7

Flowcharts, data flows, SADT, IDEF and NIAM for enterprise engineering

M. Zgorzelski
GMI Engineering & Management Institute
Flint, MI 48504, USA
Tel: (810)762-7841, e-mail: mzgorzel@nova.gmi.edu

P. Zeno
MIT Sloan School of Management
Cambridge, MA 02109, USA
Tel: (617)367-8439, e-mail: pzeno@mit.edu

Abstract

Numerous modeling tools of diagrammatic nature are available today, whether for incremental (TQM style) or radical (reengineering, BPR style) enterprise improvement. In practice, two simple techniques dominate: flowcharts and data flow diagrams. Very few reengineering specialists realize that these techniques are too primitive and inadequate when used for large scale business process analysis. IDEF0 and IDEF1X, more sophisticated tools, have also been used extensively for BPR purposes, but also with mixed results. The authors point out to the need of further development work, and present an outline of their proposal (called NIDEF) based on the synthesis of NIAM, SADT and IDEF0.

Keywords

Reengineering, business process, IDEF0, SADT, NIAM, NIDEF

1 INTRODUCTION

Many modeling tools of diagrammatic nature are available today, whether for incremental (TQM style) or radical (reengineering, BPR style) enterprise system analysis, redesign and improvement. In practice, however, two simple but well known techniques dominate: flowcharts (together with their modification known as workflow diagrams) and variations of data flow diagrams (DFD's). Very few reengineering practitioners seem to realize that these techniques are too primitive and inadequate when used for serious, large scale business process rengineering. Activity modeling with IDEF0 and data modeling with IDEF1X, certainly more sophisticated, and

G. Doumeingts et al. (eds.), *Modelling Techniques for Business Process Re-engineering and Benchmarking*

somewhat standardized in the US tools, have also been used extensively in many BPR undertakings, but also with mixed results.

2 FLOWCHARTING AND DATA FLOW DIAGRAMMING

Flowcharts

Flowcharts and data flow diagrams (DFD's) were introduced as computer software design tools and serve this purpose to this day, at least in the domain of traditional, procedural computer programming. When applied to real life business activities modeling, flowcharts have serious shortcomings. Trying to use flowcharts to analyze a business system and then to generate its reengineered model, we quickly discover that these charts, especially in their popular versions can - of course - show the sequence of events in a single process, but they do not provide any means to show multiple potential process paths, as well as their parallelism, synchronization, interrelationships of multiple processes or multiple thread feedbacks. Simple traditional flowcharts also do not support multi-level, gradual decomposition of processes and activities, needed to analyze and design new systems to a satisfactory level of detail.

There do exist some more sophisticated versions of flowcharts, allowing multiple process threads, and multi-level hierarchical decomposition of processes. These are used mostly in computer control software design. They do not fit at all the purpose of enterprise modeling at a high level of aggregation and abstraction. One can hardly envisage their application to rather ambiguous and naturally fuzzy general enterprise modeling.

Data flow diagrams

Data flow diagrams, the popular *bubblecharts*, can show process parallelism and interrelationships, but are limited to function as the design tools of information processing systems. They describe, in the functional (or procedural) way the processing of data. DFD's are useless when it comes to the analysis of processing of real world objects (products, parts, services, etc.). Furthermore, DFD's use only data inputs and outputs. The ability to make the distinctions between activity inputs, outputs, controls and mechanisms, inherent in SADT and IDEF, is missing in standard DFD's, and this renders them almost useless for constraint limitations studies, resource needs and utilization analyses. Techniques such as Activity Based Costing, indispensable in any transformation of a conventional business into a reengineered enterprise, require the analytical tools allowing clear distinctions between various types of resources and constraints.

DFD's, however, have an advantage of providing concepts which are badly missing in the approaches used by the other contender to the honorary title of The True Toolbox of Reengineering: the SADT/IDEF family of methods. These are simply: sources/sinks and storages. Representation of external and internal suppliers and customers can be done through sources and sinks in DFD's (although obviously only for data, not for real world objects!). DFD storages can be utilized to represent inventories and databases. The availability of the source/sink and the storage symbol in DFD's is one of the reasons for the practical success of this technique in reengineering, in spite of its other limitations.

