Towards an Effective and Integrated Design of Information Systems

B. Koninaf

King Faesal University-Saudi Arabia

Received ٠٣/٤٠/٨٩٩١
Accepted ١١/٥٠/٢٠٠٢

ABSTRACT

The development of Information Systems (I.S) is of great importance for researchers and industrial users. Different methods for (I.S) design have been proposed until now. Some of them put emphasis on the statical aspect, and others on the dynamical aspect. A third category of methods has recently appeared, which try to take into account both aspects, and therefore provide unified view of data and treatments. Also there exist approaches that put the stress on rigour in the specification and validation process.

This paper presents a tool (CASE) which constitute a conceptual help to Information Systems design: that tries, on one hand, to handle the statical and dynamical, aspects while providing at the same time, the user with products that are readable and easy to understand and on the other hand, to validate the specification obtained in a rigorous way. In fact, it is an attempt to satisfy both the designer and the user.

Key words: Information System design, Conceptual Schema, Validation, Statical Aspect, Dynamical Aspect, NIAM, JSD, Rewrite Logic.
INTRODUCTION

The design phase is paramount in the life cycle of Information System (I.S.). Its aim is to produce a detailed specification of I.S. Designing an I.S is a complicated task: it could be achieved by using
methodologies which are generally supported by tools. However, it can be noticed that the latter are themselves complicated systems, either because they are too much theoretical to be accessible to the designer, or because they are “black boxes” which produce documents and whose logic is seldom understandable by their users (it is the case for most help design tools) [8].

The difficulty inherent to the design process has lead us to develop a tool (CASE) which provides help to the designer, because the design phase is an essential one in the software life cycle. Section 3 of this paper presents the design methods under three different points of view. Our problematic is presented in section 4. The objectives of the tool, its architecture and its description constitutes section 5. Finally, a conclusion is proposed in section 6.

DESIGN METHODS

In this paper, the I.S design methods have been looked at from three different points of view:
- the Historical or chronological point of view
- the Aspect point of view
- the Formalism point of view

a) A Historical or Chronological view

I.S Design methods can be chronologically classified into Cartesian, systemic and, currently, object-oriented methods.

The Cartesian methods are characterized by the way in which the design process is managed. These methods focus on the decomposition of the different phases of the design process into steps, which are in turn subdivided into sub-steps. These methods propose a functional approach. In this approach, the I.S is considered as an information processing system which memorises, treats, formats and communicates data. The I.S, seen as a “black box”, is defined by the output it produces. These methods have been influenced by the Structured Programming Method. They are based on top-down functional decomposition, like HIPO and SADT.

Other methods, inspired from the structured design methods, like YOURDON and MYERS, pay attention to the structure of one module. These methods use the information flow principle. The decomposition is represented by a diagram of data flow among modules like, PSL/PSA & SREM. Some methods are often associated with the use of Petri nets; which modelize the triggering conditions in the I.S process.
The I.S is considered as a system which generates transitions from one state to another. The I.S structure is a transition graph. Other methods have insisted on the development of high level languages, which allow the automatization of the I.S specification, like USE & CS.

**The Systemic methods:** the bearing is not essential for these methods. In this approach the I.S is viewed as a system composed of elements and of relationships between these elements. They emphasize on the global aspect of I.S and allow to design, in a global way, an I.S which is able to produce all the pertinent facts about the organization. The methods based on the systemic approach make the design process look like a modelization process: the result is an abstract representation (Conceptual Schema) of the real world built with the concepts of the model that was used.

The first design data models modeled the statical aspect. One can mention the Relational Model which represent data through tables or relationships. It uses the attribute or property concepts, and the functional dependencies between properties. These models do not represent the semantics. The basic idea of the semantical models is to represent most concepts (entity (or object) and association (or relationship)) in the same schema. The semantical net proposes two concepts: aggregation and generalization, that are borrowed from Artificial Intelligence. Among semantical models, one can mention ACM/PCM, TAXIS, ...

More recently, the current models suggest the integration of the dynamical aspect in the Conceptual Schema (C.S) of the real system. These models allow the enrichment of the C.S by representing the structure and the data behavior, like ACM/PCM, REMORA.

**The Object-Oriented methods:** The object-oriented approaches, at first developed for Programming Languages, have greatly been extended beyond the initial scope of programming, they aim to apprehend the Data Base domain and therefore represent the current methods. The main interest of these approaches lies: on the fact that the users can work on new applications that demand the representation of new data types (in particular graphics, texts and images) that classical relational DBMS can not handle. While for the designers, on the fact that these approaches provide unified view of data and treatments. By contrast to previous approaches an object is, at the same time, defined by its data structure, the manipulations it has to undergo and the triggering events.

