From Analysis to Code Generation: Experiences from an Information Engineering Project Using I-CASE Technology

Karl Kurbel

University of Muenster, Institute of Business Informatics,
Grevener Strasse 91, D-48167 Muenster, Germany

Abstract. Development of a large information system, following the Information Engineering approach by James Martin, is described. KnowledgeWare's Application Development Workbench (ADW) was used as I-CASE (Integrated CASE) environment. The 3½-person-year project went all the way from analysis to code generation. Within the project, 160,000 lines of code were successfully generated; the final system is expected to amount to 330,000 lines. The information system is probably one of the first ones of such size that was actually generated with ADW for an OS/2 target environment. The paper gives an outline of the project and reports on experiences with I-CASE technology. The project was part of advanced business-informatics education at Muenster university.

1 Computer-aided Information Engineering

In this paper, a medium-size project following James Martin's approach to Information Engineering (IE) is described. Both Information Engineering methodology and corresponding integrated CASE tools were applied. The goal of the project was to develop an information system (IS) that supports administrative work of a fairly large university institute.

Information Engineering as introduced by James Martin looks at the organization as a whole. According to Martin, Information Engineering stands for "the application of an interlocking set of formal techniques for the planning, analysis, design, and construction of information systems on an enterprise-wide basis or across a major sector of the enterprise" ([4], p. 1). The comprehensive Information Engineering view covers all stages of IS planning and development, starting from strategic planning down to technical construction of programs and data structures. It also means that the focus is not (only) on a particular information system, but on enterprise-wide information processing as a whole. Finally, separate views of information systems are integrated: data, functions, and processes are analyzed and modelled within a unique framework.

Information Engineering consists of four main stages: *Information Strategy Planning* is the top stage where strategic goals, critical success factors, and informa-

tion requirements of major parts of the enterprise are determined. The result of Information Strategy Planning is a global model of the enterprise and its division into business areas. On the second level, *Business Area Analysis* is performed within one or more major sectors of the enterprise. Data models (e.g. entity-relationship diagrams), process models (e.g. decomposition diagrams) and other models are developed, and desirable information systems within the business areas are defined. *System Design* is the third level where procedures, data structures, screen layouts, windows, reports, etc. are specified. On the fourth level called *Construction*, programs and data structures are implemented, tested, and integrated. Figure 1 shows a pyramid view of the stages as presented by James Martin.

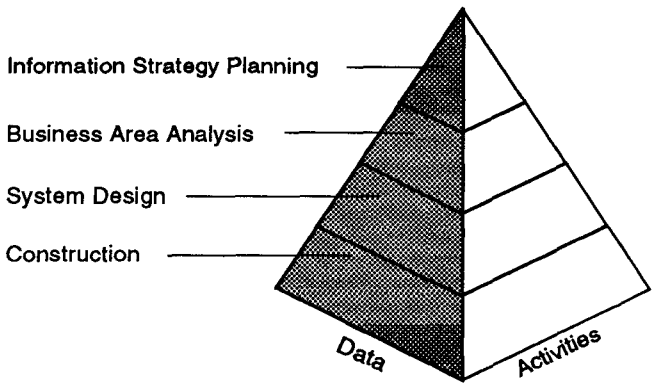

Fig. 1: Stages of Information Engineering according to Martin [4]

Information Engineering requires integrated CASE support for all stages. One of the objectives is to generate code automatically. Tool integration is to be achieved by a common repository, the so-called *encyclopedia*. All information collected during the stages of Information Engineering is transformed into a common representation format, and stored in the encyclopedia.

Although there is quite a number of integrated CASE (I-CASE) tools (see [12] for a survey, for example), only two of them support the comprehensive Information Engineering approach so far: ADW (Application Development Workbench) by KnowledgeWare [2] and IEF (Information Engineering Facility) by TI Information Engineering [13]. Both have been used to develop mainframe-oriented applications in practice [8, 10, 12]. However, little has been reported on workstation LANs as target environments yet. In this paper, we set the focus on the workstation level and describe our experiences with ADW in such an environment.

2 What are the "Business Areas" of a University Institute?

The Information Engineering approach is tailored to meet the requirements of enterprises and does not lend itself naturally to bureaucratic organizations like German

universities. Information strategy planning with regard to strategic goals and critical success factors would certainly have been a challenging task, but it was beyond the scope of our project. Therefore, only some global relations were considered and modelled, and the business areas to be analyzed later were defined during this stage.