3 MODELING ACTIVITIES IN IDEF0 AND DATA IN IDEF1X

The use of IDEF0 and IDEF1X is certainly beneficial and became quite popular in a variety of TQM and BPR undertakings in the US. IDEFs however, in their present form exhibit serious difficulties in addressing some of the very common, typical business system analysis problems, and need some more conceptual development.

IDEF0 problems

The impossibility to clearly identify (in "classical" IDEF0 models) **objects**: various groups of internal and external customers and suppliers of the business processes diagrammed, is the first of the difficult problems in the business use of IDEF0. Its shortcomings become particularly obvious here when, to design a customer-focused lean/agile enterprise, a model of the activities occurring between vendors, suppliers and final product manufacturers, as well as customers - has to be generated. Those objects - fundamental components of the process cannot be clearly identified in a standard IDEF0 (activity-only) model, making it difficult, if not impossible, to study the overall process. A business process, after all, is commonly defined as any group of activities that takes an input, adds value to it, and provides an output to an internal or external customer. Evidently, there exists a serious modeling problem, if we are unable to model explicitly objects, such as a customer, because of IDEF0 limitation to activities only. In fact, data flow diagrams handle this problem slightly better, external sources and sinks at least are among the readily available concepts in this modeling technique, as we have mentioned earlier - thus external suppliers and customers may be modeled in DFD's.

Another serious weakness may be defined as an inherent inability of IDEF0 to show the critical difference between traditional, functionally divided organizational structure of an enterprise, and the modern process-oriented, teamwork-based organization. Changing the organizational paradigm of an enterprise, from functional to process oriented, is one of the most essential current trends in management. IDEF0, with its limitation to activities, irrespectively of who or what performs the activity, is obviously unable to clearly expose the reasons for the ineffectiveness of the functional organization as well as clearly show the sources of present day success of the process/team approach. The only way in which "doers" of activities may be shown in IDEF0 is through the specification of their mechanisms, not as objects. This limitation does not allow the analyst to show, in an IDEF0 model, that functionally organized systems fail because they exhibit convoluted communication patterns between multiple objects (traditional functional organizational units, structured in hierarchically positioned layers of management). In such organizations the fragmented activities constitute eventually a process, which is, however, very inefficient when compared with streamlined, process-focused organizations. Modern self-managed teams show very tight coupling and short, effective communication lines between the process and its "doers" - the process team. Modeling and analyzing this issue and variations of possible organizational solutions for an enterprise is impossible in IDEF0, as it does not provide the way to explicitly study objects - people, organizational units, functional units, etc. combined with the activities they perform. It may be worth to notice, that as far as this particular problem is concerned, data flow diagrams are even less adequate than IDEF (as they do not provide the **mechanism** concept and thus do not show "doers" of activities at all!), thus the issue

just raised is rather a more general reflection about the current incomplete state of development of enterprise modeling methodologies, not just a controversial score point in the ongoing match between data flow diagrams and IDEF's.

The lack of a clear distinction (in IDEF0) between flows of material objects and flows of information (or data) results in one more IDEF deficiency: its inability to expose problems occurring when it comes to the separation of real life objects and their related data, frequently their identifying data! A very typical manufacturing process, where a component is attached to a final product and the serial number of the component is stored in a database together with other information related to that particular unit of the final product - cannot be correctly modeled without the ability to show the two flows: of parts flowing through the assembly line and of related data simultaneously processed by the computer systems. Tracking the causes of inaccurate data in the databases and improving on the process can be hindered by the inability to identify the complete sequence of events, occurring in two separate chains of activities: the product assembly process and the accompanying, but separate, data manipulation process. To make the last paragraph possibly more clear: IDEF0 is capable of showing both - flows of data and flows of real objects. The problem with IDEF0 lies in the fact that both types of flows (semantically different) are shown in this methodology with the same (syntactically) symbol: a simple solid line arrow. An additional element of confusion here is created by the fact that arrows in the original US Air Force IDEF0 documents, are referred to as **data**; it is only in the recently established IDEF standards that the dual role of the arrows (object/data) was finally recognized and started to penetrate into everyday use of IDEF0. The solid line arrow for the symbolic single representation of both objects and data remains, however, in effect in the IDEF0 federal standard.

