1985

Computer-Aided Specification, Evaluation and Monitoring of Information Systems

Francois Bodart
Institut d'Informatique, Facultas Universitaires Notre-Dame de la Paix Namur-Belgium

Anne-Marie Hennebert
Institut d'Informatique, Facultas Universitaires Notre-Dame de la Paix Namur-Belgium

Jean-Marie Leheureux
Institut d'Informatique, Facultas Universitaires Notre-Dame de la Paix Namur-Belgium

Yves Pigneur
University of Lausanne Switzerland

Computer-Aided Specification, Evaluation and Monitoring of Information Systems

Francois Bodart
Anne-Marie Hennebert
Jean-Marie Leheureux

Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix
Namur-Belgium

Yves Pigneur

HEC, University of Lausanne
Switzerland

ABSTRACT

One of the crucial issues in specifying requirements for an Information System is to guarantee the **effectiveness** and the **efficiency** of their future implementation. The objectives of the methodology, proposed in the IDA project and presented in this paper, have been:

1. To propose a model and an associated language for a more rigourous specification of Information Systems,

2. To develop among others two complementary software tools allowing the experimental *evaluation* of this specification, by prototyping and simulation, before to implement it and, more recently,

3. To outline an automated monitoring of the implemented Information System.

Introduction

The design of an Information System (I.S.) is an activity during which are realized choices related to important and interdependent aspects like: information technologies, information memorization and processing rules, organizational structures and behavior of people. For a long time the constraints of information technologies have deeply oriented and constrained these choices. But now there are more and more degrees of freedom at the level of information technologies choice. Consequently, there is an increasing necessity to introduce more precise specifications of I.S. under design and to evaluate their **impact upon the organization**.

This evaluation aspect is especially important as the I.S. are becoming more and more vehicles for global strategies in organizations. Indeed, they may be related to any activities (operational, managerial, . . .), to any jobs, to any responsibilities; they integrate all the domains of application of information technologies (E.D.P., Process Control, Office Automation, D.S.S., . . .), and all the functions of information processing (acquisition, storage, processing, communication, etc.); they contribute to various kinds of business strategies (productivity increase, definition of new products, improvement of working conditions for employees, etc.).

As a corollary there is also an increasing necessity to monitor the real behavior of the implemented Information Systems in order to control that their behavior reflects their specifications.

It will be noticed that in the past, very little attention was paid to this kind of evaluation. Most of the efforts were oriented to the evaluation of the performances of the technical choices related to the implementation design of the I.S. such as the data base and file design [CARD-73-77], [HAIN-81], [INF-77], [RUST-77], [SCHKOL-78], [TEOR-82], [WIED] and the programs architecture design with the pioneer work of [NUN-71] and [TEICH-71].

The contribution of this paper is to provide an integrated approach for specifying the I.S. requirements and for evaluating them from the point of view of the contribution of the I.S. under design to the achievement of the goals and the performances of the organization, it's to say to improve the **effectiveness** and the **efficiency** of Information Systems.

Related to this perspective, the paper presents the IDA Requirements Engineering Environment—a set of integrated tools—:

- A model and an associated language for a more rigourous **specification** of Information Systems,
- Two complementary software tools allowing the **experimental evaluation** of this specification, by prototyping and simulation, before to implement it and,
- An outline of an **automated monitoring** of the implemented Information System.

SPECIFYING INFORMATION SYSTEM REQUIREMENTS

The model of requirements specification on which IDA is based relates to three specific views of the Information System to be developed:

- The **Process-Event** view to specify the activation condition and linkage of processes;
- The classical **Process-Data** view to specify the data, their structure, their use and translation by processes;
- The **Process-Resource** view to specify the human, technical and organizational resources used by processes.

Each of those three views covers both the **structural** and **behavioral** aspects of the Information System to be implemented.

The structural aspect refers to state, static and decomposition properties (process, event, data and resource structure). According to the decomposition property the model permits the controled decomposition by successive refinements of a global processing structure into more and more elementary processing structures.

The behavioral aspect concerns state change and dynamic properties (triggering of processes modification of data according to processing rules, use of resources during process execution). The behavior or system dynamics is therefore mainly described, at the 3 levels, by the relationships between the Process concept and the three others. The process concept consequently achieves the integration of the three different views.

In addition to the definition of the concepts related to the different aspects of this model, a maximum of consistency and completeness criteria have been defined, against which the requirement specifications can be analyzed.

DSL is a computer-processable language [BOD-79] [BOD-83] which belong to the PSL family [TEICH-79]. Its statements allow the specification of requirements based on the concepts belonging to the presented model.

Therefore, since

- the model allows the specification of sufficient operational details,
- the integrity criteria allows an automated verification of requirements,
- the DSL language is a notation that has a well-defined semantics, the DSL requirement specifications could be "animated" or "executed" through the use of adequate tools.

On the one hand these "executable" DSL specifications can be used to automatically drive **performance simulations** and functional **prototypes** in order to **check for efficiency and effectiveness** of Information System under development.

On the other hand these DSL specifications could even provide the implementation of a **mechanism for monitoring** the actual—developed—Information System.

The first part of the paper briefly outlines and illustrates the three views of the requirements specification of an Information System. A more detailed presentation may be found in [BOD-79], [BOD-83] and [PIG-84].

THE PROCESS-EVENT VIEW

The objective of this submodel is to provide concepts and mechanisms which allow the specification of the functional behavior of an Information System by representing the activation condition of each processing activity. This submodel, illustrated in figure 1, rests on two basic concepts: process and event.

Process

In this view, a process is seen as the execution of a processing procedure which corresponds to a specific activ-

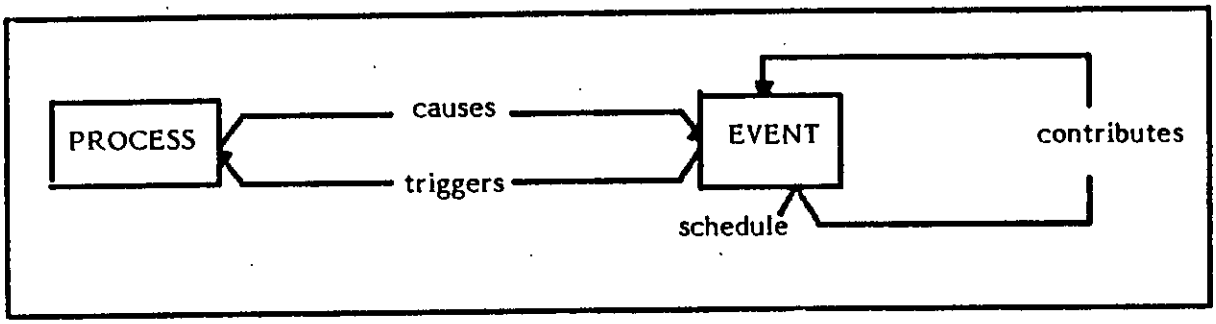

Figure 1

Diagram of the Process-Event view

ity (process type) to be accomplished in the Information System. The procedure itself (how to do) is considered as a "black box" whose progression can be represented, at points in time, by relevant **state changes. In a first step,** we **limit** ourselves to the two most significant state changes of a process life cycle:

- the **triggering** which corresponds to the beginning of the procedure execution and
- the **termination** which occurs when the execution has reached its final stage. Between its triggering and its termination, the process is active or in the course of execution; before its triggering, it was inexistent and after its termination, it ceases to exist having produced all its effects.

