

# Noé: Expert system for technical inspection of waterproofing on flat roofs

Autor(en): **Poyet, Patrice / Delcambre, Bertrand**

Objektyp: **Article**

Zeitschrift: **IABSE reports = Rapports AIPC = IVBH Berichte**

Band (Jahr): **58 (1989)**

PDF erstellt am: **11.09.2024**

Persistenter Link: <https://doi.org/10.5169/seals-44905>

## **Nutzungsbedingungen**

Die ETH-Bibliothek ist Anbieterin der digitalisierten Zeitschriften. Sie besitzt keine Urheberrechte an den Inhalten der Zeitschriften. Die Rechte liegen in der Regel bei den Herausgebern.

Die auf der Plattform e-periodica veröffentlichten Dokumente stehen für nicht-kommerzielle Zwecke in Lehre und Forschung sowie für die private Nutzung frei zur Verfügung. Einzelne Dateien oder Ausdrucke aus diesem Angebot können zusammen mit diesen Nutzungsbedingungen und den korrekten Herkunftsbezeichnungen weitergegeben werden.

Das Veröffentlichen von Bildern in Print- und Online-Publikationen ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Die systematische Speicherung von Teilen des elektronischen Angebots auf anderen Servern bedarf ebenfalls des schriftlichen Einverständnisses der Rechteinhaber.

## **Haftungsausschluss**

Alle Angaben erfolgen ohne Gewähr für Vollständigkeit oder Richtigkeit. Es wird keine Haftung übernommen für Schäden durch die Verwendung von Informationen aus diesem Online-Angebot oder durch das Fehlen von Informationen. Dies gilt auch für Inhalte Dritter, die über dieses Angebot zugänglich sind.

## **Noé: Expert System for Technical Inspection of Waterproofing on Flat Roofs**

Noé: système expert de contrôle technique de l'étanchéité des toitures terrasses

Noe, ein Expertensystem für die Kontrolle der Wasserdichtigkeit von Terrassendächern

### **Patrice POYET**

Docteur ès Sciences  
CSTB  
Valbonne, France

### **Bertrand DELCAMBRE**

Ingénieur  
CSTB  
Valbonne, France

Patrice Poyet obtained his Doctorate degree (D.Sc.) at University of Nice, prepared at INRIA Sophia Antipolis, France. He was involved in the development of computer systems for mining exploration, using both numerical processing and symbolic modelling. His main subject of interest is artificial intelligence, and for three years he was head of the simulation department at ILOG S.A., a daughter company of INRIA research center for Building (CSTB) at Sophia Antipolis, and is currently involved in the development of advanced information systems.

Bertrand Delcambre is Engineer of the Ecole Polytechnique and Engineer of the Ecole des Ponts et Chaussées, and has been involved in research activities at the French Scientific and Technical Center for Building for ten years, including building energy analysis, solar energy storage, robotics, computer assisted design, expert systems. He is currently head of the Computer Science and Building Service, grouping three Divisions and about twenty five persons.

### **SUMMARY**

Within the framework of feasibility study the Centre Scientifique et Technique du Bâtiment (CSTB), has jointed together with a technical inspection organisation the Centre d'Etude Technique des Apaves, to elaborate an expert system prototype to simulate technical inspection work applied to waterproofing work on flat roofs. Stimulated by the results of this application, we can already envisage the interest for an expert system shell specifically for technical inspection, and also we look at the way to use it for teaching and deisigning.

### **RESUME**

Dans le cadre d'une étude de faisabilité le Centre Scientifique et Technique du Bâtiment (CSTB), s'est associé à un organisme de contrôle, le Centre d'Etude Technique National des Apave, pour élaborer une inaquette de système expert de contrôle technique relative au sous-domaine de l'étanchéité des toitures terrasses. Les résultats encourageants de cette réalisation nous permettent d'imaginer plus globalement l'intérêt d'un générateur de système expert d'assistance au contrôle technique, ainsi que l'extension du système vers des utilisations plus variées, telles que l'aide à la formation des contrôleurs techniques ou encore à la conception des ouvrages d'étanchéité.