At first glance, "teaching", "research", and "administration" might be considered "business areas" of a university institute. A closer look reveals, however, that only formal aspects of research lend themselves to analysis and modelling, whereas administration is a very complex field that needs to be split up in several business areas.

The Institute of Business Informatics at Muenster university in Germany for which the information system was developed employs about 70 people (including professors, scientific and non-scientific personnel as well as some 30 student assistants). Large portions of administrative tasks are not performed by the central university administration but have been delegated. The institute, however, has no comparable administrative machinery. This means that scientific and technical staff have to spend significant portions of their time for activities like buying equipment, updating inventory lists, paying bills, accounting, budgeting, keeping leave books, and so on. In 1992, for example, about 3,300 order positions had to be booked and 1,840 invoices had to be paid.

Effective support for administrative tasks was therefore urgently required. The problem of finding the "right" tasks to attack was solved in a straightforward manner, as the institute director was involved in the project as supervisor. In pre-project planning, three "business areas" emerged:

Budgeting and Purchasing

Administration of financial means on the one hand, and preparations of purchases (hardware, software, furniture, books, etc.) on the other hand are the most time-consuming tasks, involving about 15 people throughout the institute. Different procedures apply, depending on where the funds come from (state, foundations, enterprises, etc.), whether they are assigned to the institute as a whole or to individual professors, and what their intended appropriations are. Accounts correspond to those determinants. Purchases are closely related with accounts. For example, reservations have to be made when procurement orders are placed; they have to be confirmed when goods are delivered and finally booked when invoices are paid. Therefore budgeting and purchasing are based on the same data model.

Resources Management

Resources to be administered are chairs, projects, persons, posts, hardware, software, rooms, keys, etc. A large number of connections between them - some of them rather sophisticated - have to be considered. For example, hardware configurations - which monitors, boards, disks, etc. belong to which computers? - are treated as relationships between hardware components. Some procedures depending on re-

sources data are: updating inventory lists, leave books, and lists of official tours, assigning office rooms to persons and lecture rooms to courses or other events, etc.

Teaching

"Teaching" as a business area does not refer to contents of courses, but to operational and administrative activities: Announcements of lectures, catalogs of lectures, tables of contents, handling of admission requirements, textbook lists; distribution, collection, and marking of exercises, grades, certificates, etc. Exercises including computer-work may require assigning students to computer pools, workstations, and times.

3 Information Engineering Environment

As to computer support, KnowledgeWare's I-CASE environment ADW (Application Development Workbench), version 1.6, was employed. ADW is the OS/2 version and successor of IEW (Information Engineering Workbench). The tools of ADW are grouped into four categories corresponding to the four stages of Information Engineering:

Planning Workstation
Analysis Workstation
Design Workstation
Construction Workstation

These workstations run under OS/2, whereas target environments are primarily IBM mainframes under MVS. Since 1992, OS/2-based PCs as target environment are also supported. Major tools of ADW are:

Decomposition Diagrammer (for hierarchical structures)
Entity Relationsship Diagrammer (for data modelling)
Data Flow Diagrammer (for specification of data flows)
Association Matrix Diagrammer (to represent relationships between encyclopedia objects)
Minispec Action Diagrammer (to describe procedural logic)
GUI Layout Diagrammer (for interface design and generation of Cobol source code)
Structure Chart Diagrammer (to define module hierarchies)
Module Action Diagrammer (for detail specification of procedural logic)
Relational Database Diagrammer (for database design)
Data Structure Diagrammer (to specify data structures, records, and relations)
GUI Code Generator (to specify database access and generate Cobol source code)

These tools are highly interconnected as shown in figure 2. Unfortunately, I-CASE means here also that *all* the tools have to be employed, and that the user has to know

all the connections and dependencies depicted in the figure. Tools are integrated by means of an *encyclopedia*. Objects of the information model that were generated and stored in the encyclopedia by one tool can be read by other tools. Sometimes different tools may be employed to generate objects of a certain type. Figure 2 indicates this kind of relation by double-headed arrows. In two cases, integration has to be achieved by auxiliary functions (broken lines). For example, the Relational Database Diagrammer does not process objects of an entity-relationship diagram directly, but requires transformation into a so-called "first-cut model" by means of a generating function first.