From this presentation, it is obvious that a process is a dynamic entity created at each execution of the procedure characterizing the process type and that, at any moment, several processes of the same type may coexist in the Information System at different states.

Event

An event corresponds to a state change of the Information System, localized in time and space, representing a stimulus upon which the system has to react.

An event is called **external** if it corresponds to the crossing of the Information System boundary by a message (see below) originating from its environment. Among external events, we distinguish the so-called temporal events that correspond to the creation of a clock-scheduled message. Those external events happen in the Information system life following the generation **schedule** of their associated message. Then the generation schedule constitutes a mechanism to specify the estimated **workload** of an Information System.

An event is called **internal** if it corresponds to an internal state change of the Information System. These events represent:

- either a state change of a process such as termination,
- or the realization of a synchronization point which is a mechanism of event coordination.

A **synchronization point** defines a compound event and allows for the following elements:

- the events to be coordinated of same or different types;
- the contribution modalities (when, how, how long) of events contributing to the coordination;
- the synchronization condition which specifies the coordination that must occur between the events taken into account for causing the compound event.

TRIGGERING OF PROCESSES BY EVENTS

An event is a stimulus upon which the Information System may react by the triggering of one or many processes of the same or different types. The global behavior of the Information System is then specified by a set of relationships expressing the type of processes triggered by the events of given types. These dynamic actions or **behav-**

ioral rules may be characterized by a **condition** which stipulates that the dynamic action is optional and/or by a **factor** of **multiplicity** which sitpulates the number of processes of the same type to be triggered.

The concepts proposed in this submodel, when combined, allow the description of the conventional structures of process linkage (sequence, decision, parallelism, synchronization, cycle or indeterminism).

Figure 2 shows an example of DSL statements illustrating the Process-Event view of specification.

THE PROCESS-DATA VIEW

The concepts and mechanisms proposed in this classical submodel, illustrated in figure 3, allow the specification of:

- the structure of data stored in the Information System memory and conveyed by means of message,
- the use and translation of data according to processing rules.

We will shortly remind the content of the Process-Data view of specification not only to provide an integrated approach of the modelization of an I.S. but also to describe the prototype of an application which is partially generated from those specifications.

DATA STRUCTURE: AN ERA APPROACH

We have chosen an ERA approach [BEN-76], [CHEN-76], [BOD-83] to structure the Information System memory because of its good qualities of communication and its great capacity for operational representation of information belonging to the real world.

Any information belonging to the Information System memory has as direct or indirect origin (after transformation carried out by a process) the messages conveyed in the Information System. A message conveys, from that time on, informations defined in the Information System memory. These informations are exchanged by the communication channels of the organization:

- either between the Information System and its environment,
- or within the Information System, between the processes.

So, in the diagram of figure 3, the "DATA" box represents Entity, Relationship, Attribute as well as Message and the "RELATES and CONSISTS OF" link corresponds to the different interrelations between these kinds of data.

Transformation of Data by Processes

On the one hand, the relationships between Process and Data are used to refine the behavior (state changes means here data transformation) by specifying input messages and/or the portion of memory respectively received and used for generating output message and/or for modifying memorized data.

On the other hand, the procedure—internal behavior—of a process type, initially considered as a "black box" in the Process-Event view needs to be now specified by describing its processing rules which assures the translation of input data into output data.

Figure 4 shows an example of DSL statements illustrating the Process-Data view of specification.

THE PROCESS-RESOURCE VIEW

This submodel, illustrated in figure 5, serves to characterize resources required to perform the process. This is indeed a vital way of estimating if a specified behavior (see above) is efficient by taking into account the availability of organizational resources, such as, for example, personnel, equipment, financial means, work schedule and job positions.

From a theoretical point of view, it could seem strange to specify resources at the level of requirements that should deal only with functional properties. However any information system has to be considered as a supporting mean for the organization, the objective of which is managing scarce resources to meet performances goals. Therefore requirements may very well include resource constraints and the requirement model must then be capable of specifying them.

Process

At this level of specification, the Information System is considered to be a collection of dynamically created processes which have a given processing duration and compete with each other for shared system's resources.

Therefore the basic model described in the "Process-Event View Section," has to be enlarged because of the

```
* Global behavior

DEF PROCESS Verify-Order ;
   TRIGGERED ON GENERATION OF Order-Voucher ;

   DEF MESSAGE Order-Voucher ;
      GENERATED BY Client EVERY around-3-min DURING Working-Hours ;


DEF PROCESS Stock-Updating ;
   TRIGGERED ON TERMINATION OF Verify-Order IF Order-Valid ;


DEF PROCESS Orders-Delivering ;
   TRIGGERED ON REALIZATION OF Orders-Grouping ;

   DEF SYNCHRONIZATION Orders-Grouping ;
      CONTRIBUTED ON TERMINATION OF Stock-Updating ;
      REALIZED-WHEN :
         COUNT(10) TERM OF Stock-Updating ;
```