### **ZUSAMMENFASSUNG**

Diese Studie hat den Zweck, die Machbarkeit eines Prototyp-Expertensystems zur technischen Prüfung der Dachterrassenwasserdichtigkeit abzuschätzen. Die Originalität eines solchen Systems besteht darin, dass sie den Vorgang des technischen Kontrolleurs sowohl in der Abwicklung der Kontrolloperation als auch in ihrem eigentlichen Wesen wiedergibt. Die ermutigenden Resultate dieser Forschung erlauben es, weitere Werkzeuge für technische Kontrollen für Lehre und Entwurf zu verwirklichen.



## 1. INTRODUCTION

### 1.1 Technical control, what for ?

Technical Control is a human activity intended to verify that a project, either at a preliminary stage, when implemented, or even at completion time conforms to a set of well defined specifications. This brief definition already illustrates some of the most challenging properties that an expert system should account for in order to reproduce the step of controller.

First of all, the process implies a peculiar kind of logic seldom used by most programming habits, referred as temporal logic [1], [2], [3], devoted to the modelling of concurrent or successive interdependent events, in so far as the process spans over time and the project must be controlled during incremental phases leading to the final configuration.

The second difficulty is tied to the control activity in itself, as it can be considered as the reciprocal function of design, an area where a lot of work has instead been accomplished.

From the designer viewpoint, getting a suitable response to a problem can be described as a sequence of actions that aims to transform a set of initial specifications into a valuable assembly of objects, recognized as a solution [4], [5]. On the other hand, the technical control services given by experts have been less studied by scientists, mainly involved in this research area with plants or industrial control process.

### 1.2 Legal context

Thus, the field covered by this paper, technical control carried out by human specialists and viewed as an intellectual practice based on experience, is still rather an unexplored area. In this domain, the controller takes the result of a design, an intended use to be reached, and tries to determine if the overall set of constraints has been respected following the inverse path of the designer. The model developed by [6], states that constraints can be classified according to three main characteristics: the properties of the constraint generator (introduced either by the conception process, the customer, the end-user, or by legal requirements), a domain (among internal or external including the environment of project), and the underlying functions (practical concerns, functional, formal or symbolic).

This project deals with the stronger set of constraints aforementioned, as we wish to model the state of the art rules encoded within different unified codes of practice, considered as the regular documentation to be used. Nevertheless, even for this restricted context due to coercive legal clauses, many alternatives can be encountered by the designer, and an automated rule-based control process involving pragmatic knowledge to assess the changing reliability over time of an evolving proposal is a difficult task.

The Noé expert system is a practical software dedicated to a narrow field within the wide area of technical control, and aims to account for the technical inspection of waterproofing on flat roofs, at the different epochs of the project's life. This selection was due to the high contribution of the various waterproofing techniques to the global percentage of observed building damages.

## 2. COGNITIVE APPROACH

### 2.1 Task analysis

The first objective in developing such a system, was to observe the behaviour of the human controller, and to build a model of the tasks to be carried out so as to account for the intended activity. This analysis had to split the controller's work into separate phases, from a temporal viewpoint, and to associate for each of them a semantic model of the goals pursued.

This was easily achieved, in so far as the controller primarily acts as soon as the first drafts can be obtained for the project, then reviews the proposals made by selected firms to implement some solutions among a wider initial set, and finally inspects the working site at the end of the installation of the waterproofing layers and of the related protections. This incremental involvement of the control process gives the general framework to be modelled by the software, and the system should of course reproduce such a temporal logic, as the information flows obey this sequence.

At the beginning of the project, the framework is still ill defined, and many solutions can be accepted according to the intended use; this step will be referred as "design level". Then, depending on the successive refinements of the design, and of the products retained for the effective installation, the controller verifies that responses made by firms are correct at the proposal level; this will be curiously referred to by specialists as the "execution level", even if the effective installation of the waterproofing system is only done later on. Finally, when the system is installed on the work site, leading to an "in situ" control, we talk of the "installation level". This terminology is important, in that it will be used throughout this paper, and as it helps to give a frame to the software functionalities.