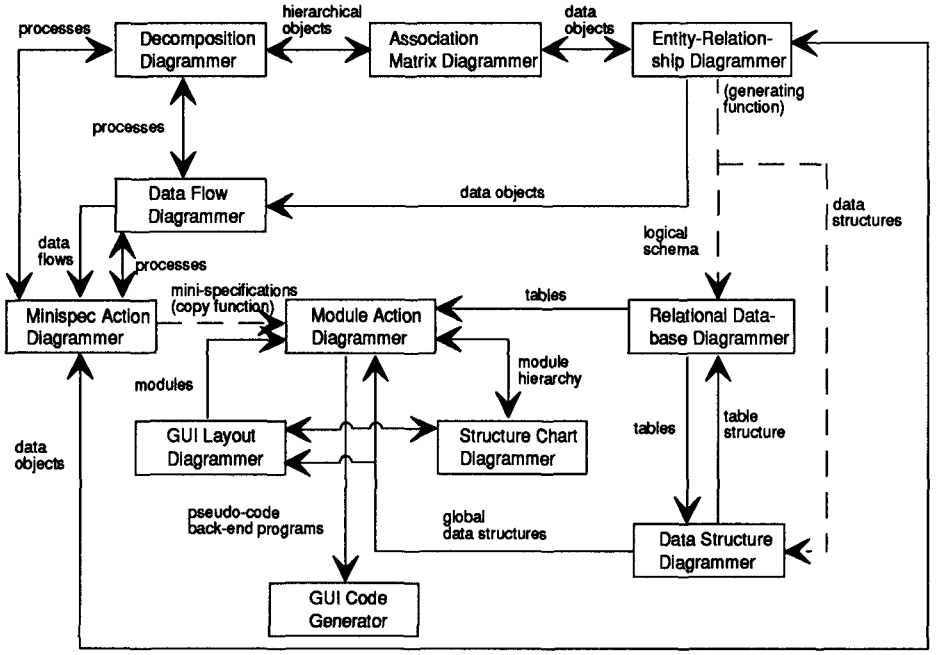

Fig. 2: Connections between ADW tools

The hardware and software environment for the project was a Novell network where ten PCs (80486, 33 MHz) had been additionally equipped with OS/2. Those PCs could be run both under MS-Windows/DOS and OS/2. Some of the tools had to be installed locally, whereas others could remain on the server.

Unfortunately, ADW does not have a LAN encyclopedia yet. As 30 people had to work simultaneously, a master encyclopedia on one computer and nine additional working encyclopedias (three per business area) on the other computers were created. They were consolidated at regular intervals. Between consolidation runs, read-only copies of the master encyclopedia and of the working encyclopedias were given to other project subteams.

User interface specifications generated by ADW were translated by the OS/2 Resource Compiler. It is part of the Developer's Toolkit for OS/2. For the other com-

ponents generated by ADW (procedural logic, database access), Micro Focus Cobol Compiler (version 3.0) had to be employed. It contains a precompiler for embedded SQL generated by ADW for relational data manipulation. The target database system was the Database Manager that is part of IBM's Extended Services for OS/2. The data definition statements generated by ADW could be used directly as input for the Database Manager.

4 Project Stages

4.1 Business Area Analysis

The three business areas outlined in chapter 2 were analyzed with the help of the Analysis Workstation tools. Results were hierarchies of functions and processes on the activities' side, and entity-relationship diagrams on the data side. Data flow diagrams were used to describe data flows between processes, transformation of data by processes, access to databases, and communication between processes and the environment. Coarse procedural logic in elementary processes was outlined in so-called "mini-specifications".

4.2 System Design

User-interface design was the main concern during this stage. Elementary processes and their respective input/output interfaces had been specified during business area analysis. Now the windows of the information system were designed, and those interfaces were mapped to the windows. On the data side, a so-called first-cut database schema was generated from the data model. Following this intermediate step, attribute hierarchies and ranges from analysis were available to the design workstation tools. They only needed to be connected to the windows where respective attribute informations were required. In order to make windows appear in a unique way, guidelines developed during analysis had to be observed by all team members. Layouts of reports and forms (e.g. application for leave or official tour) were also specified.