Figure 2

Example of Process-Event view statements

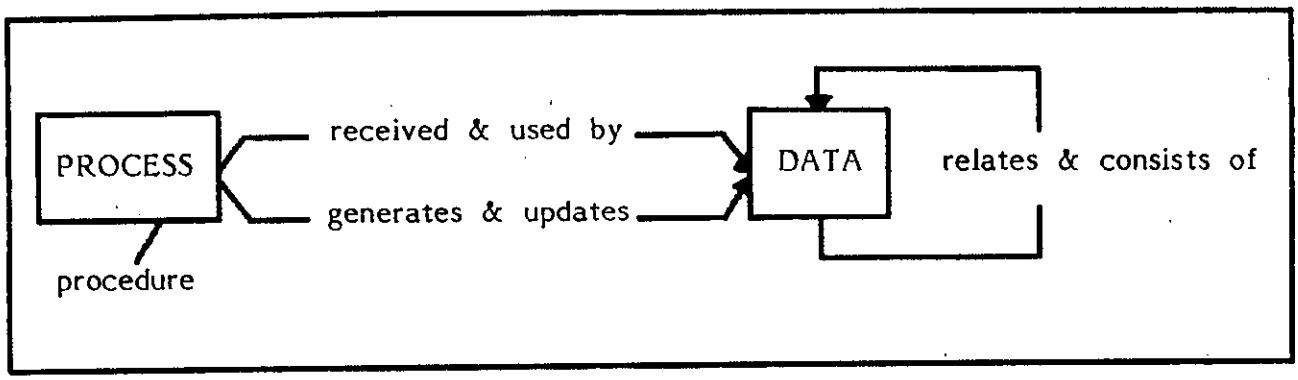

Figure 3

Diagram of the Process-Data view

```
* Data structure


DEF ENTITY Item ;
    SYNONYM IS Product ;
    DESCRIPTION :
      any reference found in the catalog of the firm ;
    CONSISTS OF Item-N°, Item-Name, Stock-Quantity, Item-Price ;
    IDENTIFIED BY Item-N° ;


DEF RELATION Order-Line ;
    RELATES Order, Product ;
    CONSISTS OF Order-Quantity ;
    CONNECTIVITY IS One-many FOR Order, Zero-many FOR Product ;


DEF MESSAGE Order-Voucher ;
    CONSISTS OF Client-Identity, one-or-more Order-Lines ;


* Use and translation of data by processes


DEF PROCESS Verify-Order ;
    RECEIVES Order-Voucher ;
    USES Client, Product ;

    GENERATES Invoice-Entry IF Order-Valid,
              Invoice-Reject IF NOT Order-Valid ;
    UPDATES Client ;

    PROCEDURE :
       *** not specified yet ;
```

Figure 4

Example of Process-Data view statements

introduction of a **new state** in the process life cycle: a process may **await** execution because of lack of resources. So we distinguish three **new state changes**:

- The **inception** which corresponds to the actual beginning of the procedure execution when the required resources are available;
- The **interruption** which occurs when the process, in the course of execution, has to be temporarily stopped because resources become unavailable; and
- The **resumption** which corresponds to the continuation of the interrupted execution when the required resources become again available.

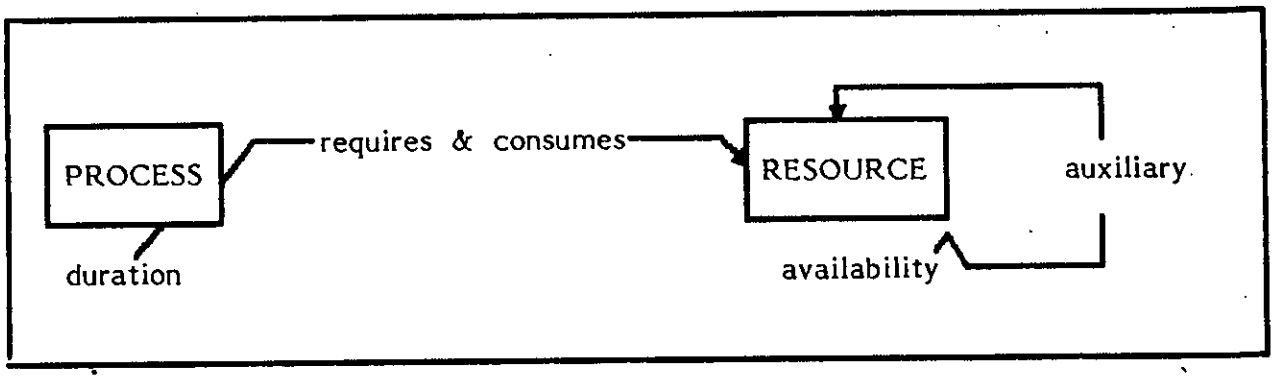

Figure 5

Diagram of the Process-Resource view

```
* Resource description

DEF REUSABLE-RESOURCE Order-Workstation ;
    CAPACITY IS 9 ;
    AVAILABLE DURING Working-Hours ;

    REQUIRES 16 Kbytes-Memory ;

    DEF CALENDAR Working-Hours ;
       ACTIVE Morning ON Working-Days ;

          DEF INTERVAL Morning ;
            SPANS FROM '8H30' TO '12H30' ;


* Use of resources by processes


DEF PROCESS Verify-Order ;
    PERFORMED DURING around-7-min ;
    REQUIRES 1 Order-Workstation ;

    DEF SYSTEM-PARAMETER around-7-min ;
       VALUES ARE FROM '3m' THRU '11m' ;
```

Figure 6

Example of Process-Resource view statements

Resource

A resource represents a volume of identical means a portion of which can be necessary at the time of a process execution.

A resource is called **reusable** if, as soon as a process for which it is required is terminated, is available for other processes. On the other hand, a resource is called **consumable** if it was irreplaceably consumed at the time of a process execution: the consumed amount is then no longer available for other processes.

Any resource can be characterized by an availability calendar specifying the maximum available capacity during given periods of time.

USE OF RESOURCES BY PROCESSES

The **use** of a reusable resource and the **consumption** of a consumable resource by a process are expressed in the form of linear function of execution time of the process and characterized by a requisition or consumption rate.

The execution time of a process is an estimation of the time required to perform the processing rules of its procedure.

Since this submodel may be used at an early stage in the specification, we also introduced a simplifying hypothesis: the requisition rate of a reusable resource is defined on the total duration of the requiring process. In other words, the portion of resource is required during all the execution time of the process.

Duration of processes, requisition rate as well as consumption rate can be expressed in the form of a constant value or a probability distribution.

The use of a resource may also be specified as being equivalent to the total or partial use of other resources. This equivalence relationship expresses the use of a resource considered as "auxiliary" by another resource considered to be "principal," the latter being directly or not required by a process. In this case the amount of required auxiliary resource is proportional to the amount of required principal resource.

Some refinements have also been introduced in this submodel to specify sharable—non additive—use of resources by many processes at the same time, priority rules among processes and interruption mechanisms. Figure 6 shows an example of DSL statements illustrating the Process-Resource view of specification.

EVALUATING REQUIREMENTS SPECIFICATION

The IDA Requirements Engineering Environment,[*] developed in the framework of the SEM System [TEICH-79], [YAMA-1982], consists of an integrated set of tools supporting specification and verification of DSL requirements.

As indicated in figure 7, central to the system is a data base which regroups the ensemble of introduced requirement specifications. The DSL requirements are integrated in this data base through a translator and can then support a wide variety of analysis and display tools. These automated tools have mainly four kinds of functions:

- Generation of **documentary reports** presenting in various forms different aspects of the specified system, for the use of the different people involved in the project;
- Execution of **consistency and completeness** checks on the introduced descriptions. As mentioned above, many integrity criterial have been defined against which specifications can be automatically analyzed;
- Generation and execution of **performance simulations** to check for efficiency of information system under development;
- Generation of **functional prototypes** to check for effectiveness of the requirement specifications.