Each of these separate phases leads to a specific analysis, involving dedicated knowledge bases, designed to produce successive regular reports, such as the preliminary report, intermediate report, end of phase report. For each of them, careful observations are produced and should be interpreted at three different semantic levels. The observations are defined as follows: prescriptions correspond to compulsory alterations to the intended design and represent the most coercive action given by the controller as they arise when a regular guiding rule is violated, recommendations offer well known solutions that should be substituted to an odd but not irregular design, and finally advice has an indicative meaning coming from well tried concepts. These reports are produced at three monthly intervals and in the meantime the project evolves from a pure design phase to an operational waterproofing system. The general activity of the system, and the different phases previously noticed can be summarized by Fig 1.:

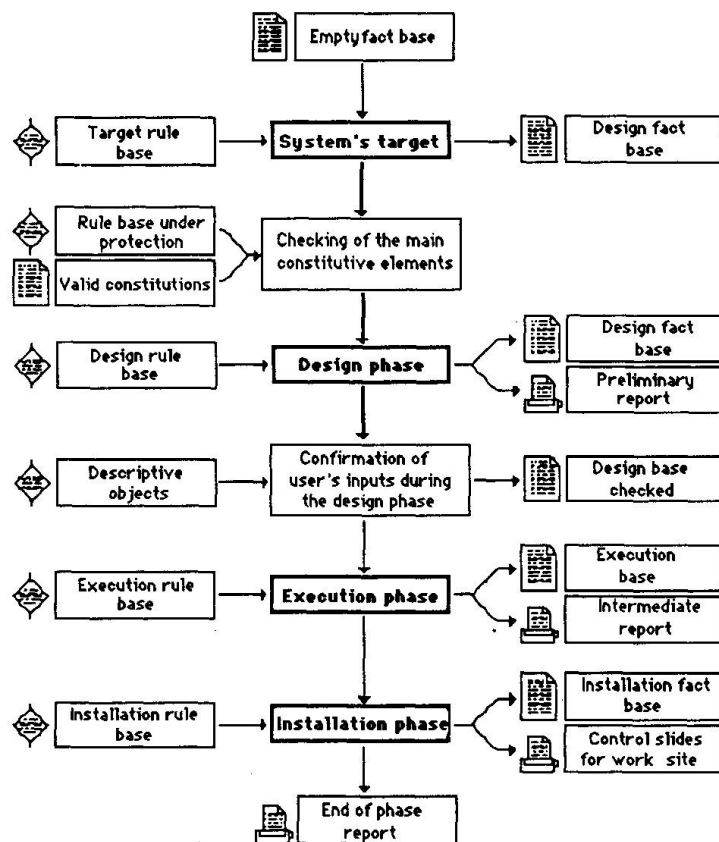


Fig. 1: The general task organisation of the Noé system.



In Fig. 1, we see the three main phases of the control cycle of the software (design, execution and installation phases), and moreover an initial plan to determine if the problem described by the user is within the system's scope, (i.e. the competence domain of the program). If this assertion is verified, the system controls the correctness of the submitted preliminary design (no unrecoverable errors in a compiling analogy), and if it is correct enables the scheduling of the real design phase (considered from the controller's viewpoint and not from the designer's), for which prescriptions, recommendations, and advice are given. A preliminary report is then produced.

As some delay can occur before execution takes place (i.e. analysis of the responses transmitted by firms), a general checking of the overall properties previously determined is performed. If inconsistencies arise, the culprits are suppressed from the fact base, and the control is transferred to the first rule base (i.e. target rule base) to reconsider the problem. When the design can be submitted to the execution module (i.e. no subsequent noticeable alteration to the initial ordering was observed), the execution rule base is activated and leads to an intermediate report, including more detailed instructions than the preceding ones but dealing with the same scope.

Finally, when the waterproof coverings are installed, the system produces checking forms to be used on the work site to ensure the final checking of the waterproofing system and to guarantee its conformity to the unified codes of practice.