Impossible integration of IDEF0 and IDEF1X

There also do exist significant difficulties in integrating the two fundamental IDEF methodologies: activities modeled in IDEF0 and data modeled in IDEF1X. There are two reasons for these difficulties:

1. Not all arrows in IDEF0 represent in effect data in motion between activities; some really do represent data (i.e. they correspond to IDEF1X entities and their attributes), others, however, indicate real life objects, which may only indirectly, if at all, be represented by data and thus do not fit the IDEF1X format. Suggestions, that in order to integrate IDEF0 and IDEF1X models, one should limit IDEF0 arrows to 'pure' data, amount simply to the avoidance of a serious problem. In present authors' opinion this problem is unsolvable within the present conceptual framework of IDEF0/IDEF1X.

2. Both: real life objects and data, while at rest (in an enterprise system) will sit inside some appropriate storage facility (respectively, a warehouse and a database). The structure of data at rest can be represented by IDEF1X entities, but we do not have, within the IDEF family, a technique to show real objects, their structure and their storage. IDEF0 does not even provide a symbol for data storage, something readily available in DFD's.

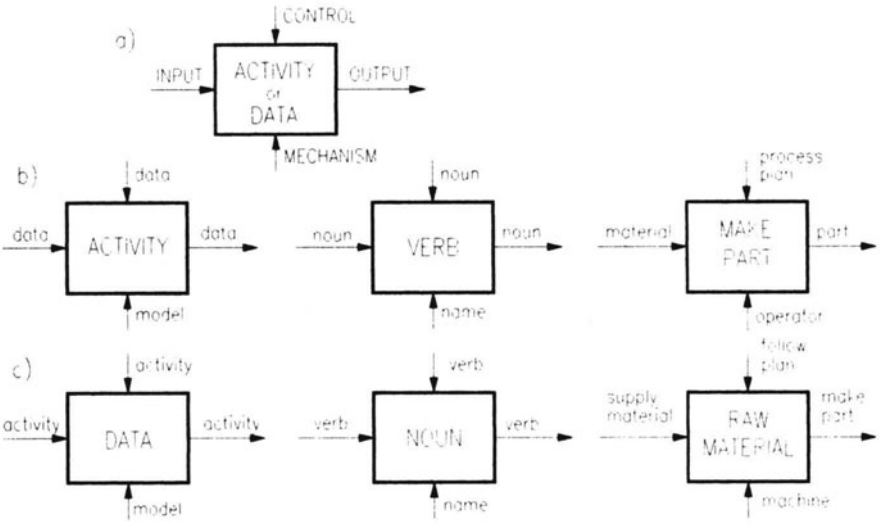

Figure 1 The system modeling constructs of SADT: basic box (a), activity-centered (b) and data-centered (c) representations, after Ross, 1977. Combination of the two techniques, rather than abandoning (c) in favor of (b), as done in IDEF0, is the cornerstone of authors' NIDEF proposal.