The concepts used in these approaches are those of object, class,
aggregation, generalization,... Among the object-oriented methods, there are COAD & YOURDON [È], BOOCH, HOOD,...

b) The Aspect view

From the aspect point of view, and by analogy to systems theory, three fundamental aspects are defined, namely: the Structural, Functional and Control aspects that correspond in I.S to Data, Treatments and Behavior aspects. A system whose behavior is not mastered cannot be controlled.

The Data oriented perspective: it focuses on the types of information stored in the Data Base (D.B), the constraints applied to these data and the derived data types. This perspective is expressed at the highest level by the Conceptual Schema (C.S).

The Treatment oriented (process) perspective: it focuses on the processes or activities performed in the application domain. These perspective can help in the understanding of the way in which a particular task proceeds.

The Behavior oriented perspective: it is concerned with the way real world events trigger actions in the I.S and, more generally, how causal or temporal constraints are applied between the events.

c) Formalism view:

From the formalism point of view, there are two main families of methods:

- The Informal methods (based on graphical formalism)
- The Formal methods (based on mathematical formalism)

The methods of the first family are characterized by their power on the semantics side towards the user (the graphical aspect) because they lead to a C.S, then to the implementation. Unfortunately, they are weak on the validation side in the sense that rigorous validation cannot be done. Validation by test (with the user), or by simulation is the sole alternative. As examples of these, one can cite MERISE, REMORA, NIAM [È], JSD [È] & [È].

The formal methods are characterized by their richness on the validation side: they allow the verification of some specification properties, especially those of (completeness) and consistency. Thus, guarantees will be given before proceeding to the implementation. Unfortunately, these methods are poor on the semantics side. For examples, one can mention the Abstract Data Types [È], the Metoo method [È] & [È], the Z language [È], the VDM [È], the Rewrite Logic [È], [È], [È], [È].
SCOPE of The PROBLEM

Our work is based on the last two points of view. Our objective is not to present a tool which supports a method in order to popularize it, as is the case for SECSI [8] and OICSI [9], or a tool that takes into account only one aspect and ignores the others two, as is the case for PC-IAST and RIDL. Our preoccupation is to present a set of tools (a CASE tool) that tries, on one hand, to handle the three fundamental aspects (Data, Treatment and Behavior) in an I.S and, on the other hand, to regroup the advantages of the two types of specification informal and formal, in order to preserve both the semantics and the validation.

To do this, it was not possible to find a method which could fulfil all those objectives. Else, it would have been the ideal method. Consequently, the only solution was to combine several methods. But one must bear in mind that, in addition to the satisfaction of the specified needs, the combined methods must be compatible (in the sense that, within limits, there exists some duality between them), and, in particular, each one must complement the other [4].

In our case, we have opted for the NIAM method, the JSD approach and the Rewrite Logic formalism. The reasons for this choice and the advantages of these combinations will be progressively given during the description of the modules (tools) in order to show the integration and the linking between the different modules.

We have in fact realized a set of tools (a case tool) that aims to provide some flexibility for the user in the sense that some modules could be used independently from the others, or in relationship with the others.

ARCHITECTURE and FUNCTIONALITIES of The TOOL

Objectives of the Tool

The objective of the tool is to provide help in the specification and the conception of information systems. It is adapted as a combination of the NIAM and JSD methods, and the Rewrite Logic. It aims to satisfy the needs mentioned in the precedent paragraph (scope of the problem). This approach has the following objectives:

* In our work, a module may be considered as a tool because it supports a design method as is the case for M2 and M3, M1 (+ elaboration process of C.S) and MS (a sub-language of specification).
To Guide the designer in the elaboration of the C.S so that a comprehensive and readable product can be presented.

- To Help the designer, to complement the specification of data by those of treatments and behavior.

- To Help the designer to validate the specification (in as rigorous a way as possible) before proceeding to the implementation.

**Global Architecture**

The Tool is composed of five modules (Fig. 1):

- The M₁ module: the C.S.D.P (Conceptual Schema Design Procedure) which has originally been developed by G. Nijssen (University of Queensland) and E.D. Falkenberg (Katholieke Universiteit, Nijmegen). The idea to represent the C.S by using elementary sentences in natural language was first proposed by Falkenberg and then influenced by the linguist C.J Fillmore. A fundamental approach for design construction, that starts by specific examples and follows a well defined procedure using significants visuals diagrams easily filled up for validation, was at first developed by Nijssen at Control Data: this methodology was called NIAM (Nijssen Information Analysis Methodology). NIAM is also

---

[Fig. 1: Architecture of the Tool]
categorized as an idea-oriented model because of its insistence on the idea types. This method is similar under some aspects to the E/R (Entity/Relationship) model but we think that NIAM provides also a high level procedure for the Relational Data Base Design, which possesses many advantages over traditional normalisation techniques.