## 2.2 Consistency checking

Some difficulties arise from these overlapping phases, and we modelled the induced consequences that have to be taken into account when some important alteration occurs to the original design. A semantic network of linked objects is managed by Noé so as to propagate the effects of changes that imply modifications or suppressions of the consequents further altered by reasoning. This problem is rather similar to truth maintenance functions [7], [8], [9], [10], where the origin of inconsistencies comes from the temporal evolution of the environment, some properties being transformed when controlled by the expert. This topic is encompassed by Mc Allaster [8], and temporal consistency has long been a privileged area for Truth Maintenance Systems (TMS). We developed a simplistic framework, where any significant change is propagated to the consequents and where dependent objects from the modified node (a node refers to any transition for which a value is either asked or computed), are suppressed from the environment and should be recomputed. This is quite a radical approach, but it ensures the integrity of the database even if it is quite resource consuming.

Sophisticated models could be introduced such as a complete TMS, or a constraint-based language [11], but for the first implementation of the system we followed a pragmatic approach, for which each object is tied to its consequents thanks to semantic links (recognized and defined by the designers of the application), and any alteration of fundamental properties of antecedents induces the removal of the consequents and enables the triggering of previous rules.

The following network reproduces a very small subset of the overall semantic links modelled by the Noé expert system, and illustrates in a schematic way some interdependencies between the considered objects. The time variable does not appear in this view, but it intervenes for each one of the phases previously described (and especially during the design level), and between two successive steps as some alteration to a predefined organisation may occur. An historical recording of the order in which the inferences have been accomplished is stored (without possible concurrent paths as in [12], [13], [14], [15], [16], [17], [18], [19], and is used to remove the consequents when a support node has been altered during the checking phase (i.e. either the designer suppressed on its own a property or a modification suggested by the controller induced the transformation and the related coherence processes).