This part of the paper insists of the two latter facilities which allow the experimental verification of the requirement specifications.

Their goal is indeed to introduce a more experimental approach in the study of requirement specifications by testing to low cost the impact of diverse functioning hypotheses and by facilitating direct perception by users of what they wish to obtain.

The **performance simulation** facility provides a means to **experimentally evaluate the efficiency** of the information system under development and to predict its performances within a specific resource environment before its implementation.

The **functional prototyping** facility allows the derivation of prototypes of small-scaled systems, in the form of throw-away code, in order to check if requirement specifications are effective or conform to the needs expressed

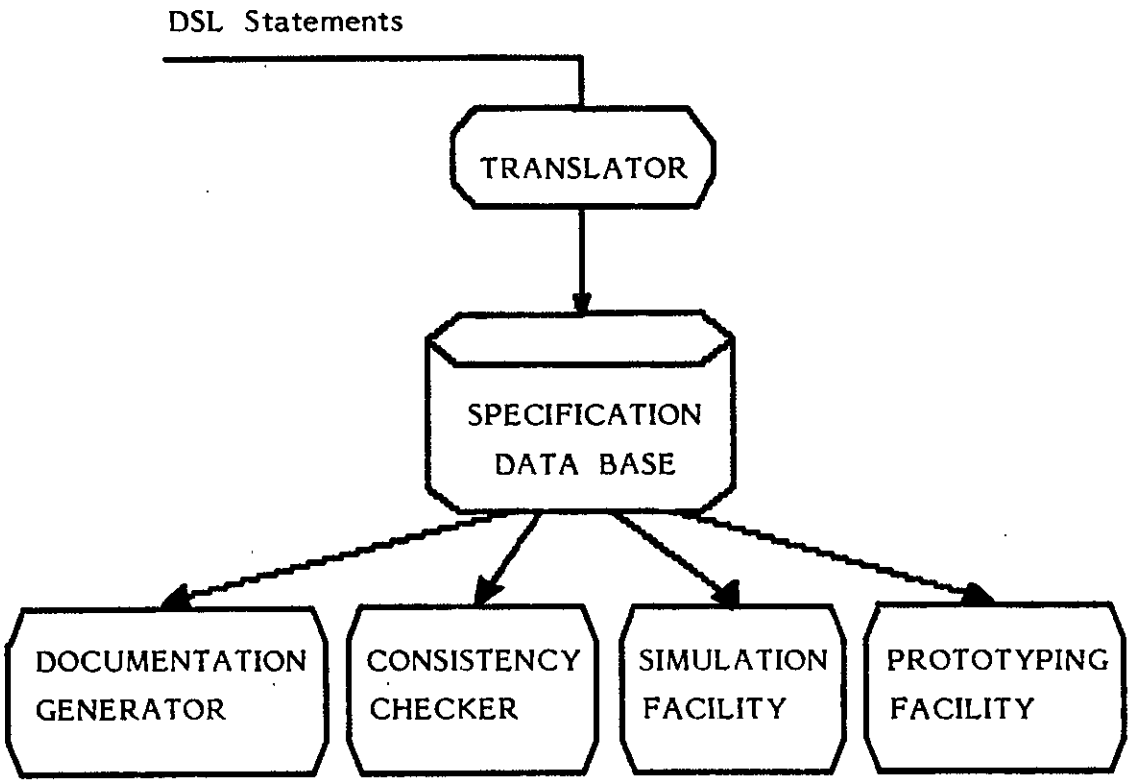

Figure 7

Information flow in the IDA Requirements Engineering Environment

by the organization. The experimental use of a generated prototype should allow to elicit feedback from the end-users early in the development of an information system.

The same idea prevails in the realization of the simulation and the prototyping packages; their execution is completely controlled by a "Behavior motor" or an "Event Processor," virtual machine which is completely driven by the Process-Event specifications of an information system. The function of this "behavior motor" is:

- to detect and interpret events (or stimuli),
- and to react to them by triggering the appropriate processes.

According to the different functions of the two tools, when a process is triggered, the behavior of the tool is different:

- In the performance simulation tool, only the Process-Resource specifications are of interest and a mechanism allows allocation and deallocation of resources, interruption and resumption of processes and recording of statistical measures;
- In the functional prototyping facility, only the Process-Data specifications are used in addition to the Process-Event ones, to animate a model which should permit users to interactively verify if processing rules produce the expected results.

The integration of these simulation and prototyping facilities into a Requirement Engineering Environment overcomes the usual disadvantages of separated approaches. The control of the tools directly by the specifications prevents divergence between the specified and the simulated or prototyped systems. It also permits also faster reactions to change.

The last section of this paper will try to show that the idea of a "behavior motor," driven by the specifications, could be recuperated at the level of the implementation.

PERFORMANCE SIMULATION TO CHECK FOR EFFICIENCY

Overview

The automated **simulation** facility provides a means to analyze the **efficiency** of the specification of an information system within a specific resource environment. The integrated approach which is provided allows designer and the users to evaluate by experimentation the **behavioral performances** of the I.S. under design, to evaluate the **resources** required in order to produce these performances, to detect **misfunctionings** such as potential bottlenecks and to consider various alternative solutions.

The use of this facility is as follows. First, a specification is expressed in DSL, entered into a specification data base and statically checked for consistency and completeness [PIG-84]; after, an executable model can be automatically produced and executed for simulating the behavior of the specified system and producing the desired performance measures.

Figure 8 gives a pictorial overview of the general architecture of DSL/SIM which consists of three processors. A more detailed description of this architecture may be found in [BOD-83-2] or in [PIG-84].

The SIMULATION GENERATION PROCESSOR takes the relevant part of the specifications stored in the specification data base and transforms it automatically into equivalent SIMULATION TABLES for incorporation in the SIMULATION EXECUTION PROCESSOR. Such an automatic generation prevents divergence of the simulation from the specification and allows fast and by product analysis of changes made only in the specification data base.

The SIMULATION EXECUTION PROCESSOR drives the simulation according to the automatically generated SIMULATION TABLES, traces all the primitive events occuring during the simulation, gathers statistics on them and stores them in a SIMULATION DATA BASE of results.

The SIMULATION ANALYSIS AND DISPLAY PROCESSOR processes the statistical results recorded in a SIMULATION DATA BASE to produce various reports on request.

The presentation of simulation results, for evaluating and displaying the behavior of the simulation, may be requested:

- for selected processes, resources or synchronizations;
- for different intervals inside the simulation period (for instance, "every Monday from 8:30 AM to 5:30 PM");
- in aggregated form that includes average and extreme values computed for the entire period considered; or
- in chronological form.

Sample simulation results for a process type is shown in figure 9.