The role of this module is to modelize all the elaboration phase of C. S (Conceptual Schema) on the basis of NIAM. This is done by starting from the Input Form (I.F) and/or Output Report (O.R) which will then be analyzed and transformed through 9 stages (described in figure 2) to obtain the global C.S corresponding to the universe of discourse.

1. Transform the information examples into familiar elementary ideas and apply the quality verifications
2. Draw a first draft of the C.S diagram and apply a population verification.
3. Cancel the type of Identities which are redundant and the common roles, and identify the type of derived ideas.
4. Add the unicity constraints for each type idea.
5. Verify that all the types of ideas are in correct order.
6. Add the constraints of identity, totality, types and of sub-types and the occurrence frequency.
7. Verify that each entity could be identified.
8. Add the equality, exclusion, sub-sets constraints and others.
9. Verify that the CS is coherent with the original examples, shows no redundancy and is complete.

Fig. 2 “Stages of C.S.D.P”

The procedure starts with the analysis of information examples which constitute the output or input (O.R or I.F) of I.S The three first stages are concerned with the identification of the idea types (stored or derived). In [81] the following stages, constraints are added to store ideas types and, during the procedure, questions are asked by the tool and verifications are performed to make sure that no mistake was made. In [81] a demonstration of the different stages with explanation examples is given.

The following example concerns the stage 2 of the figure 2. The main task in this stage is to draw a diagram which shows all the types of ideas. The output Report of the following table (fig.3) may be an input of the MI module:
Table

<table>
<thead>
<tr>
<th>Drives:</th>
<th>Person</th>
<th>Cars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adams B</td>
<td>ğEPZN</td>
<td></td>
</tr>
<tr>
<td>Jones E</td>
<td>ğEPZN</td>
<td></td>
</tr>
<tr>
<td>Jones E</td>
<td>ğeAAQ</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6 “Output Report Example”

The information in this report can be described by three elementary ideas:
The Person with name ‘Adams B’ drives the Car with reg # ğEPZN’.
The person with name ‘Jones E’ drives the Car with reg # ğEPZN’.
The person with name ‘Jones E’ drives the Car with reg # ğeAAQ’.

The C.S schema corresponding to (fig. 3) which will be generated by the tool is given in fig. 4.

Fig. 4 “Conceptual Diagram Schema”

The result of stage 4 is expressed by a bar placed above the idea. This comes from the fact that in the two columns “Person” and “Cars” (fig. 3), there are instances which can be repeated. It means that there is no unicity constraint for the two roles. The user will have the choice for specifying the statical aspect of his system: he can either use the MI module and then be assisted and guided during the whole elaboration phase of C.S, or draw up manually his C.S and input it by using the M2 module.

The M2 Module: It is a graphical interface that aims to enter a NIAM schema with the possibility to update it, to analyse its coherence with respect to NIAM rules, and provide the Relational Schema (RS) corresponding to the 6 Normal Form (application of the grouping algorithm) [6]. Note that it is possible to transform the C.S designed by the MI module to the R.S corresponding to the 6
Normal form by using the grouping algorithm of the M\[2] module. Figure 5 can be entered and analysed by the M\[2] Module. The following example is taken from the library case studied in [Ce].

In this figure, the NOLOT object “Book” is referenced by its ISBN number, i.e. its copy is a document identified by an inventory number. The latter is assigned after a command which is identified by a command number. It is the book which is the object (purpose) of a command and it is delivered at a certain date.

The application of the grouping algorithm on the NIAM schema above gives the following R.S in 5th Normal Form (without vanishing values):

\[
\text{Document } (N^e \text{inventory}, N^e \text{command, ISBN}) \\
\text{Book } (\text{ISBN}) \\
\text{Command } (N^e \text{command ISBN}) \\
\text{Delivered Command } (N^e \text{command date of deliver}) \\
\text{Calendar } (\text{date})
\]

Note that it is not always easy to design a C.S with binary...
relationships only. In [CÉ], Nijssen and Halpin underline this problem and show that it is always possible to transform n-airy relationships into binary ones, by using objectified relationships or embedded idea types.

The Mǐ₃ Module: This module enables the user to validate easily his schema with the inputs of the MI module (input form and output report). It consists in paraphrasing the C.S, that is expressing, by a set of sentences, the conceptual definition of the information system such as it has been constructed by this module.