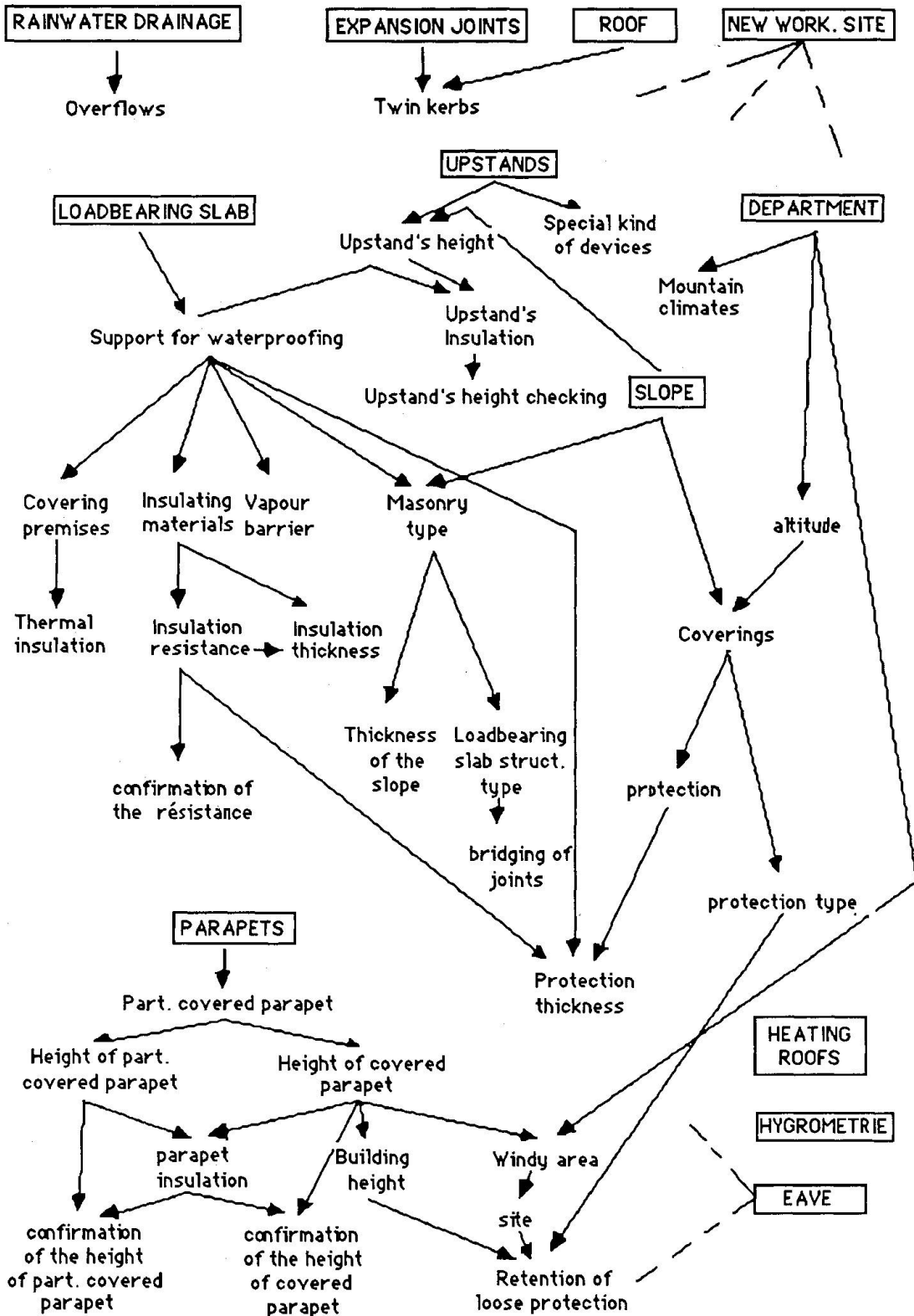


Fig. 2: A subset of the overall semantic network of interdependent objects.

The links are explored and the creation time of the encountered objects is tested to take legitimately the destructive decisions. This principle is illustrated in Fig. 2, support nodes appearing in the higher part of the graph, and consequent nodes being linked with arrows. The real network involves one



hundred and sixteen objects connected thanks to the link property of the object model we use. In fact it will be noticed that among this set of objects, some could be better considered as slots of higher level frames, that we are currently beginning to implement for Noé.

In section 3, the choices made for the current implementation will be justified, in relation to the supposed end-users we aimed to reach, and their potential hardware (mainly restricted machines with approximately one mega byte).

### 3. SYSTEM ARCHITECTURE

#### 3.1 Introduction

A severe restriction to the design of the system architecture was introduced by the kind of hardware on which it should be used. The Machintosh was initially chosen as a valuable target for the system as it is widespread within the waterproofing corporation (i.e. firms, architects, manufacturers). But it should be noticed that for basic machines, a memory of one or even two megabytes is a narrow space to implement a complex system. So we used basic Lisp mechanisms, a primitive object layer not too resource consuming, and the powerful intrinsic graphical functions of the Machintosh based on the I/O Trap of the ROM.

Of course, this framework is evolving so quickly, that our focus is now moving toward virtual concepts, including virtual machines (i.e. the LLM3 virtual machine of Le-Lisp is a good example of this concept), virtual graphical solutions (the virtual bitmap of Le-Lisp responds to such an objective) [20], fully portable interface builders from one hardware to another, using different windowing systems (i.e. the Aïda toolbox is such a kind of powerful image manager) [21], and abstract software components (i.e. only defined by an abstract structure and their associated behaviours).

The implementation strategy retained nevertheless enables us to broadcast the system to a wider public, as it does not require any specific machine or advanced underlying software; Noé is self-contained.

#### 3.2 Reasoning model

The reasoning model of Noé is very straightforward and implements a forward chaining strategy directed by the facts to be checked. For each reasoning phase previously described, a specific rule base is triggered, the one in charge of this verification level, and is processed in a standard way by the inference engine.

The principle used is that for each base submitted, the engine scans the sequence of rules and determines if at least one of them is to be fired. If so, the rulebase will be considered again (i.e. some other rules can have their status changed by the inference results added). The inference cycle is stopped, for the considered rulebase, either when no rule is left in the agenda, or when an explicit fork is made to another rulebase. The inference engine in itself is a simple function taking as an argument a rulebase. This enables the system to chain the rulebases, as the action of any rule can schedule in a recursive way the next rulebase to be examined in the new context recognized by the premisses.

#### 3.3 Knowledge representation

The knowledge managed by the system belongs to three distinct types: objects, rules, and facts. This is a conventional way of representing knowledge in most expert systems, and we did not develop any original feature in that respect. In fact, we tried to find economic memory allocation policies (§3.1.), using rather low level primitives, even if the limitations induced sometimes could be deemed as restrictive.