EFFICIENCY CHECKING

Throughout the execution of a performance simulation, a number of measures are recorded in a data base in order to serve in statistical analyzes of the behavior of an information system under development. These measures could be used to detect performance problems and to localize their causes. A method of interpretation of these results is outlined hereafter.

A. A first step is to identify and to localize the **divergences** between **expected performances**, specified before the simulation, and **estimated performances**, established by the simulation. Two types of measures will be used for this purpose:
   - A measure of the **delay** between two succeeding classes of events (for instance, the delay between initial events and ending events);
   - The **number of events** in each class in order to control if the numbers of events in the succeeding and preceding classes are coherent.

B. For the part of the graph where there are divergences between expected delays and estimated delays, the cuases for possible **lags in the triggering of process** will be identified by the analysis of the measures related to synchronization points. For a synchronization point where the realization time is large, constraining events contributing to the synchronization will be identified. For these events, if they are internal events, the paths causing them will be analyzed.

C. The next step will be to detect, mainly inside the critical path of the graph localized during the previous steps, classes of processes of which the **waiting time is abnormally too long** and/or where the **interruptions** are too much frequent or too long. These situations will be discovered by the analysis of measures related to the behavior of processes.

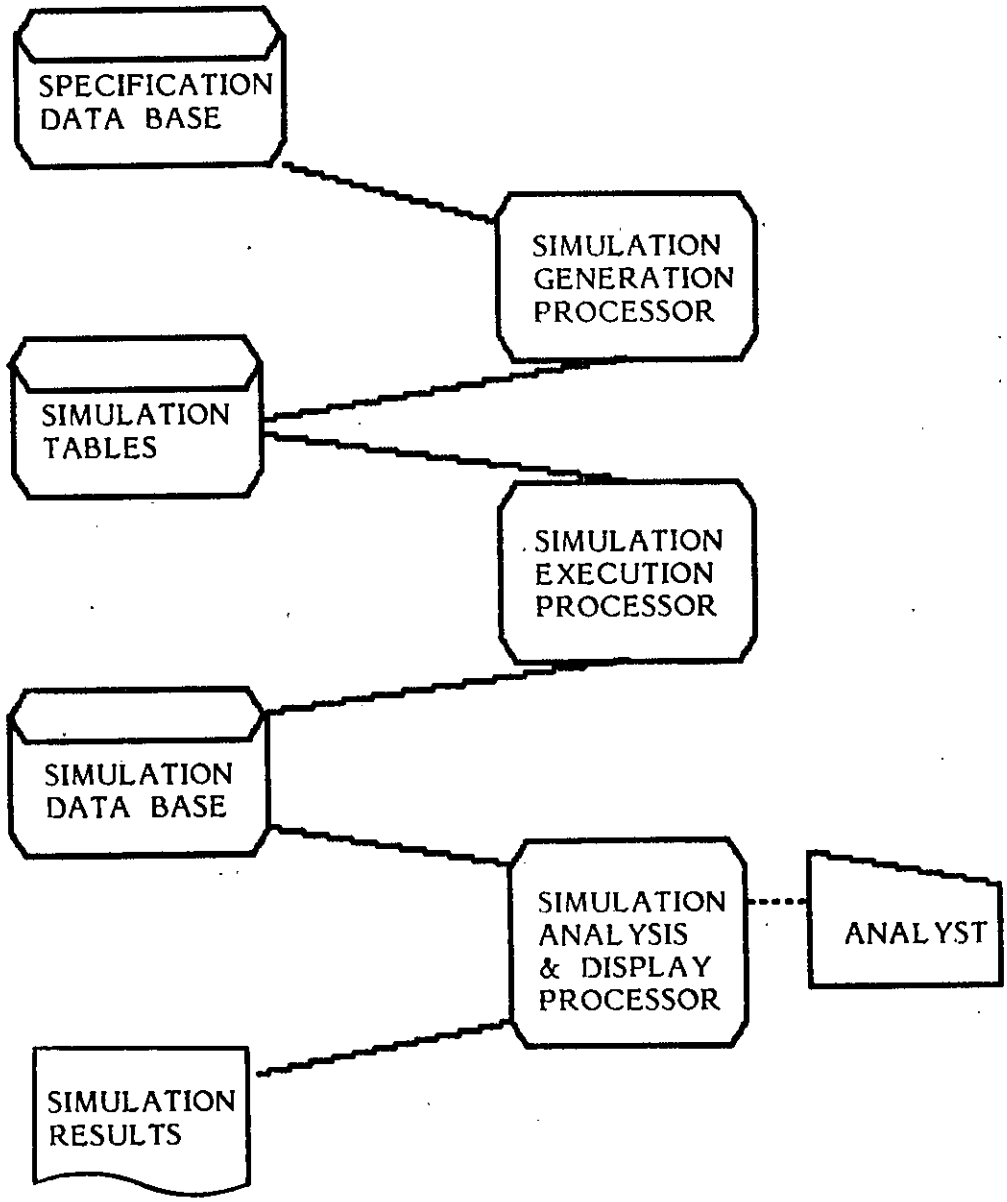

Figure 8

Information flow in the IDA Simulation facility (DSL-SIM)

```
Interactive Design Approach  Version SEM1.3R0

                          Global Statistics

PROCESS Enreg-Comm-Cli

   REPORTED BETWEEN 0s AND 10d

 DEFINE CALENDAR RELATED TO PROCESS Enreg-Comm-Cli
   SPANS FROM 8h30m TO 12h30m, FROM 13h30m TO 17h30m ON 0 TO 5 DAY PER WEEK

   Triggering Number          4603
   Inception Number           4603
   Interruption Number          73
   Resumption Number            73
   Termination Number         4603

                                MIN.          MEAN          MAX.      Std. Dev.

   Active Time                    3m          7m60s        12m60s         2m55s
   Waiting Time                   0s        1h59m21s      4h28m8s        1h8m18s
   Interruption Time              0s             0s            0s             0s
   Idle Time                      0s         37m51s           63h      6h15m34s
   Elapse Time                             2h45m48s

   Numb.of Proc.in Wt.State      0.00          143.71        330.00         88.18
   Numb.of Proc.in Int.State     0.00            0.00         10.00          0.00
   Numb.of Proc.in Act.State     0.00            9.63         10.00          1.76

   Required Resources :                    Capacity                Sharability          Queue
                                          Max.        Used        Max.        Used       Size

   Cell-Reception                        10.00        9.63                    9.63     143.71

   [Low.Bnd. Upp.Bnd.[        Number of
                          Trg.  Inp.  Int.  Res.  Ter.      Wait.Time     Int.Time    Idle Time

   08h       09h          2276   317     0    41   278         57m6s           0s     7h59m40s
   09h       10h             0   608     0     0   608         1h15s                      0s
   10h       11h             0   607     0     0   607      1h59m55s                      0s
   11h       12h             0   575     0     0   591         2h59m                      0s
   12h       13h             0   170    32     0   202        3h43m2s                     0s
   13h       14h          2327   329     0    32   281        54m40s           0s       14m38s
   14h       15h             0   584     0     0   584         1h19s                      0s
   15h       16h             0   601     0     0   602         2h11s                      0s
   16h       17h             0   594     0     0   593      2h59m57s                      0s
   17h       18h             0   218    41     0   257        3h43m9s                     0s
```

Figure 9

Example of simulation results

D. Finally, **critical resources** will be identified. A resource is critical if it is the cause of abnormal waiting times, interruption times and/or idle times either because the level of utilization is too large or because the available capacity is too small.

