ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S15CSE01

CASE in CERN's Accelerator Sector

A. Albrecht (SL), A. Cabas-Alonso (AT), F. Chevrier (ECP), A. Daneels (AT), Ch. Delamare (CN), G. Ferran (CN), S. Foffano (MT), P. Heymans (AT), J.P. Matheys (ECP), D. Manglunki (PS), Y. Marti (SL), D. Missiaen (AT), G. Moorhead (CN), O. Novakov (AT), T. Pettersson (PS), J. Poole (SL), M. Pozzato (SL), J.P. Quesnel (AT), S. Santiago (CN), J. Schinzel (AT), N. Segura-Chinchilla (AT), C-H. Sicard (PS)
Presented by A. Daneels (AT)
CERN, European Organization for Nuclear Research, 1211 Geneva 23, Switzerland

Abstract

As in the software industry where computer aided software engineering (CASE) methodologies and tools are commonly used, CERN endeavours to introduce this technology to improve the efficiency of designing, producing and maintaining software. A large project is currently under development in the administrative area whereas a dedicated group has been set up to evaluate state of the art techniques for software development relating to physics experiments. A similar activity, though on a smaller scale, has been initiated in the accelerator sector also in view of the large amount of software that will be required by the LEP200 and the LHC projects. This paper briefly describes this technology and gives an account of current experience with the use of CASE methods and tools for technical projects in the accelerator sector at CERN.

1. INTRODUCTION

Software engineering is the application of techniques which lead to the implementation of better quality software. It implies a planned process of producing well-structured, reliable, good quality, maintainable software systems which corresponds to the users' needs, within reasonable time frames [1].

This definition suggests that software engineering includes a good deal more than just producing computer programs and that good software development includes documentation, databases, operational procedures, etc. Furthermore, it focusses the planned aspect of the process: as any other engineering discipline, software production should be properly managed with scope definition, specification analysis, cost estimation, production plans, role distribution, etc.

According to industry statistics, 75% of custom software development projects are rejected because they came either too late to be useful or did not correspond to the users' needs. However the complexity of application software grows continuously and today's average business package takes 32,000 man-days, i.e. 160 man-years to develop [2]. This is not dissimilar to the effort spent at CERN on application software for accelerators. Indeed, in the eighties at least 500 man-years were invested on controls and database applications for the PS accelerator complex, SPS and LEP, not including numerous developments that have not been accounted for. The annual maintenance effort is estimated to be around 15% of the development effort and exceeds the production capacity of the groups in charge. Here, maintenance is defined as software repair and update resulting from a changed functional specification of the software product. For each new development the volume of the software increases because more sophistication is required. By the time the LHC is approved, the demand for application software may well be two to three times higher because of increased functionality, increased information volume, more severe execution time constraints due to the superconducting nature of the machine, and higher reliability [3]. Even if the groups in charge manage to develop such large packages with the help of professional, voluntary and temporary staff, they will only be able to maintain it if software of sufficiently good quality is produced so as to dramatically reduce its maintenance cost.

2. WHAT IS CASE ?

CASE stands for Computer Assisted Software Engineering.

For each new software project the engineer is recurrently performing a number of similar activities: collecting information from his client, organizing that information, cross checking with the client, etc. The process is systematic, iterative and proceeds in increasing degree of detail. A number of methods and procedures could be derived which, because of their recurrence, would be more efficiently executed if assisted by computer programs. Software tools were thus developed to assist the software engineer in collecting, organizing, storing, retrieving and cross checking that information throughout the development process of his project. The information is introduced through graphical and alphanumerical user interfaces and recorded in a repository, possibly a database management system, so that it is available throughout the production life cycle for complementing, checking and various administrative operations.

CASE is introduced in order to encourage better quality designs, to increase productivity and to render software projects more manageable. Better quality software leads to reduced maintenance: industrial companies e.g. BBC, now ABB, claim to be able to reduce the maintenance to 2% of the development cost by using such tools [4].

3. CASE Products on the Market

There exist a large variety of CASE products on today's market: Computerworld [5] published a list of not less than 33 such products that were selected on the basis of the tools they provide for analysis, design, code generation, debugging, code / configuration management, testing and integration, code analysis, maintenance, documentation, reverse engineering, project management, etc. Unfortunately, no tools exist yet that provide all these facilities in a single environment, and as they are developed by different vendors, they are often not compatible: e.g. the output of a first vendor's analysis and design tool may not be usable by another vendor's code generator. Therefore some of the larger vendors endeavour to design integrating frameworks through which tools from multiple vendors can communicate. Such frameworks are called Integrated Project Support Environments, IPSE. However, even IPSE are only effective provided the IPSE producer has established contracts with third party CASE developers.

