Object Life Cycle Modelling in the Client-Server Applications Development Using Structured Methodology

Vaclav Repa

Prague University of Economics, W.Churchill sq. 4, 130 67 Praha 3, Czech Republic
phone: 00422-24095454, fax: 00422-24223605, E-mail: REPA@VSE.CZ

Introduction

The purpose of this paper is to describe certain aspects of the methodology for information systems development which combine both the traditional (structured) and the object-oriented approach. These features of the described methodology are specific to the target environment and development technology of the information system: client-server technology, GUI environment and the CASE tool for structured system development.
The object-oriented approach is taken into consideration here not just in the sense of OO programming (as is usual in the GUI environment) but also in the sense of OO analysis. The trend to object-oriented thinking in the field of analysis follows from client-server technology.
So this paper is aimed first of all at those parts of the methodology which are connected to that problem.

The Methodology

PDIT is a methodology for information systems development under certain given conditions. It was developed by the Czech softwarehouse ITC PragoData in collaboration with the Department of Information Technologies at Prague University of Economics.
The conditions mentioned above are:

- the target environment is Progress v.7

  Progress v.7 is the relational database system allowing the development of applications with client-server technology. The most significant feature of Progress v.7 is a strong fourth-generation language oriented on GUI development
- the development environment consists of Progress v.7 GUI 4GL environment and CASE tool case/4/0

Because of the relatively good standardization in the field of client-server relational database systems and GUI development environment the PDIT is also usable under weaker conditions:

- any similar database system with the same required features can be used as target environment. The required features are:
  - a client - server technology
  - the ability to use GUI according GUI standards

- any independent GUI tool which follows GUI standards and has the same features as Progress v.7 GUI 4GL can be used as GUI development environment

The main Features of PDIT Methodology

- the **methodology is hybrid** in the sense that it is structured as well as object-oriented. The development process defined by the methodology is structured - it distinguishes between data and functions in various steps in the analysis stages (Global Data Analysis versus Business Processes Analysis in the stage Global System Analysis for example). On the other hand there are object principles supported directly by the development process and techniques in the Detailed System Analysis stage. The methodology uses the term "Model of Reality" which includes both data and processes in their natural unity via so called "Entity Life Processes".
  There are several reasons for this:
  - client-server architecture of database systems naturally leads to object-oriented thinking. Distributing both processes and data to the server and the client part of the application[1] forces the developer to take into consideration the natural unity of data and processes even at the conceptual level of system design. At the conceptual level of system analysis the developer can find reasons for decisions about which data are to be placed in the central (i.e. server) part of the application and by which (server) processes are to be manipulated. The common concept for the unit of such data and processes of the server part of the application is the "entity" in the sense of the "object".
  - on the other hand currently existing object oriented conceptual analysis methodologies are not mature enough to solve all potential problems connected with application development. Especially at the level of global analysis it is impossible to describe all aspects of the business only using the terms "object", "object method" and "object communication". There is an actual need to distinguish between conceptually essential objects such as "customer", "order" or "invoice" and the objects which reflect just the user point of view of the application as are "document" or "customer dictionary". But even at the level of detailed design we need to discover the essential order of the entity data manipulating processes (object methods) and its relationship to the objects mutual behaviour (communication between objects). These facts lead to the need to describe the developed system also in the "structured" way - using such terms as "data entity", "entity updating function", "input-output function" etc.
  - the conceptions of an available environment strongly separate the data and the process side of the application. It is based on a relational database with it's

[1] The terms "server" and "client" are used here in the sense of "logical server" ("logical client") - i.e. independent of used client/server architecture model (2-tier, 3-tier) and on the technical client-server partitioning reasons. Although there are various technical reasons for partitioning the processes and the data to the central and the other parts of the technical environment (for example system performance, technical flexibility and stability etc.) the nature of the concepts "server" and "client" remains the same as in the conceptual point of view. Client parts represent the input and presentation part of the application while server parts represent the underlying common business logic of the application based on real-world behaviour.

typical one-sided conception of application technology support - it is oriented on realization of data structures. The Functionality of the system is only supported statically - via integrity rules and event triggers. These environment features require traditional - structured way of thinking. On the other hand the client parts of the environment follow the concepts of the object oriented programming using the concepts of Windows applications design.