From an economical point of view, a resource could also be critical when the availability of the resource is too large in relation to the level of utilization. Critical resources will be identified through the analysis of measures related to the behavior of resources.

An efficient solution will be achieved through a searching process based on the interpretation of the simulation results. The searching process will be defined by acting upon:

- The expected performances and the workload of the system,
- The resource constraints,
- The specified functional behavior (Process-Event View).

FUNCTIONAL PROTOTYPING TO CHECK FOR EFFECTIVENESS

Overview

The **prototyping** facility (DSL-PROTO) provides a means to implement the requirements: these are extracted from the specifications data base and transformed into executable statements. So, the analyst and the end-users are able to determine **experimentally** if the requirements specification will produce the results they are expecting.

Three goals are assigned to the current version of DSL-PROTO:

1) First, in proportion as the system is specified and before starting the implementation design, to provide the customer/end-user with an **integrated and global view of the behavior of the I.S.** Indeed, DSL-PROTO allows to prototype the man-machine interface, the application system functions (process behavior) and the flows of messages as well as the interaction between manual and computerized process.

2) Secondly, to allow an **experimental validation** of the requirements by the analysts and the end-users. Indeed, it provides them with facilities:

   - To **understand** the requirements as soon as they are established for limited but coherent subpart of the project.
   - To **validate** them by evaluating their adequacy with respect to the user's effective needs.
   - To **improve** them as a consequence of the validation.

3) Thirdly, DSL-PROTO may be used as an **educational tool** for the **training** of the end-users of the project while the latter is under development.

Figure 10 gives a pictorial overview of the general architecture of DSL-PROTO which consists of three processors. Only a few aspects of this architecture will be outlined. A more detailed description may be found in [BOD-85].

In order to minimize the sources of discrepancies between the requirements specification and the prototype, a maximum number of elements of the prototyping programs are **automatically generated** from the requirements specification data-base with the requirements specification constituting **consistency constraints** to be verified by the prototyping programs. Three main aspects of the automatic generation are to be considered:

- In the prototyping programs, the declaration and the description of messages and data structures are automatically generated from the part of the requirements data-base related to the Data Structure. (see the Process-Data view).
- The inputs of process (data contained in messages and/or elements of the data-base) and the outputs of processes (messages and/or updating of the data-base) are fully defined by the statements of the Process-Data View.
- The execution of a sequence of processes is totally controlled by the Prototype Generation Processor which interprets the statements of the Process-Event view.

The analyst who prototypes the application has to complete the elements derived from the automatic generation by replacing the text of the procedure written in free format by an algorithm written in a very high level **prototyping language**.

This language includes more, particularly:

- Structured programming constructs, and
- A set of data manipulation primitives allowing to manipulate the data structures described in DSL.

The following example (see figure 11) will give a flavor of the prototyping language, the statements of which are nested [KONS-75] in DSL statements.

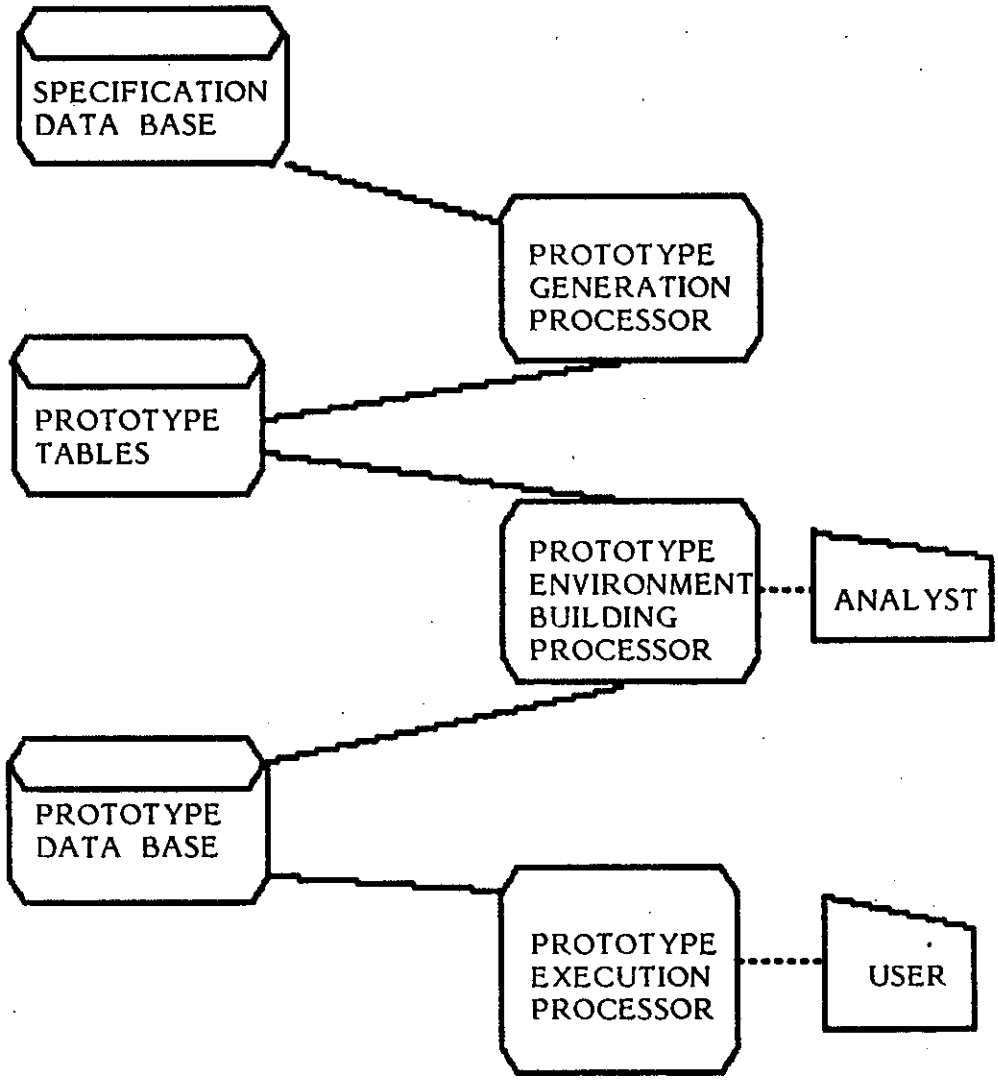

Figure 10

Information flow in the IDA prototyping package

COMMENTS

The algorithm written in the nested syntax is described as a text associated with the DSL statement PROTOTYPING PROCEDURE. Other statements are DSL statements about inputs, output and the dynamics of the process.

