low      !
```

Figure 9.a

8 REFERENCES

/CAME86/ - J. R. Cameron: An overwiev of JSD, IEEE Software Engineering 1986/2.

/DEMA78/ - Tom De Marco, Structured analysis and system specification, Prentice-Hall, New Yersey, 1978.

/HIGG79/ - David A. Higgns, Program Design and Construction, Prentice-Hall, 1979.

/INFO83/ - Information Systems design methodologies: A Feature Analysis, Elsevier Science publishing company, Amsterdam, 1983.

/JACK83/ - Jackson M. A.:System Development, Prentice-Hall, 1983.

/LUND81/ - Mats Lundeberg, Goran Goldkuhl, Anders Nilsson: Information Systems Development: A System Approach, Prentice-Hall, 1981.

/PORC83/ - M. Porcella, P.Freeman, Ada Methodology Questionarie Summary, ACM Software Engineering Notes, Vol 8, No1, Januar 1983.

/THEI83/ - The Information Structures Subgroup of the Dutch Database Club:' Software Engeneering: Methods and Tehniques, Wiley - Interscience Publications International 1983.

/YOUR79/ - Yourdon E., Constantine L., Structured design, Fundamentals of a Discipline of Computer Program and Design, Prentice-Hall, 1979.