A very important part of this stage was to design a multi-level system of access rights. Particular consideration had to be given to the fact that users are professors, scientific and technical staff, and students. On the other hand, there are user groups such as staff of a particular chair, students taking the same course, etc.; and finally, there are users that have specific tasks (such as administration of funds). Access to certain data may thus depend on several factors and furthermore, on the mode of access (create, read, write, modify). For example, read access to a student's grade is granted to professors, perhaps to other scientific personnel, and to the student himself but not to other students, whereas write access may be restricted to professors. In order to satisfy all these requirements, a rather sophisticated system of access rights had to be designed. Its basic idea is that *roles* can be defined and rights can be associated with these roles. There is a number of default roles, but new roles may

also be introduced. One or more roles may be assigned to each user. Access rights are checked whenever the respective windows are to be activated or the respective data are to be addressed.

4.3 Construction

During the construction stage, procedural logic, connections between procedural logic and user interface, and database accesses were developed.

First, the logic of elementary processes was specified in detail. According to Martin's Information Engineering approach, this task actually belongs to the design stage. For several reasons including poor tool support, it was postponed until construction. The tool for detailed specification of logic is the *Module Action Diagrammer*. The language to be used on this level is *Enriched Cobol*. It contains Cobol elements as well as constructs supporting communication with the window interfaces (e.g. "get from window") and data manipulation. Embedded SQL statements lying behind those data manipulation commands may be modified by the developers.

Next, connections between procedural logic and user interface were established. The tool supporting this step is the *GUI Layout Diagrammer*. The hierarchy of modules was examined with the help of the *Structure Chart Diagrammer*. It depicts the hierarchy of calls in a graphical manner. Nodes of the calling tree are elementary processes.

Rather awkward is ADW's distinction between GUI and non-GUI programs (GUI = graphical user interface). *GUI programs* are programs containing only procedural logic, windows, and connections between those components. They determine primarily when and which windows have to be called, and how data read from the windows are to be processed. *Non-GUI programs* are programs that contain not only procedural logic but also access to files or databases. They are treated in a different manner and are much more awkward to create than GUI programs (see section 6.1).

GUI programs were developed first, by adding Enriched Cobol procedural logic to the window specifications. From the program source texts ("module actions"), the *GUI Layout Diagrammer* generates so-called "front-end programs" in Micro Focus Cobol code. The front-end programs were then compiled and tested with respect to user input. Afterwards, the non-GUI "back-end programs" were developed. They were generated by means of the *GUI Code Generator*. The necessary databases and relations were created by the *OS/2 Database Manager*.

Since reports based on user data are not supported by ADW, separate Cobol programs had to be written. The *Micro Focus Workbench* was used for this purpose. At least, Cobol programs could be called directly from inside the programs generated by ADW. Necessary call statements did not have to be put into the generated Cobol code but could be inserted into the module actions.

5 Project Conditions and Results

The information system was developed within a project that was part of advanced business-informatics education at the University of Muenster, Germany. Projects are part of the curriculum, summing up and integrating experience from other courses such as information-system development, software engineering, data modelling and database management systems. Participants were at the end of their eight semester studies. They had worked before with tools supporting the above fields; in particular, they had gained some experience solving "small" problems with ADW during a one-semester course. What was still missing was substantial experience with cooperative project work, subject to activity schedules, milestones, and delivery dates.

The project team comprised 30 people altogether, not counting the users involved into JRP and JAD sessions [3] nor technical staff (network, OS/2 administration, etc.). Since the project was embedded in a semester curriculum, its duration, beginning, and end were predetermined: The project had to be completed within a period of exactly three months. Project conditions thus were rather untypical. Whereas "ordinary" projects with comparable output might be executed by two or three persons over a period of one or two years, here a fairly *large number* of people had to be coordinated for a rather *short time*. Project planning and management had to take these circumstances into account; for example, project management was more rigorous and stricter than in normal projects. Both the project manager and the project supervisor had successfully completed projects of that type before. In a report on one of them, the expression "million-monkey approach" was used (by others) as a description [7].

	Project preparation and strategy planning	Project execution	Total hours
Project management and supervision	150 h	600 h	750 h
Technical administration	100 h	700 h	800 h
Students		4,500 h	4,500 h
Total hours	250 h	5,800 h	6,050 h

Fig. 3: Development effort

Development effort amounted to 38 person months in total. Figure 3 shows how it is distributed among management, students, and major phases. Management and administration were time-consuming (1,550 hours), partly because of inherent complexity of novel I-CASE technology, partly because of problems with the toolset.