The examination of the example allows to discover an important characteristic of the prototyping language: it works on the object representation provided by DSL. It means that the user, the analyst and the "application-programmer" **shares a common representation** of the I.S. objects. So, it becomes easier either to establish a true dialog between them or for the same person to assume these different roles more easily.

EFFECTIVENESS CHECKING

The effectiveness checking may be achieved through many facilities provided by DSL-PROTO.

A. Through its different processors, DSL-PROTO will control the **consistency between** the **requirements specification** and their **implementation**. For instance:

```
    * Process inputs, outputs and dynamics

DEFINE PROCESS     Verify-Order ;
  RECEIVES         Order-Voucher ;
  REFERENCES       Client-Identify INTO Client ;
  GENERATES        Accepted-Order-Voucher IF Client-Exists ;
  TRIGGERED BY GENERATION OF Order-Voucher ;
  ON TERMINATION TRIGGERS Client-Creation IF NOT Client-Exists ;
  ON TERMINATION TRIGGERS Order-Registration IF Client-Exists ;

    * Process Procedure

PROTOTYPING-PROCEDURE ;

  PROCEDURE Verify-Order

    DEFINE Index FORMAT IS INT (2)

    DEFINE I FORMAT IS INT (2)

    RECEIVES Order-Voucher

      LET Index := CARDINALITY (Order-Line OF Order-Voucher)

      FOR EACH Client
        WHERE Client-Identify OF Client = Client-Identify OF
              Order-Voucher

        LET Client-Exists := T

        OPENGENERATES Accepted-Order-Voucher

          LET Client-Name OF Accepted-Order-Voucher :=
              Client-Name OF Order-Voucher

          LET Client-Identify OF Accepted-Order-Voucher :=
              Client-Identify OF Order-Voucher

          LET I := 0

          WHILE I <= Index

            LET Item-Number OF Order-Line (I) OF
                    Accepted-Order-Voucher :=
                Item-Number OF Order-Line (I) OF Order-Voucher

            LET Order-Quantity OF Order-Line (I) OF
                    Accepted-Order-Voucher :=
                Order-Quantity OF Order-Line (I) OF Order-Voucher

          ENDWHILE

        GENERATES Accepted-Order-Voucher

        WHEN NONE

          LET Client-Exists := F

        ENDFOR Client

    ENDRECEIVES Order-Voucher

  ENDPROC Verify-Order
```

Figure 11

Example of a prototyping procedure nested in DSL statements

During the translation of the DSL statements into an executable program (see figure 10), the Generation Processor checks if the algorithm written in the nexted syntax is consistent with its external specifications written in DSL. For instance, verifications are realized about the **reciprocal** relationships between DSL primitives used to specify actions on data structures and the equivalent prototyping primitives. So, if a DSL statement specifies that "a process REFERENCES an element of a data structure" the Generation Processor will verify if there is an associated "FOR EACH . . . WHERE . . . " statement in the nested syntax. Reciprocally such a "FOR EACH . . . WHERE . . . " prototype primitive cannot exist in the prototype procedure related to a Process if a "REFERENCES . . . " statement is not specified for that process.

At the level of the environment building (see figure 10) a screen manager directly interacts with the specification data-base. So the definition of a screen is driven by the specification of a Message and its data-structure (elements of the message, their structure, their format).

During the execution of the prototype (see figure 10) the integrity constraints (format, domain of values, connectivity of a relationship, . . .) formally specified in DSL are automatically checked when data are stored in the data-base or when they are assigned to a message.

B. While executing the prototype, the end-user may obtain **"tracing" informations** about the behavior of the information system.

Before the execution of a procedure he may obtain the display of:

- The messages received by the procedure;
- The state of the data structure tables (entity and relationship tables) that will be used by the procedure.

After the execution of a procedure he may finally request the display of:

- The messages generated.
- The state of the data structure tables used by the procedure.
- The values assigned to a condition that would control the next processing path to be selected.

C. There are also facilities to help the analyst or the end user to **choose initial values** to assign to external messages in order to execute a well defined path in the system.

D. For an educational purpose the user may ask to save the environment related to a subpart of the system in order to be able to **re-execute** it in the same conditions.

COMPUTER-AIDED MONITORING OF IMPLEMENTED INFORMATION SYSTEMS

Even if a computer-aided methodology, as outlined above, is available, the specification of both the structural and behavioral aspects of an Information System requires a great deal of energy. It should be frustrating to lose the advantages of well-established and—experimentally—validated specifications by a less careful implementation.

Now the implementation of an Information System mainly concerns the transformation of the Process-Data view of the specification: the specifications of the processes and data are respectively transformed into programs (or manual procedures) and data bases (or files). On the other hand the Process-Event and the Process-Resource views of the specification are rarely automated in the implemented Information System. The triggering of processes and the allocation of resources are almost always manual. Indeed a human agent:

- Recognizes and interprets the external stimuli and
- Reacts by triggering the adequate processes, by allocating the required resources to the processes.

One can easily imagine the inconsistencies in the behavior or at least the discrepancies between the specified and the observed behaviors usually caused by these manual interventions. These inconsistencies or discrepancies originated from:

- Either a bad interpretation of external solicitations,
- Or a bad reaction corresponding to the triggering of inadequate processes or to a nonoptimal allocation of resources.

These discrepancies are difficult to control and can easily generate an ineffective Information System resulting from a behavior not conform to the functional specification or an inefficient Information System coming from a suboptimal use of resources.

Therefore it seems very attractive to consider a computer-aided monitoring of Information Systems. This idea was already advocated in the ISO report [ISO-82] (see the concept of Information Processor). Our contribution comes to say that it should be possible to adopt the same strategy as into the prototyping and simulation facilities by implementing a "behavior motor," directly driven by the—executable—Process-Event specifications, into the actual Information System.

Indeed this monitoring should lead to entrust a software facility—or "behavior motor"—with the total or partial control of the Information System behavior. This facility should be able:

- To recognize among the external stimuli of the Information System the relevant events and,
- To react by triggering the appropriate processes and allocating the resources required in a reasonably optimal way.

In such a context, the Process-Event and Process-Resource views of the specifications should also be implemented in the operational Information System by a tailor-made programmed mechanism. If we assume that the specifications have been—experimentally—validated, we could hope to limit the risk of inconsistencies and discrepancies mentioned above and so improve the effectiveness and the efficiency of the implemented Information System.

