SEEKING THE ACTUAL REASONS FOR THE "NEW PARADIGM" IN THE AREA OF IS ANALYSIS

Václav Řepa

Prague University of Economics, W.Churchill sq. 4, 130 00 Praha 3, Czech Republic
E-mail: REPA@VSE.CZ

1. INTRODUCTION

Lately the very attractive and frequented terms are the "object orientation" and the "change of paradigm" in the area of information systems analysis. After the object revolution in the area of programming the areas of analysis and design are now attached by these thoughts. Unfortunately, not all of the principles, techniques and methods for the object-oriented programming are fully implementable also in the area of analysis and design. So the old truth that analysis is something quite different from the programming is gradually fading away. The aim of this paper is to indicate the *actual reasons* for the paradigm change using analysis of the *general problems* of the old but still living "structured methods" of IS analysis and design.
By the term "structured approach to IS development" I mean various data-oriented as well as function-oriented methods of IS development. One of the most complete concepts of Structured methods is, for example, the Yourdon structured method [YOUR-1] which includes both data and functional models on the conceptual level of the IS development. Yourdon method recognises the need of integration of these two (data and function-oriented) approaches, until recently regarded as substantially different.

2. GENERAL CHARACTERISTICS OF THE "STRUCTURED APPROACH" IN IS DEVELOPMENT

As the name of this approach indicates it is based on the separation of different areas of interest:

- Data and processes are taken into consideration separately
- Hierarchical abstractions are used for separating the abstract (high level) concepts and their relationships from the subordinated (more detailed) ones
- Modelling as the key principle enables the developer abstract view on the general characteristics of the information system deprived of their particular shape, which is complicated by various non-essential aspects
- Building information system gradually on three different levels - conceptual, technological and implementational (the "Three Architectures" principle) reflects the effort to separate three different, relatively independent problem areas: essential concept based on the model of reality, technological design and implementational shape of the system

The main reason for such separation is simplification of the problems, which have to be solved. Even if the system is relatively small there are too many aspects in too complex relationships to describe them simply (i.e. clearly). And there are too many relatively simple problems in too complex relationships to solve them at the same time. Separation seems to be the vital condition for the mental control of a problem.
In the following text the general characteristics of the main forms of separation mentioned above is described. Some of them will prove as the source of problems leading to the need to change the paradigm of IS analysis.

2.1 Data and processes separation

In the structured approach to IS development there is the significant difference between two basic components of IS - data and processes. Off course, this difference is partly caused by the evolution of IS development methods. The early 70's brought the first revolution in the IS development - data analysis and data modelling. The first significant paradigm was formulated: "IS is based on the model of the real world". Unfortunately, this paradigm was formulated just on the field of the "data component" of the IS and have led to well-known statement of James Martin that data are more stable than functions, therefore the model of the real world is just the database. Although this thought have started a long discussion on the differences between data and processes, it brought to life the main principle of IS analysis - the principle of modelling. This principle is discussed in the next paragraph. Significant influence on the evolution of IS development methods had also the evolution of technology - database systems whose support of non-creative activities in IS development process even more emphasised the "separation paradigm". Good design of the database was considered as the most important and creative activity. Processes in the information system have been regarded as not so important and easily implementable (with use of database systems support - report generators, user interface generators, query languages etc.) according to rapidly changing user requirements. Later evolution of data analysis and modelling has taken into account also procedural dimension of the real world model (in the form of procedural database languages (4GL), mechanisms of integrity constraints, triggers etc.). But the "separation paradigm" is in place - all these real world functionality phenomenons are regarded as characteristics of the data model. On the other hand the evolution of function-focused analysis methods also accepts the separation of data and procedural dimensions of IS. For example Yourdon method talks about three separate parts of conceptual model of the IS - data model, behaviour (function) model and control model. The last two models form the procedural model of the IS. Its separation into the two models reflects the inability to arrive at a settlement with the complexity of relationships between top-down hierarchical structure of system functions and not top-down structure of data entities. From this contradiction between two antagonistic concepts of hierarchical structuralisation follows all well-known problems with control flows in functional structure. This contradiction will be discussed in more detail in the following paragraphs. So structured approach to IS analysis looks into information system from two (or three if we accept Yourdon's division of procedural model to the behaviour and control models) different points of view. Each of them follows the specific logic and requires specific language (i.e. tools as DFD or ERD) for description of the IS structure.