Results of the project were not only programs and databases, but also a number of different models of the business areas. They are stored in the encyclopedia which

had reached a size of 14 MB at the end. The overall system consists of 17 subsystems - the "front ends". According to Information Engineering philosophy, we did not attempt to complete all possible subsystems identified in business area analysis in one run. Some of them will be treated later. Some went all the way to construction but could not be finished within the given project duration. Summing up the programs that were truly generated and tested, the code amounts to about 160,000 lines (excluding separate Cobol programs for reports). When the missing back-end programs will be completed, the total system will comprise some 330,000 lines of code. Figure 4 summarizes quantitative project results.

Data		Activities		Program components	
Entity types	73	Functions	64	Front-end programs	17
Relationship types	119	Processes	242	Windows	102
		Elementary processes	382	Back-end programs	116
				Separate Cobol programs	20
				Generated lines of code	116,000 (330,000)

Fig. 4: Quantitative project results

6 Observations and Experiences from the Project

6.1 Information Engineering Workstation Tools

The Information Engineering approach to IS development places individual information systems into an organization-wide context based on common data, function, and process models. Integration is not only conceptual, but it is also supported by interlocking tools. This means, for example, that consistency of different models can be checked by these tools. In fact, it would have hardly been possible to validate 73 entity types and 119 relationship types of the data model without tool support.

ADW's tools for the early stages - Planning, Analysis, and Design Workstation - proved to be efficient and well integrated. Our experience from this particular project primarily refers to the latter two ones, because the Planning Workstation was not employed by the students. Whenever objects were created or modified by one tool, the respective information was immediately available to other tools. Handling of the tools is mostly simple, intuitive, and easy to learn. However, this does not hold for the GUI Code Generator (see below).

Vertical tool integration across stages is satisfactory with regard to the data side. Components of the data model defined during planning or analysis can be processed directly in the design stage. They may be further transferred to construction where the relational model is generated. On the activities side, top-down refinement from business functions to processes and further on to elementary processes is intuitive

and easy to carry out. However, vertical integration is less satisfactory, going only until design (see below).

Some of the tools enhance productivity significantly. By means of the GUI Layout Diagrammer, for example, it was possible to define and validate all 102 windows within one week. Since model information stored in the central encyclopedia can be used by any tool, no additional recording is necessary. For the same reason, a good deal of documentation can be derived automatically (e.g. entity-relationship diagrams, call hierarchies).

On the other hand, the list of *drawbacks* is rather long. Many of them are due to the fact that the tools (under OS/2) have not completely matured yet. Some of them contain severe errors whereas others still suffer from their mainframe origins.

The artificial distinction between GUI and non-GUI programs is particularly awkward, as these two types of programs have to be developed in completely different ways. The separation of GUI and non-GUI components is maintained all the way down to executable programs. Figure 5 illustrates how different tools have to be employed. Whereas GUI programs can be specified and generated very efficiently with the help of the GUI Layout Diagrammer, the way on the right hand side is extremely ponderous.