- the methodology is based on **the use of prototyping.** Prototyping is the natural approach for using 4GL environment in client - server architecture. Client - server architecture offers good conditions for separating the design of the server and the client part of the application. Such separation enables "safe" use of prototyping: the prototype should concern the external face of the application, not the internal system behaviour. Internal system behaviour follows directly from behaviour of the reality which the analyst has to consider as independent of end-user view and the technology. Therefore it should be realized as the server part of the application. On the other hand the client part of the application should represent first of all the end-user view of the information system and it is always strongly dependent on used technology. So the client part of the application is that part which is to be the subject of prototyping. Prototyping is a very effective way to:
    - gather the proper information about the user needs concerning both the data and the behaviour of the system (the user's particular point of view)
    - early define the particular shape of the user environment by consulting the user about prototype.

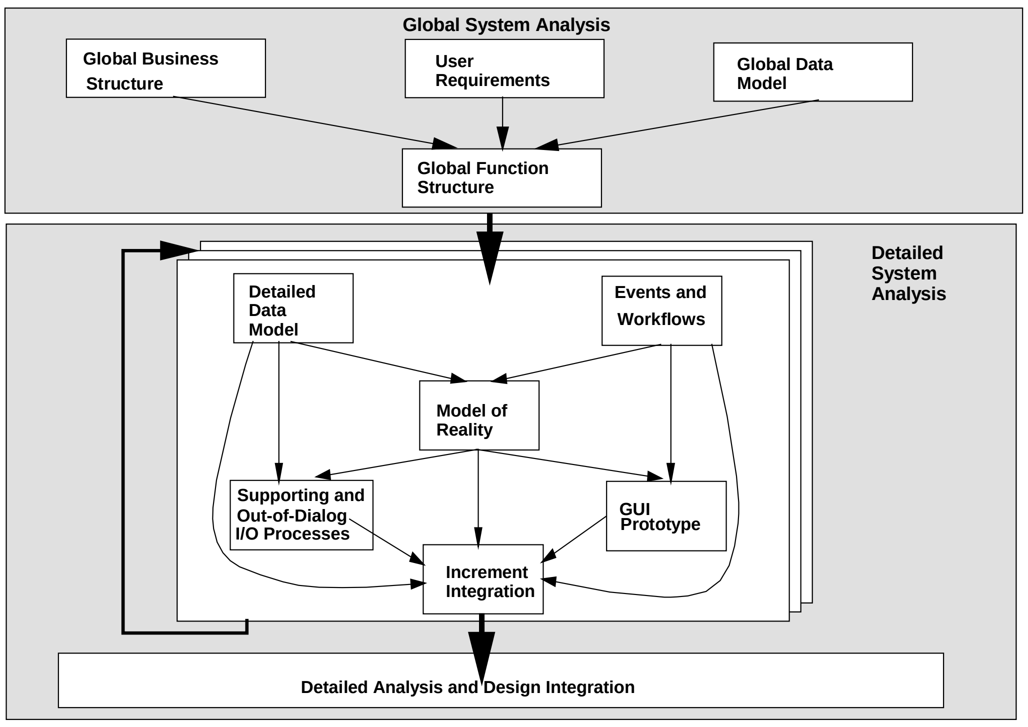
- initially requirements of the **client-server partitioning** of the application are derived **from conceptual analysis**. Then technology and implementation reasons for the partitioning are taken into consideration. The reasons for such an approach closely correspond with the reasons for using the prototyping only on the client part of the application discussed above. The server part of the application should realize first of all the essential system functions taken from conceptual analysis which describe the essential real world behaviour. Such functions are independent of a particular shape of the realized information system and of used technology as well as the end-user view of the information system. Essential system functions form the process part of the real world model hidden inside the information system and the essential system data form the data part of that model. As the conceptual data model describes WHAT the information system process, there are connected conceptual functions which describe HOW these data are processed. Such approach to the information system development inevitable leads to unified view on both data and processes - i.e. to the object oriented approach.

Outline of the Analysis Procedure

The process of the PDIT consists of stages. Each stage consists of several steps, each step consists of several tasks. For the purpose of this paper the two key "analysis and design" stages are important:

- Stage Global System Analysis and
- Stage Detailed System Analysis.

Similarly, not all steps inside each stage and not all tasks inside each step are important for the purpose of this paper. So, for example, a standard step of each stage: Stage End Assessment which consists of standard project management activities is not described in detail here.