2.2 Modelling

The principle of modelling has been first formulated from the data point of view: contents and structure of database objects reflect contents and structure of the real world objects. Correctness of the data model is measured via its similarity to the real world. For such measuring the term "similarity" must be defined exactly. Therefore the special tool - Entity Relationship Diagram (ERD) has been developed. ERD describes the essential characteristics of the real world: objects and their mutual relationships. It is constructed to be able to describe exactly the objects and their relationships in the same way as we see them in the real world. At the same time this model describes the essential requirements for the database - it must contain the information about the same objects and their relationships. The form in which particular database describes these facts always depends on technological and implementational characteristics of the environment in which the database is realised. But the essential shape of the model still remains the same. Because of the need to describe the same database in its various shapes (essential, technological, implementational) the principle of different architectures have been formulated. This principle, generalised to the scope of the whole system (not only its database) is discussed bellow. Modelling principle proves to be general too - also some parts of system processes have to be regarded as the model of the real world. But the main problem of so-called structured approach in IS development is that it is not able to recognise which system processes form the model of the real world and which do not. Such recognition requires separation of the modelling operations from the other ones and organising them into the special algorithms according to real world objects and their relationships. And this point of view is not reachable under the "separation paradigm" without accepting the natural unity of the modelling system processes and the data in database. Incidentally, acceptation of the natural unity of the modelling processes and the data entities would enable to solve Yourdon's problems with control processes - the essential controlling algorithms follow from the entity life histories.
As shown above the Modelling Principle seems to be general and independent on existing paradigms. The new paradigm can only specify its place in IS development but cannot eliminate or limit it.

2.3 Hierarchical abstraction

Hierarchical abstractions are the means for decomposition of the elements of designed information system to the level of detail. Higher level concepts consist of the lower level ones. On each level of detail the elements of developed IS and their relationships are described. The elements on each higher (i.e. non-elementary) level of detail are abstract concepts. Only the last (i.e. most detailed, elementary) level contains definite elements. This structure of dependencies between the concepts of the higher and lower levels is called "tree structure". Each element has only one parent element on the higher level (with exception of the highest element so-called root of the tree) and can have several child elements on the lower level (with exception of the lowest elements, called leaves of the tree). Abstractions used in structured methods are of two types:

- *collectivisation* is the abstraction, which collects the abstract concept from its child concepts. The abstract concept consists of the number of its elements, which are of different internal structures and attributes. The only criterion for the connection of child elements is that they belong to the same parent concept - subordinated elements are *parts of* the superior concept. Example of this type of abstraction is the concept of "forest" which consists of its parts - particular "trees".

- *generalisation* is the abstraction, which generalises the abstract concept from its lower-level concepts. The abstract concept consists of the number of its elements, which have the same basis of internal structures and some common attributes. The criterion for the connection of child elements is that they are the specific cases of the same parent concept - subordinated elements are *particular types of* the superior concept. Example of this type of abstraction is the concept of "tree" which can be specialised into the specific types as "spruce", "pine", "maple" etc. Each particular tree has root, branches, bark and other common attributes. All particular trees have also the same rough structure of their life cycles - each tree grew from the seed, is growing up and each tree must also die one day under the wood-cutter's axe or another way.

The names of the two basic types of abstraction are taken from Michael Jackson's classification of the real world abstract concepts - JSD entities [JACK-1].
The collectivisation type of abstraction is typically used in structured methods of IS development for decomposing the functions into sub-functions (using the famous Top-Down procedure, for example) while the generalisation type of abstraction is typically used for decomposing the entities of the conceptual data model into sub-entities. Incompatibility of these two basic approaches to the concept decomposition inside the structured methods has often played the role of source of vital problems of the "structured paradigm".