### 3.3.1 Objects

Objects, better entitled entities, are implemented at a very low level, using directly Lisp symbols and attaching them properties thanks to the basic property list mechanisms (P-Lists). It ensures a very efficient functioning, as it corresponds to low level *wired* Lisp faculties, and enables a set of data to be packaged within a memory structure. It lacks some obvious features of real object based environments [22], [23], [24], but was deemed satisfactory for the specific goals encompassed by the prototype. Nevertheless the road towards true object-orientedness is long, and the system suffers in some manner from the current implementation. An "object" is given hereunder, and the link property appears including pointers to the connected objects:

```
(PLIST 'slope '(title "The slope is "  
  quest "Is the slope ? "  
  type (code 1 2 3 4 5)  
  1 "flat roof"  
  2 "1%"  
  3 "3%"  
  4 "5%"  
  5 "more"  
  order ("flat roof" "1%" "3%" "5%" "more")  
  asked ()  
  to-be-checked t  
  link (covering  
    upstands-height  
    waterproofing-composition)))
```

Some properties are used to store useful data for the interface management, such as "title" enabling to produce intelligible sentence frames filled with the value of the referred entity (to give an understandable insight to the factbase or to ease the production of readable reports), "quest" is a string used to ask a question to the user, to get the entity's value during reasoning, the "type" can be *entity* leading to boolean choices, *value* enabling numerical values such as integers or reals to be stored within the property, *interval* leading to express constraints on the acceptable domain of the slot, *code* or *constrained-code* for which a set of possible values is computed and prompted thanks to a graphical device offering a selection among possible values with checkboxes, the "help" slot is a text of aid, and link enables the system to keep trace of structured representations as many of Noe's entities are inherently hierarchically organized (Fig. 3), permitting coherency checking.

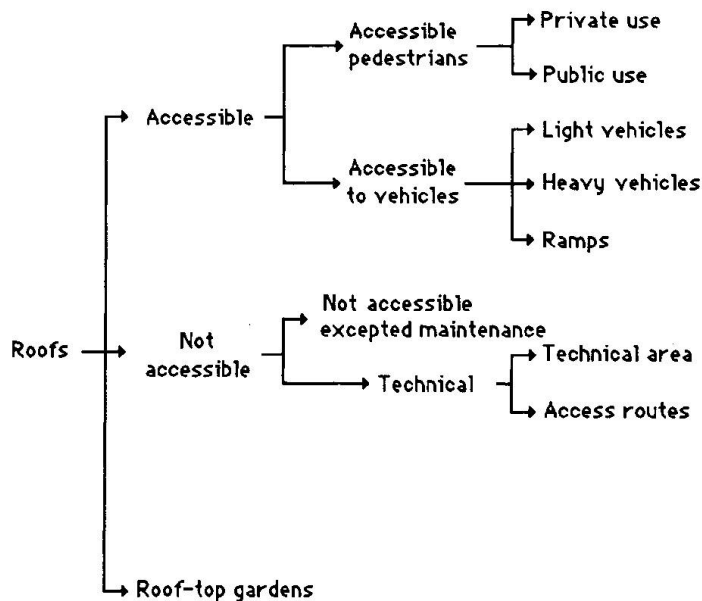


Fig. 3: Roof classification according to intended use.





### 3.3.2 Rules

Rules obey propositional logic, and enable the user to check entities slots' values, evaluating the Lisp code corresponding to the premises, and to modify the database in the same way by the triggering of the code attached to the actions. They offer basic functions, and do not require further explanations. An example is given bellow (as it appears when listed by the system):

```

IF      reverse-insulation = true
      and roof = roof-top-garden
      and ask the value of protection
      and protection <> foreseen

THEN    Modify : protection = foreseen
      and Modify : protection-type = Applied
      and Modify : applied-protection-type = draining-layer

```

The original Lisp form is the following:

```

( ( (test 'reverse-insulation ':= "true")
    (test 'roof ':= "roof-top-garden")
    (ask 'protection)
    (test 'protection ':<> "foreseen") )
  ;
  ( (replace3 'bfl (list ':= "foreseen") 'protection)
    (replace3 'bfl (list ':= "applied") 'type-protection)
    (replace3 'bfl (list ':= "draining-layer")
              'applied-protection-type) )
  404)

```

This Lisp low level representation induces some limitations, but they were not problematic within the scope of a prototype, more especially as the software had to run on machines with limited resources. Moreover it gives the full flavor of Lisp as any kind of Lisp call can be merged within the rules to implement specific semantic purposes. Rule bases are just a superset, for which rules of the same topic are grouped within the same higher level entity.