This picture illustrates succession of the main steps of the analysis stages.

In the stage **Global System Analysis** the methodology uses a clear structured approach. This stage begins with the analysis of business processes, user requirements and business data (i.e. global data sets). As the result of balancing these three factors the global function structure of the information system is formulated. Tools and techniques used in this stage are the Data Flow Diagram and Entity Relationships Diagram. There are two main outputs of this stage - Global Data Model and Global Function Structure. The purpose of the stage Global System Analysis is to give the outline of the whole system structure as the basis for incremental detailed development of particular parts of the system. Therefore the decomposition of the system structure must be in order with assumed process of detailed (object-oriented) analysis of its particular parts. The main requirement is that the partitioning of system functions must very closely match the partitioning of the data model. For that purpose particular rules for parallel data model and function structure development are formulated.

Stage **Detailed System Analysis** combines both structured and object-oriented approach. The object of the analysis is here one part of the global system structure - one system increment. The central point of this stage is object conceptual model of developed information system - so-called Model of Reality. As the basis for the model of reality the detailed data model, workflows and reality events are used here. The model of reality is

clearly object-oriented: entities from the data model (conceptual objects) are completed by processes of their life histories. Then the communication of the objects is described and balanced with their life histories. Balancing consists of determination of such points in the object life in which communication (i.e. inputs and outputs) of the object occur. Each such specific point of the object's life must be joined with a specific real-world event. So the real-world events and the needs of the entire communication between objects are the basis for the object life history specifications. On the other hand the object life history process is the basis for seeking the new real-world events and objects communication needs.
Besides the model of reality which describes the central part of the system functionality also the other function parts of the system have to be described:

- the prototype of user interface (GUI prototype)
- out-of-dialog input/output functions (periodical batch inputs and outputs, interface to other systems etc.)
- other supporting functions

While development of the model of reality is object-oriented the tools and techniques used here as well as the overall process of the detailed analysis is structured. For the description of entity life histories the methodology uses a State Transition Diagram, communication of the objects is described with the use of Data Flow Diagram. For the purpose of conceptual description of user interface based on the workflows the methodology uses a Windows Navigation Diagram.

Underlying Principles

There are two main principles which are present in all features of the PDIT methodology:

- the principle of modeling and
- the principle of "Three Architectures"

Modeling

The principle of modeling 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 Entity Relationship Diagram (ERD) has been developed as the special tool for description of the essential characteristics of the real world: objects and their mutual relationships. It is constructed to be able to exactly describe 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 implementation characteristics of the environment in which the database is realized. But the essential shape of the model still remains the same. Because of the need to describe the same database in it's various shapes (essential, technological, implementational) the principle of different architectures have been formulated. This principle, generalized to the scope of the whole system is discussed below.

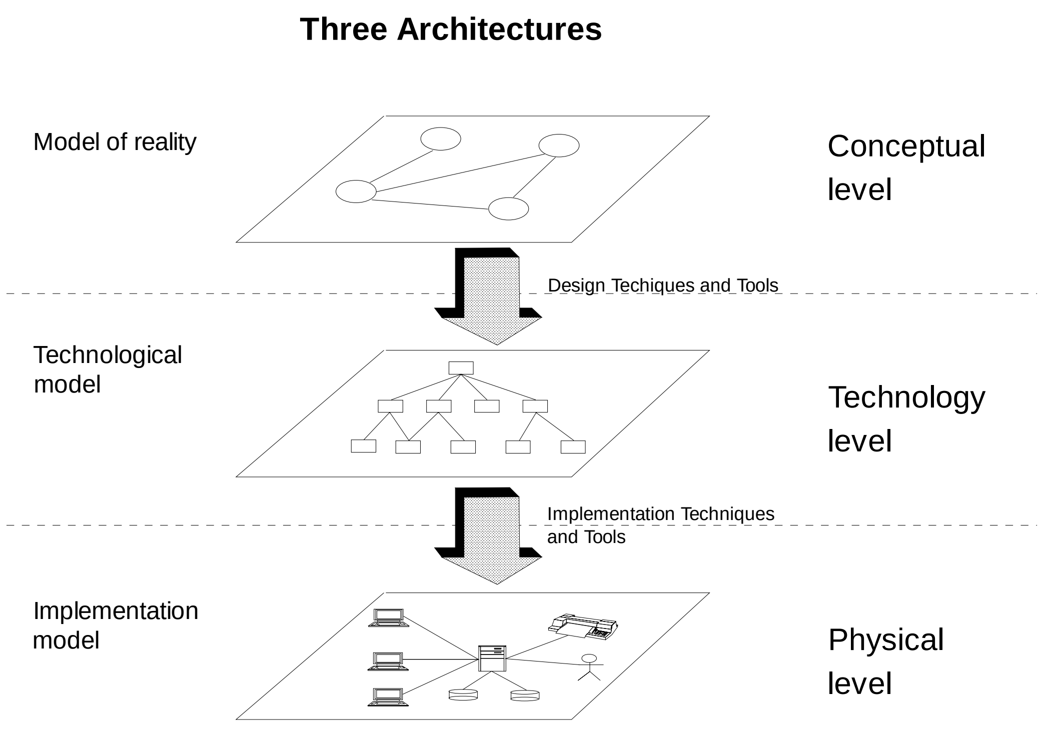
Modeling principle proves to be general in the sense that it is valid not just in the area of system data. Also some parts of system processes have to be regarded as the model of the real world. Key task for putting this principle into practice is to recognize which system processes form the model of the real world and which do not. Such recognition requires separation of the modeling operations from the other ones and organizing them into the special algorithms according to real world objects and their relationships. This way organized modeling algorithms represents the essential controlling algorithms of the entity life histories.