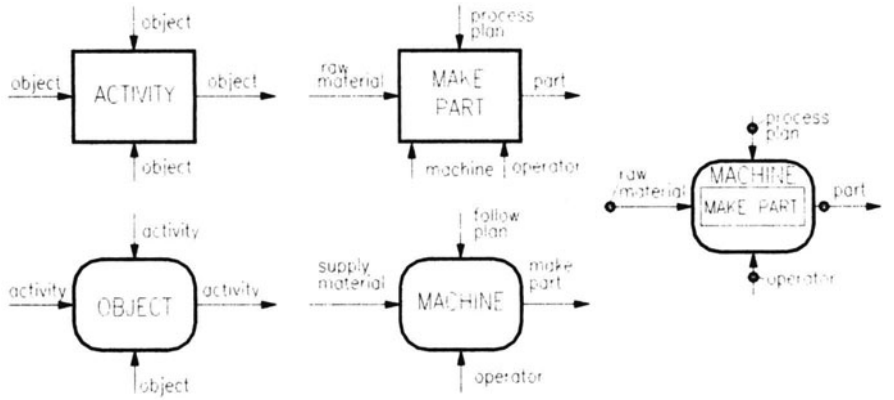

Figure 2 Authors' NIDEF (NIAM-ized IDEF) proposal: traditional activity-centered (above) and proposed new object-centered (below) diagrams. This modification may be extended into combined object-activity models illustrated by the single box on the right. We also introduce here symbols for the processed objects (e.g. raw material, part) flowing in the system through the active processors (such as the machine tool shown).

4 NIAM AND OBJECT-ORIENTED METHODOLOGIES

NIAM

Among the widely known methodologies, only NIAM - Nijssen Information Analysis Methodology (Nijssen and Halpin, 1989, Wintraecken, 1990) clearly separates real

life objects and data about them. NIAM, however, is primarily perceived and developed today as an information modeling tool, aimed at the design of relational databases, and it is used today mainly as the object-role database design technique in some CASE tools, in effect an alternative to the entity-relationship modeling of the IDEF1X type.

NIAM objects are potential entries into relational database fields, and thus they are 'atomic' in nature, i.e. they cannot be decomposed into constituents - subobjects (i.e. NIAM does not support hierarchical decomposition of objects). Nijssen and Halpin (1989) provide in addition to information modeling a rudimentary technique (Information Flow Diagrams) to describe information processing activities. This technique, in accordance with Nijssen's philosophy, also makes the critical distinction between the processing of information flows and the transformations of real world objects, but unfortunately it does not provide any form of hierarchical decomposition of activities, and thus it is rather not useful outside the information processing domain. To provide this possibility NIAM has to be combined with some concepts derived from SADT and IDEF0. Some preliminary ideas as to how this could be done were proposed earlier by the present authors in the form of a modeling tool called NIDEF (NIAMized IDEF). A brief outline of this concept is presented in this paper.

Object-oriented programming methodologies

The object-orientation paradigm, which has currently a fundamental impact upon computer programming, seems to have influenced only to a small extent the way we think about, real world systems. There are obviously significant differences between real life objects and abstract computer programming object concepts; but there are also surprising similarities and some valid analogies. The concept of object class seems to be valid both for the abstract and the real objects. Also the idea of encapsulation is a valid approach both in object-oriented programming (when applied to software objects) and in, say, business system analysis. It may be applied to a machine tool, an assembly line, or a complete plant, again each of these encapsulating its components, its attributes and its methods, i.e. its potential activities. Inheritance, another element of the object paradigm, is not that easily translated into the real world concepts from the computer abstractions.

There are also some significant differences, however, between the real world and the information world. The only function of computer programmers' objects is to send messages to other objects, resulting in their transformations (if they fit allowed methods of the recipient object). Real life objects perform activities upon other objects, transform them or -frequently - create new objects. To make things more complicated, a real life activity performed upon real life 'things' may require the cooperation of several objects: an object machine tool, together with several other objects: tools and fixtures, and under the control of another object - operator - may be, for instance, transforming the raw material object into a finished part object. This, common situation, cannot be translated into the present day computer programmers' terminology of objects, methods and messages, without bending over backwards!

The object-orientation paradigm has not found -so far- its way in any significant manner into into enterprise modeling. Although a huge number of object-oriented methodologies exists today (see, for example Coad and Yourdon, 1991, Shlaer and Mellor, 1992, Firesmith, 1993, Booch, 1993, Page-Jones, 1994), they all have computer system analysis and design as the objective, and they do not apply readily, if

at all, to direct modeling of enterprise systems per se, not of their information processing subsystems