To date the application software market seems to be subdivided in two major domains: database and real time applications. Whereas database applications focus the structuring of information so that it can be easily retrieved and processed in a variety of associations or combinations as are useful to a business, the real time applications, in particular the control applications, concentrate on the functional and temporal aspects of the process: what functions have to be performed - and when - to respond to the process' behaviour. This difference is reflected in the facilities that are provided by current CASE tools: those for database applications provide extensive facilities to model the data, whereas the tools for real time applications are more concerned with modelling the functional and temporal behaviour of the process. One would expect tools for both database and real time applications in future to converge towards a single environment.

4. CASE Methods

CASE methods cover two aspects: a technical one that is concerned with the analysis, design and implementation of the application, and a managerial one that provides various project control and quality assessment techniques with rules for a proper organization of the project team where every member has a well defined role, in addition to recipes for the organization of reviews, feedback sessions with the users, quality assurance sessions, etc.

As an initial step, most CASE methods start with defining the objectives and the boundaries of the project. Next they enter into a strategy phase where a model is produced of what needs to be developed and describing its relation to the environment. This model is then worked out in more detail during the analysis. Until this stage there is no consideration of how the requirements will be fulfilled, this is the purpose of the design. Next comes a build phase followed by implementation and commissioning. It should be noted that these methods bear strong resemblance with those developed in other engineering disciplines.

5. CASE Tools

CASE tools help the developer in reaching a full understanding of his project. They guide him through the various phases of the development process so as to produce a series of models to entirely depict his program: they are the "blue prints" of the program he will implement.

Software involves in general three basic components: data, processes that operate on data, and the time or events at which the processes are executed. CASE tools allow to model respectively the structure and the internal relations of the data, the functions and the events; they further depict how data relate to functions, functions to events, etc.

The models are represented by diagrams constructed by means of a number of generally accepted graphical notations: e.g. Yourdon / DeMarco for control flow and state transition, Jackson for data structure, Chen, ERD (entity relationship diagram) or NIAM (Natural Information Analysis Method) for entity relationships, Stevens, Myers and Constantine for structure charts. CASE diagrams thus appear to the reader as roadmaps through which he can navigate to understand the structure of the program and the functionality it provides. These diagrams are complemented by structured textual description of e.g. entities, attributes, processes, tasks, events, etc. and free format specification. Textual specification are also entered through appropriate editors. The more advanced tools further provide extensive checking such as referencibility of entities, completeness and consistency of the diagrams, identifying e.g. unlabeled or unbalanced flows, entities and attributes , etc.

Most CASE tools support multi-user development and run in a distributed environment. Their user interfaces are based on graphical and textual editors with Macintosh or MOTIF style interaction, through which the programmer enter its specifications that are placed in a central repository. The repository integrates the various tools into a single environment (hence i-CASE, i.e. integrated CASE); data are available at each phase of the project life cycle and can be checked against the functionality for completeness and consistency. Project administration tools are also provided to define access rights to the repository for each member of the project team depending on his privileges, to control the versions of diagrams, etc..

6. Pilot Projects and CASE Tools Used in the Accelerator Sector

In order to evaluate the applicability of CASE in the accelerator sector at CERN, a number of pilot projects have been selected in various fields relating to accelerators: controls, data acquisition, cryogenics, modelling, radiation monitoring and survey.

The pilot projects for database applications concern new designs, retrofitting existing systems or modelling for documentation purposes. ORACLE*CASE was selected because it provides a full information system engineering environment based on the ORACLE relational database that is standard at CERN. However, in order to compare the ORACLE methodology to model entities with NIAM, a "shadow" exercise has been initiated using RIDL, Relational IDea Laboratory, of Intellibase NV (Antwerp, Belgium), that is based on the latter methodology.

The evaluation of CASE for real time applications has focused on StP, Software through Pictures produced by Interactive Development Environments, IDE (San Francisco, USA) and Real Time Engineering Environment, RTEE, of Westmount Technologies (Delft, the Netherlands). These CASE tools complement Teamwork of CADRE Technologies that is in use in SL division[1].