Three Architectures

The principle of "Three Architectures" was referenced in the paragraph in which the "modeling 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 Modeling Principle into practice. Without such separation the developer wouldn't 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:

- **the conceptual model** represents a clear model of the real world which is not distorted by the non essential aspects given by assumed technology and implementation environment of the system
- **the 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 realization of the database leads to the data structures considerably distant from the conceptual entities and their relationships. On the other hand relational database technology preserves maximum of the original shape of the data model. So the degree of shape changes always depends on the technology used.
- **the implementation model** represents the final shape of IS. It depends on the used technology taken into consideration in the technological model and respects also implementation details given by the used particular environment. Thus the implementation 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 next figure:



Such model of the three different views of 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 a 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 a specific goal, a specific type of developer's activity and specific techniques and tools to use. Also the transition of the design from one to the next level requires specific activities, techniques and tools.

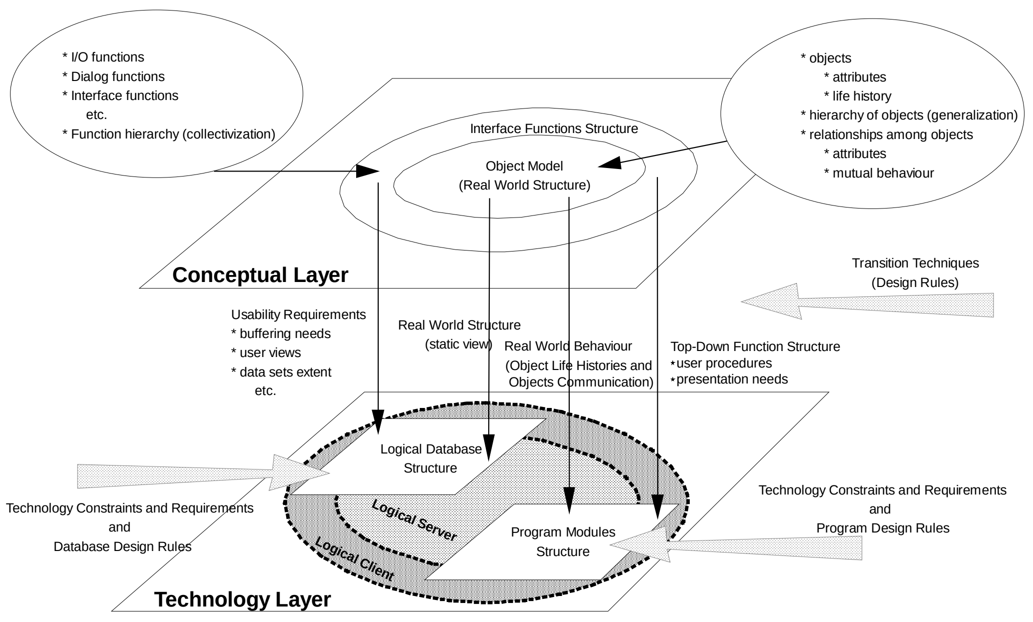
Principles of modeling and three architectures manifest themselves in the PDIT methodology in various ways:

- the central part of the system functionality is the Model of Reality. The model of Reality is based on a detailed data model which represents a static view of real-world objects. Dynamic aspects of the real world (real-world behaviour) describe the entity life processes (which describe the behaviour of the entity) and the communication of entities (data transfer between the functions ordered via entity life process). Particular entity actions are caused by real-world events. So the "entire" system functionality is the **model of real-world behaviour** as well as the "entire" system stored data structure is the **model of real-world structure** (data model)

- on the **conceptual layer** the structure of the system consists of the Model of Reality and the conceptual interface functions structure which includes all shapes of interface processes:
    - user interface (dialog) functions
    - out-of-dialog I/O functions
    - interface supporting functions etc.
- On the **technology layer** the object-oriented conceptual Model of Reality is transformed in the structured manner into the logical database structure and the program modules structure. This is necessary because used technology is based on the "structured" separation of the "data part" of the system (relational database) and the "process part" of the system (4GL procedures and host language procedures)

Features of the PDIT methodology discussed above illustrates the following picture.

Transition From the Conceptual to the Technological Architecture



Model of Reality

The development of the model of reality is the central point of the analysis part of the methodology. The methodology regards a set of connected models of the entities behaviour as the reality model. The purpose of the entity behaviour model is to describe entity life histories. The purpose of the model of reality is to describe all important (i.e. essential) features and rules of the reality behaviour.
Model of reality consists of

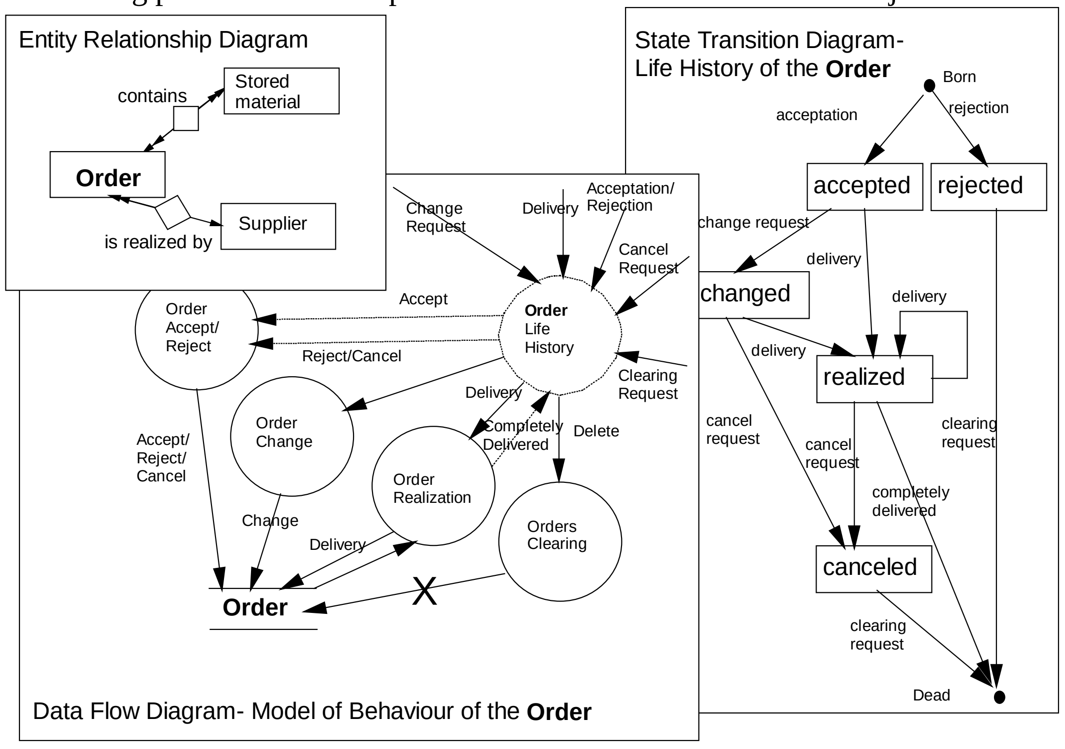
- a set of Data Flow Diagrams (DFD) and
- an Entity Relationship Diagram (ERD)