This process starts from the representation of schema saved in the data base under the form of LOT, NOLOT, BRIDGE and IDEA, then to produce a text in natural language composed of simple understandable sentences. So for the explanation on an object, we indicate: the type to which it belongs: this object “is-a”….

Example: Book is-a NOLOT

Explanation types:

The explanation types can be of an elementary form or a composed one.

(C). An elementary form is defined by two NOLOTs linked by an idea and, possibly, by its cardinalities.

Providing an elementary explanation consists in clarifying the first role between the two NOLOTs and the second role (inverse role)

A <NOLOT₁ > <Role₁> <NOLOT₂>
A <NOLOT₂ > <Role₂> <NOLOT₁>

Example: The elementary explanation corresponds to the idea between Book and Command of the fig. D

The explanation for the first role is: A Book is the purpose of Command.

The explanation for the second role is: A Command turns on Book.

(Ç). The composed explanation is the group formed by an object and all the other objects directly linked to it by associations (Bridge/Idea).

The composed explanation consists of expressing, for a given
NOLOT, its hierarchy and structure, according to a strategy. The adopted strategy allows to determine, starting from a NOLOT, its identifier, the Bridges and the Ideas to which it relates.

In this way, the composed explanation for the object Book of the figure 5 will be:

- Book is a NOLOT
- Book is identified by ISBN
- Book has Document as a copy
- Document is a copy of a single Book
- each Document is a copy of a single Book
- Book is the purpose of Command each Command turns on a single Book

The statical aspect (DATA) is taken care of by the three modules M1,M2 and M3 by using the NIAM methodology. The inconvenient of this method is that it is not able to take into account the dynamical aspect (Treatment & Behavior) in the I.S. To treat this aspect, we have chosen the JSD method (Jackson System Development). NIAM is specialized in the statical aspect and JSD in the dynamical one, in addition, there is a duality between the two types of specification. The idea of this combination of NIAM and JSD has been prompted by the works of H. Habrias and Barbier [11].

The Md'Module: The dynamical aspect (treatment+behavior) is represented by this module (JSD Graphical interface) and aims to enter a C.S (based upon JSD) in an interactive or a graphical manner with the possibility of update it, to analyze its coherence with respect to the rules of JSD and, to provide the interpretations of specified schemas (which are structured texts and regular expressions that provide more semantics) [3]. The specification is then completed by that of the data of the “book” object (figure 5) by the specification of treatments like the following figure 6.

![Diagram of Book and related treatments](image-url)
The semantics of the “Book” entity is completely described. The book referenced by its ISBN number, ordered by the library and received after delivery at a certain date, will be catalogued and given an inventory number. It can then be borrowed several times and, finally, archived or sold. The figure presents a S.S.D example (System Specification Diagram) or the network communication in this application. This S.S.D can be generated by our system.

In the above figure, there are two principal processes (objects or entities): Member and Book. The process red-list is a function which is triggered during the week-end (this explains in the figure the presence of the temporal marker “clock” called in JSD “TMG”: Time Grain Marker) in order to create the list of members that will be convoked or sanctioned. To do this, this process consult the books’ state and gives the result (the list) by means of the DS data stream.

This example shows that JSD takes into account the adjunction of time and synchronization to the order of the events. The JSD techniques that take care of these notions are well detailed through different examples in [31]. To concretly see the behavior of the system (the change from one state to another depending on the events of the external world), a simulation (graphical animation) of this behavior is made possible by the M module. Having noticed here the lack of formalism of the JSD method (which explains the absence of rigorous validation of the specification), we had to study other formalisms (formal approaches) and to choose one which will be coherent with the JSD method. In other words, our search was oriented in such a way that this formalism will be the theoretical support of this method.

We finally opted for the Rewrite Logic (R.L) because this logic and
the JSD method are both concerned with systems having a time-dependent behavior, and treat parallelism and concurrency notions. Furthermore, the two approaches focus more on the dynamic of the system (treatment and behavior) than its statical part (data). This explains the link between the M₄ and M₅ modules. Our objective was to keep the advantages of the two types of specification (informal and formal). Notice that this combination of JSD with the R.L for the specification of I.S will be developed in another communication.

**The M₅ Module:** “Functional validation of the specification using the R.L”

**Rewrite Logic:** The concepts of Rewrite Logic constitutes the necessary formal framework for the specification of systems, the study of their behavior, and their validation. In this theory, the object-oriented concurrent treatment correspond to the notion of deduction logic. It is realized by concurrent rewriting. The rules allow to reason about the state changes in a system, and help to deduce valid conclusions on its evolution from a certain number of type-changes, thanks to the rules R [⁻, [⁵] & [⁶].