Brief Description of Pilot Projects

- **Data base Applications**

The database for cryogenics should provide a full inventory of the equipment and information on the state. It should also provide information concerning control signals and algorithms and incorporate archive measurement and test data of the various instruments for maintenance purposes.

The Survey group is responsible for the proper alignment of approximately 10,000 accelerator and transfer line elements over approximately 60 km. Most of that information is stored in a central ORACLE database and should be extended to include data about the stability of the measurement devices.

The SL database aims at providing a central description of the SPS and LEP accelerators, reference information on the state of the machine and at maintaining historical reference data for a variety of applications in different areas: controls, vacuum, survey, beam instrumentation, magnets, power converters, radio frequency, mechanical design, and accelerator physics.

- **Real time applications.**

The LEAR control system is currently running in a VAX/VMS environment with UIS as console interface software. The front end hardware is based on PDP with Pascal software. By the accelerator start-up in 1992, the front-end part of this control system should be adapted to an X-system environment and its database migrated into an ORACLE one. It was intended first to model the control system by the use of ORACLE*CASE for documentation purposes, and next, to design an enhanced version.

The General Supervisory System monitors the environmental conditions in the four experimental sites of LEP through more than 1,000 detectors. The system is based on an expert system with its knowledge base and security rules stored in an ORACLE database. It grew as experience accumulated over the years and has now reached a point were a major overhaul is needed to rationalise and enhance its functionality. The need to port the system to a UNIX environment provides the opportunity to undertake its retrofit.

For StP two parallel evaluations were carried out. The first one concerned an asynchronous data-collector service that will be incorporated within the control system of the PS accelerator complex. That system has been analysed with the aid of data structure, data flow and state transitions editors. The second evaluation concerned the use of structure chart editing facilities for the detailed design of an error logging program.

In contrast to this, the evaluation of the RTEE for real time application was not based on a real life project. Instead, the idea was to analyse the facilities the tools provide, in some cases by modelling parts of existing real time programs, and by evaluating the method and notation on which they are based.

Objectives

The prime objectives of the pilot projects are to gain experience with CASE methodology and tools and to evaluate their applicability to technical projects. All projects, except the one using RTEE, aim at producing real applications. This was in particular a pre-requisite for being entitled to an ORACLE*CASE licence.

From a management's point of view, CASE is evaluated as a way to produce better quality software and to enhance communication with the user, cooperation across projects and progress visibility. It is also aimed at economy of scale by sharing the resources invested in the evaluation exercises.

Constraints

The mixed nature of the projects reflects the constraints.

The pilot projects have limited financial and human resources: even if the overall team involves around 25 persons, depending on their role some are only spending a fraction of their time in the project. In addition the teams belong to different Divisions: Accelerator Technology, Computers and Networks, Electronics and Computing for Physics, Mechanical Technology, Proton Synchrotron, SPS-LEP. They are thus geographically distributed and their members are also involved in other activities whose priorities depend on the local divisional objectives. The teams cover a wide range of disciplines, each having it own habits and jargon. However, this multidisciplinary, multidivisional and part time nature are typical for the way projects are often carried out at CERN.

[1] Note: Teamwork, that was recommended in 1989 by CERN's Technical Board for Process Controls and Accelerator Electronics (TEBOCO), became unusable for PS division when it was decided to standardize all CERN controls on UNIX or Ultrix. PS division was equipped with VAX stations and their VMS O/S was replaced by Ultrix: although operational on VAX-VMS or DEC-Ultrix platforms, Teamwork appeared not to run properly on PS' hybrid VAX-Ultrix configurations.

In addition the projects are different in size: though most are small and well encapsulated, the database project for cryogenics appears as a large one. It was thus difficult, not to say impossible, to keep the pilot projects synchronised so as to reach a more homogeneous level of expertise, possibly sharing some common designs, agreeing on a common glossary, etc.

The project involves a mix of platforms: VAX stations running VMS, DEC and SUN-SLC & IPC running Unix. The workstations involved in the evaluation of ORACLE*CASE host the tools and are linked by ethernet to a central VAX 6420 with VMS 5.4 that houses the central database and the dictionary. The workstations communicate with the central engineering database through SQL*NET (Fig.1.).