Each **Entity Behaviour Model** (DFD) describes the behaviour of one object (entity or relationship) from the Data Model (ERD). The Entity Behaviour Model consists of:

- **entity** (or relationship in the sense of associative entity) from the Data Model which expresses the data part of the object
- several essential functions which express the actions of **object state changes**
- one control process described in the State Transition Diagram (STD) which describes the **object life history** as the succession of the entity states and transitions from one state to the following one(s)
- control flows between the STD and essential functions which describes **the events and the actions of the object**. Each input control flow expresses one event stimulating the transition of the object from one state to the other one. Each output control flow from the STD to the essential function expresses one action of the object as the reaction on the event
- **data flows** which are of several types:
  - flow of data from the input/output interface to the essential function of the Entity Behaviour Model. Those data expresses the **data part of the real-world event**
  - flow of data from (or to) the essential function of the other object. Such data flows describe **communication of the Entity Behaviour Models** - transition of the information about real-world events between objects

In practice the developer does not have to describe all of the object behaviour models. In the case of simple entity life history (i.e. a life history which consists of just two states: the beginning and end of the entity life) it is not necessary to describe it. This is because the functionality of such object is too simple to regard such object as something more than just a data entity.

The following picture is an example of the model of behaviour of the object Order.



Conclusions

The main purpose for the development of the PDIT methodology was to develop a sophisticated methodology which will naturally allow:

- the respect of the features of the client-server technology even on the level of conceptual analysis
- the use of the GUI development environment
- the use of the integration features of CASE tools.

This purpose led us to the development of the hybrid methodology which combines a structured approach with object-orientation in the field of design as well as analysis.
Reasons for the hybrid approach are:

- immaturity of the OO analysis methods on one hand and insufficiency of the structured methods in the case of the client-server technology on the other hand
- technology constraints of available environment - client-server environment is based on relational database which forces the "structured" style of the IS development
- quite good support of quite maturely structured development tools:
    - the possibility to generate the database structure from the conceptual data model in the CASE tool and possible reverse engineering ability
    - the possibility of generating the part of interface function structure using GUI development environment

We regard the usability and efficiency which follow from this hybrid approach to be the strong features of PDIT methodology.

The weakest point of the methodology in the current state is the transition from conceptual specification of the model of reality to its technology realization. The methodology supports this activity using a set of rules for the implementation of particular typical constructions in the model of reality. But there is not any formal technique for supporting this activity. One possible way forward is to develop the technique based on the "program inversion technique" taken from JSD (M.A. Jackson). Unfortunately, such a technique is strongly dependent on the target implementation environment and it is a hardly supportable by structured development tools (CASE Tools). So this transition activity will be the main topic for the future work on the methodology.

References

Chen P.P.S.: The Entity Relationship Model - Towards a Unified View of Data, ACM TODS, Vol 1 No.1, 9-36, 1976.

Coad P.,Yourdon, E.: Object-Oriented Analysis, Prentice-Hall Inc., NJ, 1990.

Date C.J.: An Introduction to Database Systems, Addison-Wesley, Massachussetts, 1977.

Jackson, M.A.: System Development, Prentice-Hall Inc., Englewood Cliffs, NJ, 1982.

LBMS - Process Engineer - User Manual, London, 1993.

Martin, J., Odell J.: Object-Oriented Analysis and Design, Prentice-Hall Inc., Englewood Cliffs, NJ, 1992.

PDIT - Functions and Data Analysis, ITC PragoData, Prague, 1994.

PDIT - Process Modeling, ITC PragoData, Prague, 1994.

PDIT - Client-Server Partitioning, ITC PragoData, Prague, 1994.

PDIT - Server Design, ITC PragoData, Prague, 1994.

Repa V.: Seeking the Actual Reasons for the "New Paradigm" in the Area of IS Analysis, Proceedings of the ISD 94 International Conference, Bled, 1994.

Rumbaugh J.,Blaha M.,Premerlani W.,Eddy F.,Lorensen W.: Object-Oriented Modeling and Design, Prentice-Hall Inc., Englewood Cliffs, NJ, 1991.

Yourdon, E.: Modern Structured Analysis, Prentice-Hall Inc., Englewood Cliffs, NJ, 1989.