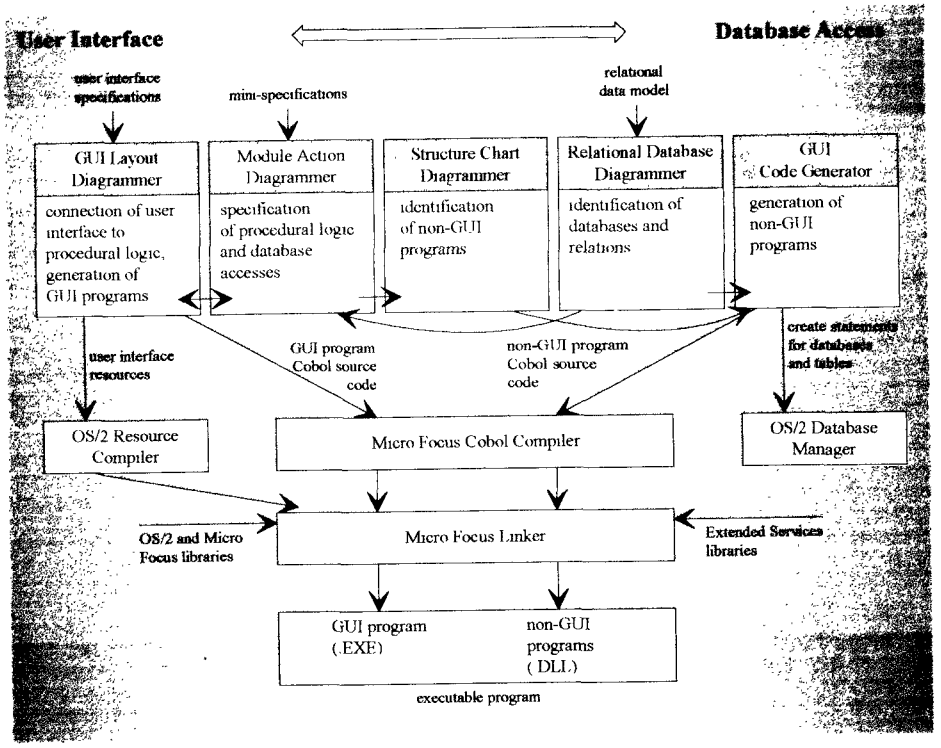

Fig. 5: Relations between construction tools

Response times are sometimes very long. The process of building up the screen took up to two minutes when a group of windows had to be loaded. *Tool documentation* proved to be incomplete and full of errors. This was particularly hampering to project progress as lengthy trial-and-error processes were necessary to find out what was wrong, what was right, and what was missing in the documentation. For example, there is no coherent description of which specifications are needed and how to proceed to generate Cobol programs. Motivation of the project team was severely damaged by the effects of documentation flaws.

As to vertical integration of *procedural-logic* components, there is a complete break between analysis and design/construction. Mini-specifications from business area analysis cannot be processed in the design stage. The only way to make use of those earlier descriptions is to copy them into program specifications where they may serve as comments. Procedural logic has to be redeveloped completely. Since the language is Enriched Cobol, the level of expression is only slightly above "ordinary" 3GL programming! Another drawback is that error messages refer to the code generated by ADW. Debugging becomes rather awkward as developers have to examine code they did not write! Unfortunately, no debugger on the specification level is available (yet).

6.2 Encyclopedia

Quite a number of ADW's shortcomings are related to the current state of the encyclopedia. The encyclopedia is basically a single-user encyclopedia and, at most, suitable for very small development teams. A LAN-based encyclopedia as needed in a 30-people project is not available. Instead, ADW allows several parallel encyclopedias to be kept and consolidated from time to time. This is a rather insufficient substitute, however. Consolidation runs take long; in our case (9 encyclopedias) they amounted to ½ - 1 day during which encyclopedias could not be used.

Some *consistency checks* are made during consolidation, but often the user has to ensure consistency himself. For example, the master encyclopedia will not notice that an element has been deleted in one of the decentral encyclopedias. Consistency may also become a problem between tools when several persons work with the same objects. Consider, for example, the case that one person deletes - by means of the Data Flow Diagrammer - a process that has subprocesses specified by other persons. The Decomposition Diagrammer now is no longer able to associate the subprocesses correctly unless they are reassigned by hand. When several people are involved, problems may arise if information about the deletion is not passed to all of them in a coordinated way.

A severe setback occurred once when storage for the encyclopedia was used up. Some members of the project team had worked until late. Before going home, they proceeded as usual to store their results. Frustration was big the next morning when they discovered that yesterday's work was not there any more. The reason was that ADW does not issue a message when there is not enough storage left; it simply does not store! To avoid this kind of problem, oversize extra storage had to be provided

from then on, considering that the encyclopedias grew at a rate of 10 MB per day during that particular phase of the project.

KnowledgeWare meanwhile seems to have recognized that many ADW problems are due to the weak encyclopedia. The announcement was made that the *Rochade* repository will be supported in the future, too.

6.3 Information Engineering Methodology

To some extent, Information Engineering methodology as proposed by Martin was applied in the project. End users were involved at several stages. In particular, requirements were analyzed and specified with the help of end users, and prototyping was applied to demonstrate and revise intermediate results.

During analysis, JRP (joint requirements planning) workshops were conducted for each subarea, including both end users, designers responsible for that subarea, and the subproject leader. Some JAD (joint application design) workshops were also scheduled. For several reasons, however, workshops were not as elaborate as suggested in IE publications [3, 6]. First, sufficient experience with JRP and JAD methodology was lacking. Second, time for the project was extremely limited. Third, the persons heading the project (project/subproject managers, supervisor) knew the business areas very well themselves. Thus, JRP and JAD were not so much conducted in the form of workshops, but resembled more ordinary requirements analysis with some end-user prototyping.