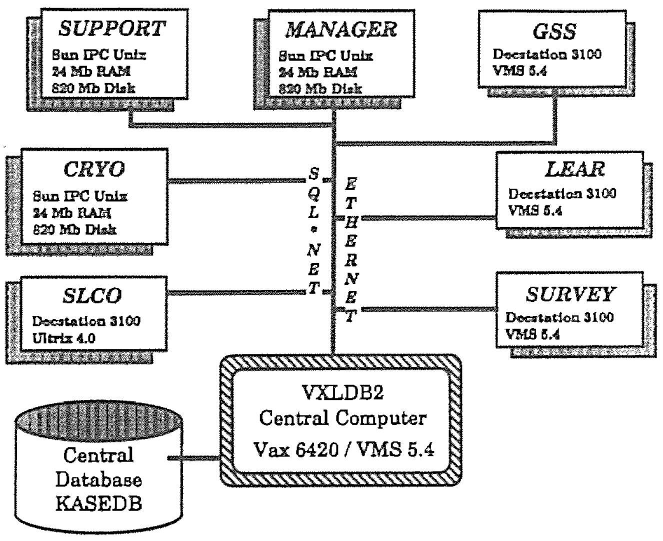

Figure 1

On ORACLE Corporation's request a tighter management scheme was adopted for the database projects than for the real time ones.

Evolution of the projects

It was felt that learning the method and the tools by reading manuals and playing with the tools alone would be too time consuming. Therefore systematic training on the method and the tools was introduced for the database pilot project teams and also for those involved in real time CASE evaluation who wished so. Professional consultancy was called upon to define the scope of the projects, for formal reviews, feedback and quality assurance in order to rapidly reach a reasonable degree of expertise within the CERN software engineering team.

In general the methods were followed but needed to be adapted to the way of working, proper to scientific organizations. This is particularly true for ORACLE*CASE Method whose rigor was difficult to accept since it broke with the usual individual style of working. The methods for real time applications adapt easily to a more familiar bottom up design without excluding the rigid top down approach.

The Survey and SL database projects are currently in the build stage and are generating tables and forms. Those involved in retrofitting the database for survey claim that by using CASE they were able to optimize their design so as to improve the performances of data retrieval by a factor of 5 when compared to the original design.

The cryogenics database however grew to become a much larger project than anticipated. This is in part due to the fact that an overall picture was originally not available and only became apparent as one proceeded through the method. Much time was spend on understanding the cryogenics problems in addition to learning the method and the tools. However, at this time a model has been designed that is reasonably stable and has obtained agreement with the users.

LEAR failed to model its control system by using ORACLE*CASE. Despite hopes to model the system for documentation purposes, it turned out impossible to describe the time dependencies and the process sequencing that are fundamental aspects of real time systems.

Eventually modelling the General Supervisory System could not be pursued because of lack of manpower. However, it came to a point where similar problems as for LEAR became apparent because of its strong real time nature.

7. COMPARING METHODS AND TOOLS

*ORACLE*CASE*

ORACLE*CASE Method follows a top down approach whereby a number of tasks have to be performed in a specific sequence. These tasks are grouped in the various project phases which must yield well defined deliverables before one can proceed to the next phase. The method also includes many cross checking techniques to ensure the accuracy, consistency and completeness of the design. It also provides techniques and procedures for team and project management emphasising the user involvement throughout the life cycle and control techniques.

The following phases are identified:

Scoping to define the limits of the project, its objectives and constraints. ORACLE puts a lot of emphasis on this phase as a means to keep the project on its tracks.

Strategy: in this phase a model is produced of what needs to be developed. The functionality is represented by a hierarchical breakdown of the functions to be performed and a structural model is built with the entities. This phase also includes statements about quality standards to be achieved. The requirements are next translated into written specifications, drawings, data sheets, etc.

Analysis: the previous model is now worked out by adding attributes to the entities and by describing them in detail. The data flow diagram describes movement of data between the functions. Matrices allow the cross checking of completeness of the relations between functions and entities, entities and attributes.

Design: this phase concentrates on **how** the detailed requirements, as defined in the strategy, will be fulfilled. An information system architecture is produced identifying the various applications covered by the functions that will access the database together with a detailed program specification.

Build stage: the system is now built and reviewed with the user.

Transition phase: the implementer provides the customer with the necessary support to ensure a smooth transition from the old system to the new one.

Production (Operation): ensures the running of the system whilst its performances are being monitored.