### 3.3.3. Facts

A fact is a symbol to which we attached a list of properties with the following form, using the same mechanism as previously described:

((operator-1 value-1) (operator-2 value-2) ... (operator-n value-n))

and a factbase is a list (used as P-list to be coherent with the rest of the system), where two sorts of elementary data are stored, the one that was asked of the user and the one deduced by the expert system. So as to keep track of the modifications endured by the fact base, a global variable is used as a pointer to a structure enabling the transformations to be recorded.

## 3.4. Main functions

Main functions belong to two separate kinds, aiming either to address entities within the premises (ask the value, test the value of an entity, is-it-known ? or unknown ?), or to alter them by actions (add a new fact, suppress some value, substitute some value).

### 3.4.1 Functions to address entities

- (ASK <entity>): The ASK function questions the user about an entity's value, and stores the given result as a fact within the fact base. If ASK is solicited with an argument already being filled with a value, this one is automatically returned without disturbing the user.

- (TEST <entity> <operator> <value>): This function enables the value of any entity to be tested within the fact base, and returns t or nil depending whether the predicate is verified or not. Example:

```
(TEST 'day ':<> "Monday")
```

- (KNOWN <entity>) and (UNKNOWN <entity>): These functions enable the user to determine if an entity is known or unknown within the fact base, i.e. if a value has been attached to it.

### 3.4.2 Functions to alter entities

- (PUT3 <fact base> <operator value> <entity>): This primitive enables a new fact to be added to the factbase. If no value was previously attached to the entity, this one is inserted, otherwise a new cons made of (operator value) is pushed inside the existing stack. Example:

```
(PUT3 'bf1 (LIST ' := "Dimanche") 'Jour)
```

- (REM3 <entity>): Allows the suppression of the stack of values labelling an entity stored in the fact base. The consequent nodes are not affected by such a removal.

- (REPLACE3 <fact base> <operator value> <entity>): This primitive is used to substitute a new fact to a list of value labelling an entity, using in fact (REM3 <entity>) then (PUT3 <fact base> <operator value> <entity>). Example :

```
(PUT3 'bf1 (LIST ' := "Lundi") 'Jour)
```

Thanks to this small set of primitives, and to application dependent functions, it is possible to express nearly watherver the programer wishes for the kind of requirements encountered.

### 3.5 System Interface

The aim we had was to take advantage of the powerful graphical possibilities of the target machine to develop a very friendly interface, either based on the Macintosh dialogs or on resources. A Noé session relies on the interaction with graphical objects enabling the user either to get a booleen answer to an accurate question, to acquire text from line editors, to give an alert thanks to a message to be printed (Fig. 4), or to build complex pictures offering a set of possible selections thanks to checkboxes associated to icons for example as on (Fig. 5).

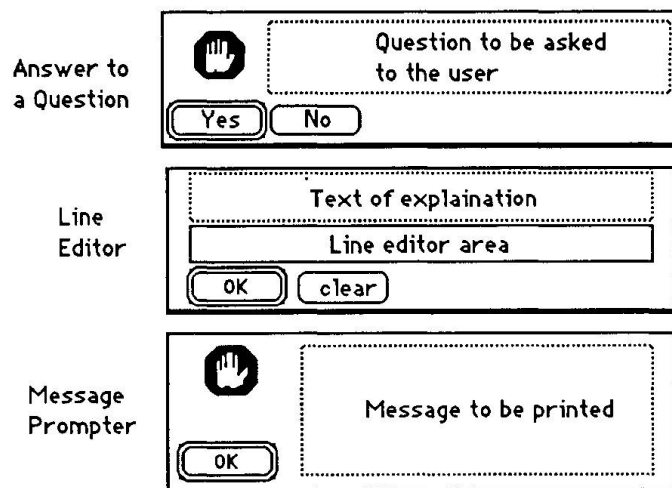
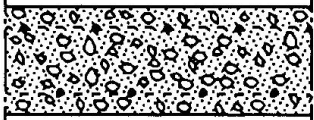



Fig 4: Some graphical objects used to manage user interaction.