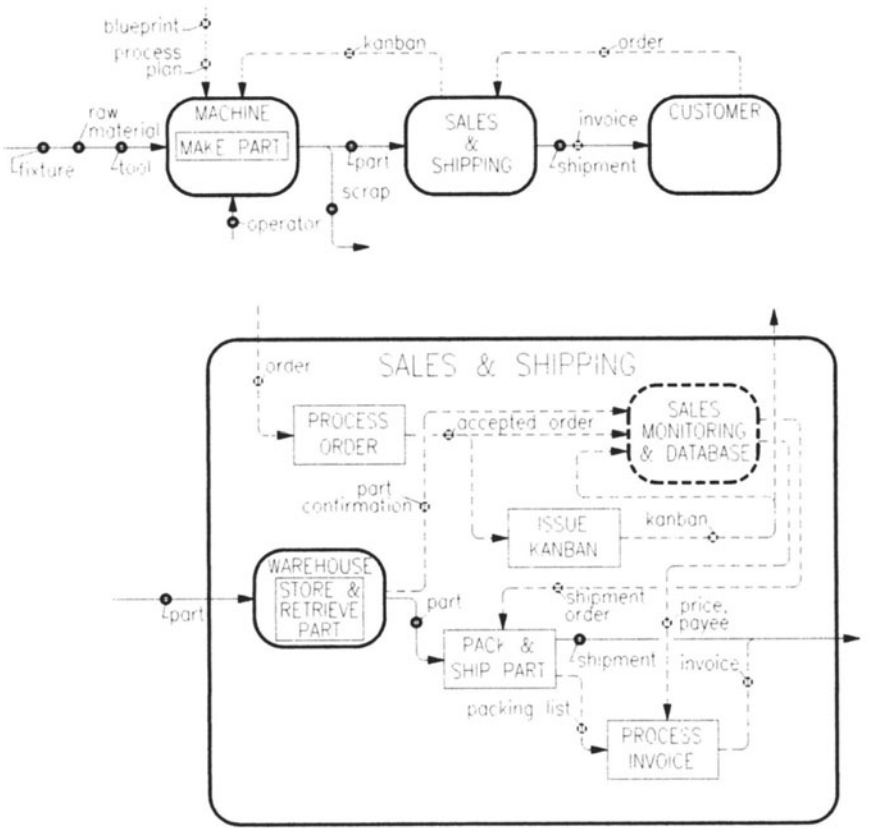

Figure 1 An example of a NIDEF model of a business process (upper diagram) with one object ("Sales & Shipping") hierarchically decomposed into the lower diagram. Solid lines represent real objects flows, dashed - information flows.

5 THE NIDEF SYSTEM MODELING TOOL

NIDEF combines the lines of thinking about activity/object modeling proposed originally by Munck and Braun (1992) with NIAM object-role modeling. Figure 2 illustrates the basic modification of Ross' 'datagrams' into 'objectgrams'. This leads to the diagramming technique illustrated in Figure 3. NIDEF preserves basically unchanged some of the powerful features of IDEF0: hierarchical decomposition, and the input-control-output paradigm. SADT/IDEF0 mechanisms disappear - **objects** performing activities take on the mechanisms' function. Significant additions are in fact limited to the use of: objects, in the roles of mechanisms supporting activities, as

well as storages, sources and sinks - and two types of arrows: solid line for flows of processed objects, and dashed for flows of information. Objects in NIDEF can be linked then by flows of real objects, and flows of data. Objects can also enter into relationships, NIAM style. The way in which NIAM-type modeling is represented in NIDEF should be clear upon inspection of our Figure 4 and 5.
We have to state very clearly here that NIDEF is not yet a complete methodology; it is continuously under development. A careful reader will find certainly some differences between the presentation given here and our earlier publications. We have tried a number of ideas and approaches, and we try to react to various extremely valuable suggestions we have received in the course of previous discussions of this concept. By no means we want to create the impression that we claim here to have found The True Toolbox of Reengineering.