It should also be stressed that ORACLE*CASE puts a lot of emphasis on the management aspect of the project and on the relation between the information system engineer and the customers.

The **tools** provided by ORACLE are CASE*Dictionary, an internal multi-user database that acts as the central repository for all information relating to a project through all stages of its life cycle. This dictionary is filled through diagrams (function hierarchy, entity relationship, data flow,...), matrices and interactive forms provided by CASE*Designer. Furthermore, CASE*Generator generates tables, menus, interactive forms using the 4 / 5 GL ORACLE tools, and reports. The central part in ORACLE*CASE is the entity relationship diagram, that is based on an extension of the Chen method. It provides a large number of facilities and coded symbols to describe the entities, their type and attributes as well as their relations.

CASE for Real Time Applications

The rigid top down approach that is followed by some CASE methods and tools has been the object of endless debate and has been abandonned by some of its major initiators in favour of a more pragmatic middle out approach (6) or an object oriented one. The tools that were evaluated within the scope of this exercise, StP and RTEE, follow this middle out approach and allow the designer to progress both towards higher levels of abstraction and lower levels of details in the course of his analysis and design. Except for this particular feature, the CASE method for real time applications follows essentially the same sequence of phases (see previous paragraph) by which a series of models are produced that are gradually refined and complemented so as to ultimately yield template code. However, the user involvement and team organization during the project life cycle is not defined to the same extent as by the ORACLE*CASE method.

The major steps through the method are:

Survey, evaluating the requests and the feasibility within budget and time constraints.

Analysis, defining the **environmental model** of the system (i.e. how does the system relate to its environment) and the **behavioural model** showing the processes inside the system.

Architecture, where **processes, tasks** and eventually **modules** are modelled. This is a not so common feature that is particularly useful in case of distributed multitasking systems.

The **strategy phase** of ORACLE*CASE appears to include the **survey** phase with some overlap in the **analysis** when defining the environmental model.

Contrary to ORACLE*CASE whose diagrams are limited to describing the functional hierarchy, relationships between entities and dataflows, the **tools** for real time application development provide diagrams to model several control aspects of the system.

The **control flow diagram** provides hierarchical models of the system's functionality showing data and control flow between processes, the processing of data and their control actions. It is based on a number of rules and symbols to represent data processes and control processes, types of flows (data flow, update flows, continuous data flow, control flow and continuous control flows), stores (data and control stores such as on/off information), and external processes with which the system communicates. The familiar **data flow diagram** one finds in ORACLE*CASE, appears as a specific view in the control flow diagram. It is complemented by a **list of events** (temporal, control or flow oriented) identifying the stimuli that occur in the external world and to which the system has to respond, and diagrams to illustrate how these events relate to the system.

The **data structure diagram** gives an abstract and static representation of the data: it shows how the data are structured hierarchically, not their relation. It allows specification of simple sequences, iterations and selections.

The **entity relationship diagram** represents the static relations between entities using the Chen modelling technique.

The **state transition diagram** describes the states of a system and the sequence of activities between the states (in our case, following the Yourdon method).

The **system architecture diagram** is an extension of the control flow diagram. It allows one to graphically assign and partition tasks to processors and incorporates symbols for processors, tasks, interrupt service routines, message queues, message boxes, event queues and event flags.

The **structure chart** depicts the functional breakdown of the system. A module is defined as a collection of program statements.

Ultimately the code is generated in three steps: generation of a **program design language, PDL,** from the structure chart; generation of **program code** from **PDL**; generation of data type declarations from data type diagrams.

8. EVALUATION CRITERIA

CASE products are evaluated relative to their architecture, the environment in which they run, the applications that are provided with the product, the tools themselves and, last but not least, a number of economical considerations relating to the vendor.

The architecture is expected to be "open" allowing the user to customize e.g. menus, commands, etc., to set defaults and to extend the tool by invoking other tools such as e.g. testers and simulators.

The environment in which the tools run should be based on standard operating systems, windowing system; they should also support heterogeneous networks.

The product should allow tracing the requirements throughout the various phases and provide facilities to produce standard documents that can be customized, etc.

The tools themselves are evaluated on the ergonomics of their editors and the techniques they support.

Selecting such a tool implies a strategic choice and one has to take into account vendors' "health" on the CASE market, the support he is able to provide, and the evolution of his product with regard to those of competitors.