Timebox methodology was not applied explicitly ([6], p. 170). However, in all phases of the project, functions and processes to be analyzed, designed, constructed, or left out, respectively, were prioritized. In this way, the basic idea of timeboxes underlay the whole project.

Consistency problems arose whenever models from design or analysis needed to be changed. Modifications at a later stage were made within the specific forms of representation of that stage (e.g. relational data structures). Models of former stages (e.g. entity-relationship model) were neither adapted automatically, by ADW, nor by the developers, because of lack of time. From this, it was inevitable that inconsistencies among analysis models, design models, data structures, and programs grew constantly.

7 Outlook

According to Ed Yourdon, it takes some 10 - 15 years for new technologies to reach widespread use ([14], p. 268). Today, dissemination of Information Engineering-based I-CASE is still at its beginning. One reason is certainly that the rather sophisticated Information Engineering approach to IS development will only work if developers have received adequate education in analyzing and modelling. In particular, they need the capabability to develop *models* of the problem domain, rather than write specifications and programs as taught in software engineering.

Another reason seems to be that tools have not reached the stability needed for industry-scale application yet; this was one of the experiences from our project. Furthermore, the code generated automatically is often considered inefficient. Therefore, some users employ ADW or IEF for analysis and design only, but leave construction to their experienced Cobol programmers. Some use less comprehensive "lower CASE" tools that generate more efficient code. Better code generation following the modelling phases in a natural way are indispensable for truly *integrated* CASE.

At present, only "typical" data processing problems are supported, i.e. transaction-oriented problems where input/output by way of windows and forms, and accesses to databases dominate. Other problem types, e.g. problems including complex algorithms or active graphics, are still beyond the scope of I-CASE. Following Ed Yourdon, some 5 - 10 % of business information-processing problems can be tackled today, but 90 % might be reached by the end of the decade ([14], p. 273).

Finally, many potential users are still uncertain about the cost and benefit of I-CASE. Although tool vendors have been promising significant gains in productivity, objective investigations are still rare. In an article of May 1993 [10], three enterprises reported on their I-CASE activities. Two of them actually tried to measure costs and benefits. One company found that productivity had raised substantially. The second one recognized only moderate savings. The third company had not quantified expectations and did not perform measurements. They felt that their vague hopes were not fulfilled and hence cancelled further CASE activities.

References

[1] Ernst & Young GmbH: Application Development Workbench; Ernst & Young CASE Services GmbH & Co.; Stuttgart o.J.

[2] KnowledgeWare, Inc.: Application Development Workbench/Workstation Basics, Release 1.6.02; KnowledgeWare, Inc., Atlanta, GA 1991.

[3] Lucas, M.A.: The Way of JAD; Database Programming and Design 6 (1993) 7, pp. 42-49.

[4] Martin, J.: Information Engineering, Book I, Introduction; Englewood Cliffs, NJ 1989.

[5] Martin, J.: Information Engineering, Book II, Planning and Analysis; Englewood Cliffs, NJ 1990.

[6] Martin, J.: Information Engineering, Book III, Design and Construction; Englewood Cliffs, NJ 1990.

[7] N.N.: Die Kunst der gebändigten Unordnung; Computerworld Schweiz (1987) 12, pp. 7-11.

[8] N.N.: Modellgetriebene Anwendungsentwicklung; IBM Nachrichten 41 (1991) 306, pp. 46-49.

[10] N.N.: The Costs and Benefits of CASE; I/S analyzer 31 (1993) 6.

[11] Short, K., Dodd, J.: Information Engineering \with Objects, Issue 1.2; Texas Instruments Technical Paper; JMA Information Engineering Ltd., Ashford, Middlesex (England) 1992.

[12] Snell, N.: Users Rethink Life Cycle CASE; Datamation (1993) May 15, pp. 102-105.

[13] Texas Instruments Inc. (Eds.): IEF Information Engineering Facility, Technology Overview, Second Edition; TI Part Number 2739900-8027; Plano, Texas, November 1990.

[14] Yourdon, E.: Decline and Fall of the American Programmer; Englewood Cliffs, NJ 1992.