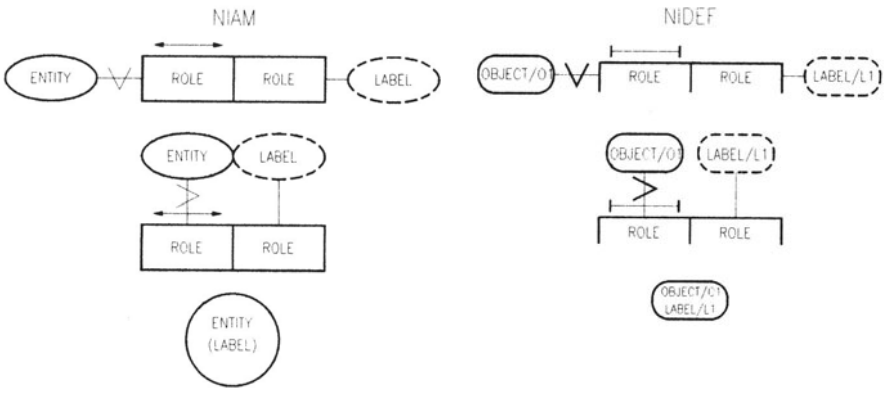

Figure 2 For the purpose of combining object-role (NIAM style) modeling with object-activity modeling, and avoiding symbology confusion - NIDEF introduces some slight modifications of the NIAM symbols as shown here.

6 REENGINEERING - ROADS WITHOUT MAPS, BRIDGES WITHOUT BLUEPRINTS?

In a reassesment of the present situation we could say that in the absence of a generally accepted graphical representation language, many ambitious enterprise improvement and/or reengineering projects, resemble (at least from an engineer's perspective) efforts to build bridges without blueprints or roads without maps. At best these efforts may be described as doing sophisticated engineering without some common, generally accepted, engineering drawing technique, the common language of communications for all engineers. From the viewpoint of an engineer, this kind of a careless approach is a guaranteed prescription for a disaster. Some catastrophes of this kind the present authors had a chance to see with their own eyes in the very

recent past. Ambitious projects of the "Factory of the Future" type have been abandoned, and installations costing hundreds of millions of dollars dismantled, to much extent because nobody ever took time (and possessed the necessary skills and techniques) to roadmap how these establishments were supposed to operate. These reengineering disasters were certainly not as dramatic and spectacular as, for instance, the failure of a poorly designed bridge. In fact, frequently these unfortunate disastrous outcomes of some *grandiose* projects are hidden from the public by involved managements, and almost never analyzed in detail as to their 'failure mode and effects', as opposed to the - usually very rigorous - screening of the reasons of failure of some typical engineering project or machine. It is thus usually rather impossible to determine to what extent the lack of proper modeling tools and techniques may have contributed to a particular disaster, and to what extent the negative outcome may be

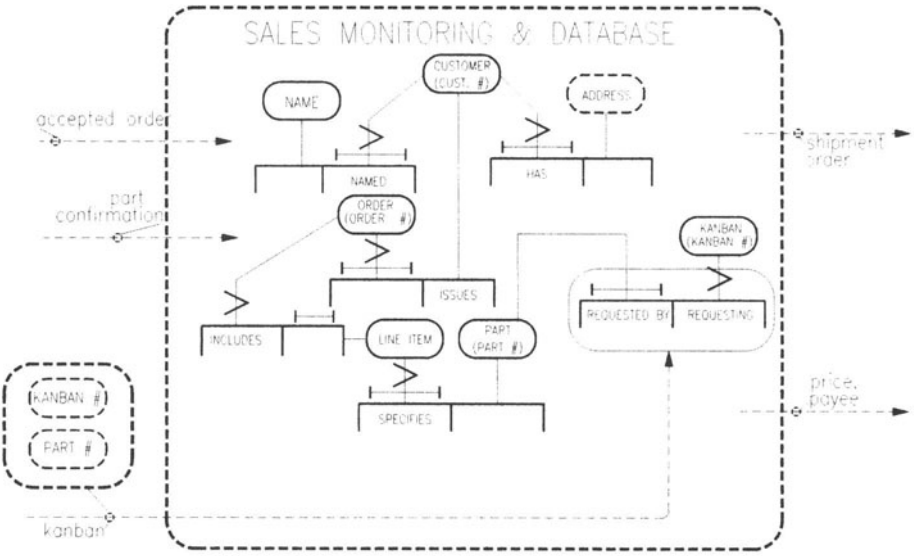

Figure 3 Further NIDEF-style hierarchical decomposition of the "Sales Monitoring & Database" object from Figure 3 is shown here. This decomposed object contains numerous sub-objects, which enter into typical object-role situations. Obviously, only a fragment of the model of information within the database and the monitoring system is shown here for the illustration of the principles.

attributed to other factors.