2.4 The "Three Architectures" principle

The principle of "Three Architectures" was mentioned in the paragraph in which the "modelling principle" was discussed. These two principles have very much to do one with the other. Separation of the implementation and technology-dependent aspects of developed information system from the conceptual ones is the vital condition for putting the Modelling Principle into practice. Without such separation the developer would not be able to see (and to discuss it with the user) the model of real world in the functional and database structure of developed IS. Three levels of the shape of IS seem to be essential:

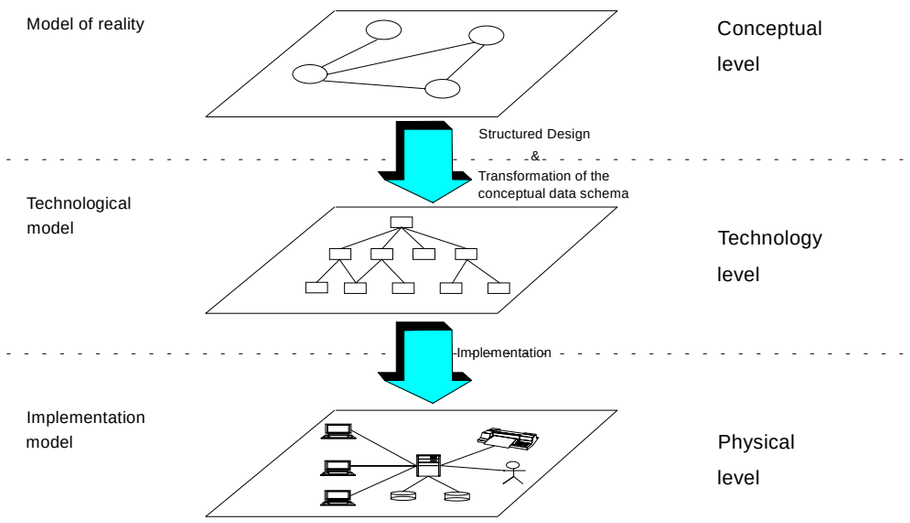
- *conceptual model* represents the clear model of real world, which is not blacked out by the non-essential aspects given by assumed technology and implementational environment of the system
- *technological model* is based on the conceptual model enriched by the aspects given by assumed technology. Including the technological aspects often significantly changes the original - conceptual - shape of the system For example 3GL technology using sequential files for realisation of the database leads to the data structures considerably distant of the conceptual entities and their relationships. On the other

hand relational database technology preserves the maximum of the original shape of the data model. So the degree of shape changes always depends on the used technology.

- *implementation model* represents the final shape of IS. It depends on the used technology taken into consideration in the technological model and respects also implementational details given by the used particular environment. Thus the implementational model is even more distant from the particular shape of the real world than the technological model.

The essential relationships between three architectures illustrates the following figure:

The Three Architectures Principle



Such model of the three different views on the same thing (information system) has some general characteristics:

- each view has specific logic and requires specific methods of examining and specific language for description, which match this logic
- for keeping the consistency between particular views it is necessary to have some means (i.e. methods and techniques) for the transition of the contents of one view into the next view

So each of these three levels of IS development represents specific goal, specific type of developer's activity, specific techniques and tools to use. Also the transition of the design from one to the next level requires specific activities, techniques and tools. In the conformity with the structured paradigm the methods, techniques, activities and tools used on three levels of IS development differ from the functional and the data points of view. These differences are described by the following two tables:

Functional point of view

LEVEL	ACTIVITY	TOOLS	TECHNIQUES	TYPE OF ABSTRACTION
CONCEPTUAL	Analysis	Data Flow Diagram	event partitioning	collectivisation (top-down)
		Structure Diagram		
		pseudocode etc.		
		State Transition Diagram		none
LOGICAL	Design	Structure Chart	modular programming	
			composite design	
			information hiding etc.	
PHYSICAL	Implementation	Programming languages	structured programming etc.	
		generators etc.		