The general form of rewriting rules is given in figure 8:

\[ M₁ … Mₙ \rightarrow O₁' C₁/ \text{list} \leftarrow Aₜₙ \]
\[ \rightarrow O₁'' C''/ \text{list} \leftarrow Aₜₙ'' \]
\[ O₁'' C''/ \text{list} \leftarrow Aₜₙ'' \]
\[ Q₁ Q₂/ \text{list} \leftarrow Aₜₙ \]
\[ M₁ … Mₙ [t] \]

**Fig. 8: General Form of rewriting rules**

The rules of this kind express the occurrence of a communication event in which n messages and m different objects are concerned. The effect of such event is:

- M₁ … Mₙ messages disappear
- The objects state O₁, …, Oₙ is modified
- The objects which appear only in the left part of the rule disappear
- New objects Q₁, …, Qₚ are created
- New messages M₁, Mₙ, …, Mₙ are emitted.

[t] introduces the service time notion. Each rewriting rule can be assigned a time t which corresponds to a temporal constraint.
The description of the dynamical behavior of the system simply consists in the instantiation of the rules general form. Under this form, the rules indicate that the system changes from the state which corresponds to the configuration represented by the left term of the rule, to another state corresponding to the configuration represented by the right term of the same rule.

In these conditions, the rewriting system must perform the transformations described by the rule in a time interval corresponding to \( t \).

This could be operated (exploited) either for expressing a service time for the needs of a simulation, or for taking into account the time aspect in real-time applications. Note that figure 8 is a very high level declarative form of the cooperation and synchronization between objects \([d]\) & \([c]\).

This module allows to check in a strict way (automatically) the specification done by the Md module. To manage this, this specification has to be, transformed in the formalism of the rewriting logic in order to be entered, lexically and syntactically, then semantically corrected, and later validated (by checking the properties of the specification: completeness, consistency...) by this Md module. Its role is to satisfy and to refine more and more the third objective of the tool (see \( d'c \)). It means that, in addition to the validation done by the Md module (Text Generation) concerning the statical aspect and the one done by the Md module (JSD Graphical Interface) concerning the dynamical aspect (by simulation), the Md module validate, in a formal and strict manner, the specification of the system as well as its behavior in time.

In other terms, this module is a sub-language of formal specification (and validation) based on the R.L. It concerns the functional validation of the specification that shows the behavior of the system (its evolution) in time in an explicit manner.

Our idea is simple and comes from the fact that, in Rewrite Logic the system is seen as a configuration of objects and messages. Therefore, if the behaviors of the different objects (defined in JSD) and the messages exchanged between them are mastered then the rules defining the behavior of the system in R.L could be deducted. Thus, the formalism of the latter could be taken advantage of. Figure \( c'c \) presents a concrete example, concerning the set of rules deducted by the specification (the library case) performed by the Md module (fig.\( E \)).

\[
\text{rule1: (borrow_book_lb_byIM) \rightarrow bk.library/id-}\]

\( c'D \)
CONCLUSION

The characteristics and the functionalities of a case tool for Information System Design have been introduced, they aim, on one side, to take into account the static and dynamical aspects, and to provide the user with products that are readable and easily understood (C.S) and, on the other side, to help the designer to validate the specification before carrying on towards the implementation.

To manage this, a combination of the NIAM and JSD methods and
the Rewriting Logic formalism seems necessary. This system has the advantage to use facilities such as:
- Starting by specifying the statical aspect, then the dynamical aspect, or conversely
- Having both specifications types, informal and formal, in the same system
- Specifying and validating, i.e. the user can see in a graphical way, the results of this specification by executing it (simulation).

The use of the help system may be partial (using only some modules of the system) or total (using all the modules). In addition, the richness of the system (that could handle the Data, Treatment and Behavior aspects, the Time and Synchronization notions,...etc...) allows the extension of its application domain to management, real-time applications,...etc..

The system is not completely achieved and some perspectives can be mentioned, like the passage from a text in the natural language describing the specification to the corresponding C.S (or within the limits to tables corresponding to the inputs of the C.S.D.P (MI Module)), the passage from the specification done in Rewrite Logic to its code in a target programming language (C++, ADA, .....,).

Finally, let us mention that a work on the formalization and the automatization of the transition from JSD to the Rewrite Logic is being done and will be the object of a future communication.

REFERENCES


ATTOUI, A. “The use of the Rewrite Logic for the specification and