However, a "too-strong" automation may also increase the rigidity of the behavior. Therefore it should be necessary to design the computer-aided monitoring facility in such a way as to maintain a flexible behavior of the Information System. The human interventions should still remain allowed but would be carried out under the control of the computer-aided monitoring mechanism which should prevent a "too-strong" distortion between the actual and specified behaviors.

The computer-aided pilotage facility should be seen itself as the (meta-)Information System of the implementation Information System and therefore should have a similar architecture where:

- The processing rules should correspond to the behavioral rules implementing the Process-Event and Process-Resource views of the specifications;
- The memorized data should correspond to programs and files implementing the traditional Process-Data view of the specifications;
- The inputs/outputs should respectively correspond to the external events from and to the environment of the Information System. Some special outputs should also be generated for monitoring purpose and should inform the human agent about the decisions taken by the computer-aided pilotage mechanism;
- The processor should correspond to a "behavior motor," virtual machine able to interpret the behavior rules. It should be able to recognize the relevant events and to react triggering the adequate processes and allocating the required resources in a reasonably optimal way. This "behavior motor" should be directly inspired of the mechanism which was successfully implemented in the prototyping and simulation facilities.

EXTENSIONS

Our experience of introducing and applying the IDA Requirements Engineering Environment in private and public organizations has convinced us to develop specific extensions such as:

- Monitoring the process of modeling the requirements. Initially introduced by C. Rolland [ROL-82] (see the pilot system, pp. 418–422), the idea is to supply the analyst with computer-assisted aides for applying the concepts and associated rules of the three views model.
- Interfacing the IDA environment with high level programing environments such as relational D.B.M.S. and associated fourth generation languages.
- Reusing the tests plan set up during the prototyping step in order to produce more accurately the acceptance tests for the product to be delivered.

REFERENCES

[CARD-73] CARDENAS, "**Evaluation and Selection of File Organization—Model and System**," Com. ACM, Vol. 16, p. 9, 1973.

[BEN-76] BENCI E., BODART F., BOGAERT H. and CABANES A. (1976), "**Concepts for the Design of a Conceptual Schema**" in NIJSSEN G.m., Modelling in Data Base Management Systems, North-Holland, 1976.

[BOD—79] BODART F. and PIGNEUR Y., "**A Model and Language for Functional Specification and Evaluation of Information System Dynamics**," Proceedings of the IFIP TC-8 Working Conference on Formal Methods and Practical Tools for Information System Design, Oxford, U.K., April 17–20, 1979, North-Holland.

[BOD-83] BODART F. and PIGNEUR Y., "**Conception assistée des applications informatiques: Méthodes, Modèles et Outils: Etude d'opportunité et analyse conceptuelle**," Masson, 1983.

[BOD-83-2] BODART F., HENNEBERT A.M., LEHEUREUX J.M., MASSON O. and PIGNEUR Y., "**DSL-DSA: A System for Requirements Specification, Prototyping and Simulation**," Proceedings of the IFIP-TC2, Working Conference on System Description Methodologies, Kecskemet, Hungary, June 1983, North-Holland, 1985.

[BOD-85] BODART F., PIGNEUR Y., HENNEBERT A.M. and LEHEUREUX J.M., "**The Experimentation of Information System Requirements by Simulation and Prototyping in the IDA Project**," ISDOS-IDA Prise Meeting Paris, January 1985.

[CHEN-76] CHEN P.P., "**The Entity-Relationship Model. Toward a unified View of Data**," ACM TODS, Vol. 1, 1, 1976.

[HAIN-81] HAINAUT J.P. (1981), "**Theoretical and practical tools for data base design**," V.L.D.B. conf. Proceedings, Sept. 1981, ACM/IEEE.

[INF-77] "**INFOTECH State of the Art Report: System Tuning**," Infotech Intern. Ltd., 1977.

[ISO-82] Van Griethuysen (Ed.), "**Concepts and Terminology for the Conceptual Schema and Information Base**," Doc. n° ISO/TC97/SC5-N695, ANSI, March 1982.

[KONS-75] KONSYNSKI B., "**A Model of computer-aided definition and analysis of information system requirements**," Ph.D. Thesis, Purdue University, 1975.

[NAUM-82] NAUMAN J.D., JENKINS M.A., "**Prototyping: the New Paradigm for Systems Development**," MIS Quarterly, vol. 6, nr. 3, September 1982.

[NUN-71] NUNAMAKER J.F., "**A Methodology for the Design and optimization of Information Processing Systems**," AFIPS Conference Proceedings, Vol. 38, 1971.

[PIG-84] PIGNEUR Y., "**Modélisation et évaluation du comportement dynamique d'un système d'information**," Thèse de Doctorat, Namur, 1984.

[ROL-82] ROLLAND C. and RICHARD C., "**The Remora methodology for Information System Design and Management**," in "**Information System Design Methodologies: a comparative review**," North-Holland, 1982.

[RUST-77] RUSTIN, "**Data Base Management System Performance Estimation**," in (INFOTECH-77), 1977.

[SCHKOL-78] SCHKOLNICK, "**A survey of Physical Database Design Methodology and Techniques**," in 1978 V.L.D.B. Conf. Proceedings, IEEE, 1978.

[TEICH-71] TEICHROEW D., "**Automation of System Building**," Datamation, Aug. 15, 1971.

[TEICH-79] TEICHROEW D., MACASOVIC P., HERSHEY E.A. III and YAMAMOTO Y., "**Application of the Entity-Relationship Approach to Information Processing Systems Modeling**," Proceedings of the International Conference on Entity-Relationship Approach to Systems Analysis and Design, Los Angeles, December 10-12, 1979, North-Holland.

[TEOR-82] THEOREY, FRY, "**Design of Database Structure**," Prentice-Hall, 1982.

[VAN LAMS-82] van LAMSWEERDE A., "**Automatisation de la production de logiciels d'application: quelques approches**," T.S.I.—Technique et Sciences Informatiques, Vol. 1, n° 6, 1982, Vol. 2, n° 1, 1983, Vol. 2, n° 2, 1983.

[YAMA-82] YAMAMOTO Y., "**An approach to the Generation of Software Life Cycle Support Systems**," Ph.D; dissertation, University of Michigan, Ann Arbor, 1982.

[WASS-82] WASSERMAN, A.I., "**The Use Software Engineering Methodology: An Overview**," Proceeding of the IFIP Tc-8 Working Conference on Comparative Review of Information Systems Design Methodologies, Noordwijkerhout, The Netherlands, 10-14 May, 1982, North-Holland.

[WIED-77] WIEDERHOLD, "**Data Base Design**," Mac-Grawhill, 1977.

FOOTNOTE

[*]This environment has been extensively used in several projects for industrial, banking and insurance companies as well as for public administrations.