Data point of view

LEVEL	ACTIVITY	TOOLS	TECHNIQUES	TYPE OF ABSTRACTION
CONCEPTUAL	Data Analysis	Entity Relationship Diagram (according to Chen)	normalisation	generalisation (hierarchical structure of entity types/subtypes)

			integration (i.e. canonical procedure)	
LOGICAL	Logical Database Design	Entity Relationship Diagram (according to Martin)	transformation of data model into the logical data structures	
PHYSICAL	Physical Database Design		Database description languages Programming languages etc.	

-
-
-
-
-

2.5 Integrity rules as the tool for eliminating negative consequences of the separations

Integrity rules are the rules use of that is necessary for keeping the consistency of different views on the same thing. How stronger the separation of the design into the different views is, the stronger and bulky the consistency rules must be. Therefore the role of this rules in the structured methods of IS development is very significant. Edward Yourdon [YOUR-1] named integrity rules as the "rules for balancing the different diagrams". This name truthfully expresses the source of main problems in the structured approach. The most important areas of consistency are balancing the data model (ERD) against the function model (DFD) and balancing the function model (DFD) against the control model (STD). Integrity rules used in the structured methods of IS development define potential problems but do not offer the way to prevent them.

3. THE MOST IMPORTANT PROBLEMS AND CONTRADICTIONS INSIDE THE "STRUCTURED APPROACH" TO IS DEVELOPMENT

This section summarises the problems and contradictions in Structured methods of IS development suggested in particular paragraphs of preceding section. There are three main problems:

- integration of the data and the functional models
- integration of the functional and the control models
- transition from the conceptual to the technological model.

All of these problems follow from the separation and all lead to the same solution - to revise and change the Separation paradigm.

Problem of integration of the data and the functional models manifests itself in various forms. The most visible form is the inability to decide what should be the starting point of IS analysis - the data or the function analysis. When the development process starts with the analysis of functions then the idea about stored data structures follows strictly from the need of the functions. Such data structures are real contrary to the normalised data model - one data element typically occurs in several data structures. Later correction of the data structures according to the rules of normalisation leads to the situation when each function needs to contact number of data stores to gather required data. This fact dramatically increases complexity of the function model. Therefore Yourdon method so strongly emphasises the need of clear description of relationships between data stores and data entities in the data dictionary according to the balancing rules. But, unfortunately, it does not tell how to secure the clarity of this description in the process of the parallel functional and data models development. On the other hand the development process starting with the analysis of data leads to the idea that functions are just supporting algorithms, providing the inputs and the outputs to/from the database. So there is no way to ensure the fact that the function structure naturally reflects the real world processes. The algorithms, which reflect the relationships of the real world events, are scattered into particular operations of the input routines and the database integrity constraints. But, unfortunately, this approach does not tell how to find these constraints and operations in the real world.

Problem of integration of the functional and the control models has the same roots. Using the tools offered by Yourdon method, the developer is able to clearly describe hierarchical function structure and the relationships of the functions on particular levels of hierarchy (i.e. particular data flow diagrams). On the other hand he (or she) is able to clearly describe particular control algorithms using state transition diagrams. But how to describe which way the functions and the control algorithms match together? Trying to describe it the developer very soon feels the need of accepting and producing data flows by control processes but, unfortunately, this is not legal in Yourdon method. Control process can accept and produce only control flows in Yourdon method. There is a good reason for such limitation: when the method would accept data flows as input and output to/from control process then the question is: What is the natural difference between function and control process? The difference between function and control processes is very significant in structured methods. This is because the function is taken here as top-down decomposable element while the control process always must be described as elementary process. So the nature of control processes is very similar to the nature of data entities (they also are not top-down decomposable). Viewing this fact from the object (i.e. not structured) point of view we can see the control processes in the system as the expression of the objects behaviour - their life history. The structured methods are not able to reach this point of view because of their need to take the data and the process models of IS into consideration separately.