9. PRELIMINARY CONCLUSIONS

Although it is premature at this stage of the project to draw definite conclusions, sufficient experience has been accumulated to be convinced of the usefulness of CASE methods. They are very rigorous and convey confidence in the quality of the product. The clients admitted, after initial skepticism, that the method provides a sound basis for discussion. Even if during the analysis some projects grew larger than anticipated and extended well beyond the part to be implemented, it shows that an overall picture has been obtained thanks to the method and that provision for coherent future extension will be built into the model. In addition the methods give good control on the project life cycle.

Despite a number of weaknesses and inconveniences that the vendors endeavour to correct, all products proved to be state of the art. They provide an excellent basis for communication: designers and users discuss over a full set of specifications both graphical and textual, with agreed definitions and terminology avoiding misunderstanding. CASE appears thus as a must for large projects, and prepares the ground for subcontracting. Though at present CASE may seem an overkill for smaller projects, it may be that as soon as the technology is well mastered it allows to produce this type of project in "no time".

No major problems were encountered when working in a mixed environment. Most tools run on workstations and access the central repository installed on a server through the network.

None of the tools for real time applications that were evaluated run with ORACLE as internal repository. This is a limitation in CERN's context where ORACLE is a standard, but no tool could as yet be found on the market that satisfies this requirement.

However, the use of CASE implies a real change of habit: it requires an analytic approach and a disciplined style of working that contrasts with the previous free style. Also it needs a relatively long learning curve: the longest probably for ORACLE*CASE that often requires a good insight of the method and its relation with the tools for efficient understanding; therefore training by the vendor is of great help. Nevertheless, even after training, one should be prepared to invest significant time in order to become fluent.

It was difficult to come to an agreement on the method amongst the participants. The old debate between top down and bottom up design became vivid again. The top down approach that is followed by ORACLE*CASE was difficult to grasp in particular by those who have a real time controls background. They felt more at home with the method and tools for real time application which tend to follow a more pragmatic middle out approach.

The entity relationship technique on which ORACLE*CASE is based led to some concern. NIAM on the other hand defines binary relationships between objects, and is felt by some as a more natural method for data modelling. It also provides diagrams that are more informative as they include explicit notations for e.g. role, subtypes and procedural constraints. A consequence of course is that NIAM diagrams are less surveyable than the ORACLE entity relationship ones. It is however possible to generate NIAM diagrams from ORACLE ones; the inverse operation is not possible because of loss of information.

The tools for real time application development highlighted the well known problem of transfer of information between the analysis and design phase. While it would seem that object oriented methods might provide an elegant solution to this problem, no tool has yet provided such a solution. The evolution of tools towards object orientation will thus need to be closely followed.

Because of the difference in version between ORACLE database and CASE tools (one version behind the database) the tables and forms that were generated were not making full use of the capabilities provided by the latest database version. A new version is scheduled for end of this year.

Several CASE tools exist on the market, each having their specific development area. In the absence of a fully integrated software engineering environment, one probably have to live with a variety of such tools. Customers would already be greatly helped if all tools could agree on a (set of) common repository (ies).

As a last and modest preliminary conclusion from the management point of view: with rather limited resources it was possible to introduce a reasonably large team into this, for CERN's accelerators at least, new technology; one can estimate the number of software engineers in the accelerator sector to date who are growing familiar with CASE, to around 30.

ACKNOWLEDGEMENTS

The authors are indebted to their colleagues at the European Southern Observatory, Garching-bei-München, Drs. G. Raffi and K. Wirenstrand, for valuable discussions and exchange of experience in the course of the evaluation of RTEE. They also benefitted from the professional advice and consultancy of Drs. J. Davis and L. Hickman both with ORACLE Corporation, U.K.

REFERENCES

[1] B.W. Boehm, Software Engineering Economics, 1981, p. 16.
[2] The Future of computed aided Programming, Corporate Communications 09/88 on "How will Technology affect America in the next 10 Years?", AEG 1988.
[3] B.W. Boehm, op. cit., p.118
[4] D.J.L. Martin, Practical Improvements in the Management of Real-Time Software Projects, 1st IFAC Workshop on Experience with the Management of Software Projects, May 14-16, 1986, Heidelberg, FRG.
[5] Computerworld, April 22, 1991, p. 76.
[6] Yourdon, Modern Sructured Analysis, 1989.