It is generally known that TQM/Continuous Improvement and Business Process Reengineering undertakings have failed in a huge number of cases. For TQM type of efforts - failure rates are currently estimated at 60% (sometimes even up to 80%) - see Brown et al. (1994). For BPR similar numbers are given - around 60% of failures, see Hall et al. (1993). Unfortunately, among the many reasons given for failures, one seldom finds the lack of precise modeling tools and the lack of use of correct, sophisticated mapping techniques for business processes in question. These problems are still rather rarely noted by the 'mainstream' business reengineering authors' community, evidently happy with their sketchy and ambiguous analysis and design tools.

7 REFERENCES

Booch G. (1993) *Object-Oriented Analysis and Design*, Benjamin/Cummings Publishing Co.

Brown M.G., Hitchkock D.E., Willard M.L. (1994) *Why TQM Fails and What To Do About It*, Irwin, New York.

Coad P.,Yourdon E. (1991) *Object Oriented Analysis*, Prentice Hall

Firesmith D.G. (1993) *Object-Oriented Requirements Analysis and Logical Design. A Software Engineering Approach*, Wiley. 1993

Hall G., Rosenthal J., Wade J. (1993) How to Make Reengineering Really Work, *Harvard Business Review*, Nov.-Dec.

Munck R., Braun C. (1992) *IDEF0-O: Object-Oriented SADT*, Fall '92 IDEF Users Group Conference Proceedings.

Nijssen G.M., Halpin T.A. (1989) *Conceptual Schema and Relational Database Design. A fact oriented approach.*, Prentice Hall.

Page-Jones M. (1994) Object-Oriented Design Notation, *Report on Object Analysis & Design*, Vol. 1, No.1.

Ross D.T. (1977) Structured Analysis (SA) A Language for Communicating Ideas *IEEE Transactions on Software Engineering*, Vol. SE-3, No.1 pp 16-34.

Shlaer S., Mellor S. J. (1992) *Object Lifecycles: Modeling the World in States*, Yourdon Press.

Wintraecken J.J.V.R. (1990) *The NIAM Information Analysis Method. Theory and Practice*, Kluwer Academic Publishers.

Zgorzelski M., Zgorzelska P. (1993) *IDEF and NIAM: on the Possibilities of Convergence of Two Outstanding Methodologies*, in Spring '93 IDEF Users Group Conference Proceedings.

Zgorzelski M., Zgorzelska P. (1994) *On the Use of SADT, IDEF, NIAM and Related Methodologies in Lean Agile Manufacturing System Studies*, in Proceedings of ISATA Dedicated Conference on Lean/Agile Manufacturing in the Automotive Industries, Aachen.

Dr. Maciej Zgorzelski was born and educated in Poland (he holds M.S. in Mechanical Engineering and Ph. D. degrees from the Technical University in Warsaw). Following that he did a post-doctoral study at the MIT. He taught at the Technical University in Warsaw for several years, held visiting assignments in various countries, and managed industrial R&D. In 1983 he emigrated from Poland and settled permanently in the US. He is currently a Professor of Mechanical Engineering at GMI Engineering and Management Institute (previously General Motors Institute) in Flint, Michigan. Dr. Zgorzelski's primary fields of interest in research, teaching and consulting are System Analysis, and Manufacturing Management. He teaches a variety of undergraduate and graduate courses in Engineering and in Manufacturing Management. He is an author of over sixty publications. Dr. Zgorzelski has been for many years a member of the International Federation for Information Processing (IFIP) Working Group 5.2, and was one of the founding members of Eurographics.

Paulina Zeno graduated from the University of Michigan in Computer Science in 1992. Following that she worked as a Systems Engineer for EDS at the Cadillac Assembly Plant in Detroit. Currently she is a graduate student at MIT Sloan School of Management. She is the daughter of Dr. Maciej Zgorzelski