The third main problem area of the structured methods - transition from the conceptual to the technological model - we can regard as the consequence of the previous two problems. On the technology level of the IS development the integration of the data and the functional models as well as the integration of the functional and the control models must be realised unconditionally. With poor (or, more exactly, none) description of these

facts on the conceptual level it is really a hard task to ensure this integration on the technology level. And the structured methods do not offer any way to ensure it.

4. CONCLUSIONS - WHAT THE SCOPE OF IS ANALYSIS ACTUALLY REQUIRES FROM THE "NEW PARADIGM"

The duty of the progress always should be removing the negative features. So the main contribution of any paradigm change should be a new approach which overcomes the limits and problems given by the old paradigm. Consequently, to ensure the progress during the paradigm change the exact analysis of these problems and limits is necessary.
This section summarises the actual requirements for the new approach to the IS development which follow from the analysis of the problems and contradictions of the Structured methods stated in preceding section. The new approach should:

- *bring to life the method of conceptual analysis, which fully accepts the natural unity of data and processes.*

To achieve this goal it is necessary to develop a tool - diagram of the conceptual model of the IS. The basic element of such diagram is an object. By the term "object" I mean the unity of data (attributes of the real-world object) and process (life history of this object). Each object in the conceptual model represents an object of the real world. Diagram of the conceptual model must also be able to describe the conceptual relationships between the objects. Each relationship between two particular objects represents the causal dependence of their life cycles. It is realised via sending the information about essential events from one object to the other. Data of this relationship then represent the attributes of the event. Michael Jackson whose method JSD [JACK-1] can be regarded as object - oriented distinguishes between two types of object dependencies:
- passive dependence realised as sending the information about the event from one object to the other and
- active dependence realised as watching the attributes of one object by the other.

The difference between stored and sent data expressed in Jackson's classification seems to be technology dependent. But much more likely it reflects natural differences between two essential types of the object relationships.

- *give the possibility of hierarchical decomposition (of the collectivisation type) of the system functions inside the new method of conceptual analysis.*

Although generalisation is natural type of the abstraction of objects (it is expressed by the principle of inheritance) there is the actual need of hierarchical decomposition of the system functions as well. The reason for such request is the same as the reason for the collectivisation in the Structured methods - it seems to be the vital condition for having the problem under control. By the way, the complexity of the model is one of the main problems in the current object-oriented methods of IS analysis (OOA). The difference between two types of system processes must be recognised in order to apply the collectivisation in object-oriented methods. For those processes, which are modelling essential behaviour of the objects (their life history), the generalisation type of the abstraction is natural. But the essential model usually contains also other processes - those, which provide the outputs of data from the system to the real world. And these

processes do not model the real-world objects behaviour, nevertheless they depend on it. So they also seem to be the part of the essential model. The natural type of abstraction to be used for those processes is the collectivisation. Of course, it is the question whether such output processes should be regarded only as a part of the technological model.

- *ensure easy and not contradictional transition from the conceptual to the technology level of IS design.*

The transition from the conceptual to the technology level of IS design is a big problem in the structured approach to IS development. It requires complete revision of conceptual analysis results and complete change of the approach to the system design. Significant contribution of the new paradigm should be its ability to solve these problems. Current object-oriented methods of IS analysis do not deal with these problems. But in my opinion the problems still exist or, better, the new ones replace them. Evolution of the methods for IS development brought to life several significant conceptions and principles known under the names as "modular programming", "programming in the large", "composite design", "information hiding" etc. Not all of them are fully accepted by the object-oriented methods. But many of the principles found in those conceptions are valid in general. Unfortunately, the extent of this paper is too small to give here detail analysis of these matters. This remains also one of the areas of research in the future.

REFERENCES:

[JACK-1] Jackson, M.A.: System Development, Prentice-Hall Inc., Englewood Cliffs, NJ, 1982.
[YOUR-1] Yourdon, E.: Modern Structured Analysis, Prentice-Hall Inc., Englewood Cliffs, NJ, 1989.