Hereunder, Fig. 5 is an example of such concepts as it enables to select the type of the loadbearing slab structure, in an easier way than a textual description:




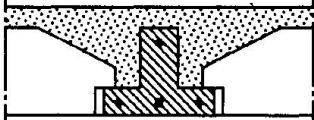
**Input**

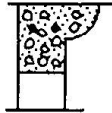
Type A 

Type B 

Type C 

Type D 

or 

 Across A-A

Is the loadbearing slab structure of type ?

A  
 B  
 C  
 D

Fig. 5: Complex graphical objects enabling straightforward intercation.

The corresponding textual description could be:

- Type A: Loadbearing members of which at least the upper part of the supporting section is constructed of reinforced concrete cast-in-situ continuously over the entire surface;
- Type B: Loadbearing members consisting of joined prefabricated reinforced or prestressed concrete members made rigid by reinforcement embedded in cast-in-situ connecting concrete;
- Type C: Loadbearing members consisting of joined prefabricated members of different materials made rigid by cast-in-situ blocks of concrete and/or transverse ties of reinforced concrete;
- Type D: Loadbearing members constructed using joined prefabricated reinforced concrete or prestressed concrete members made rigid by unreinforced concrete connections.

These data can be useful as the meaning of the results we are going to present now (section 4) are very dependent from the type of the loadbearing slabs encountered.

## 4. RESULTS

### 4.1 Control results during the design phase

When we presented the general organisation of the system (Fig 1), and made the task analysis (§2.1), we asserted that three kinds of data were produced to conform to the human controller step. Moreover some rules aim to schedule verifications and activate computing instead of the controller, such as the computing of the thermal insulation resulting from the composition and the thickness of the insulating

material. All these data make up the preliminary report, corresponding to the pre-project analysis. We give an example of a design level rule:

```
IF    slope = no
      and ask the value of upstand
      and upstand = foreseen
      and ask the value of upstand-height
      and upstand-height < 15

THEN  add : prescription "The height of upstand covered with
      waterproofing must be such that the minimum height of the waterproof
      skirting at any point be at least 15 cm when the roof is flat"
      and Modify : upstand-height = 15
```

Then the conclusions of the system are available:

*Prescriptions:*

Heavy-duty protection is obligatory for multilayer coverings,

There is to be an arrangement above the waterproof skirting to divert the water running off members of the main structure which are above it so as to prevent water from getting behind the waterproof covering,

If there is only one rainwater downpipe, an overflow device must be foreseen,

*Recommendations:*

It is recommended to have no slope within eaves,

Eaves should be avoided,

*Advice:*

It is better to foresee a hard protection for skirtings in case of hard protection of the main roof.

#### 4.2 Control results during the execution phase

We remember the reader, that the execution phase (from the controller viewpoint), takes place just before that the effective installation of the waterproofing be done. We encounter the same sort of rules but they lead to more accurate controls. The prescriptions produced at the end of this phase are really more precise like the following:

*Prescriptions:*

The skirting is to consist of: a cold priming coat, a layer of hot-applied coating to the right of the reinforcing angle piece, a reinforcing angle piece 0.20 m broad with equal limbs, of type 40 reinforced bitumen, cloth reinforcement, welded or stuck, a layer of hot applied coating, a reinforced bitumen type 40 TV with incorporated metal foil protection, with a toe of 0.15 m on horizontal part, welded,

The bridging of the joints in case of loadbearing slab of type D must be foreseen during the installation of the vapour barrier, and be at least of 0.20 m large.

The vapour barrier system should be made of a layer of cold priming coat, a layer of hot-applied coating, a bituminous felt of type 36 S (VV-HR), ended with a layer of hot-applied coating,

Any point on a flat roof must be within 30 m of the collecting device (eave or trough gutter) or rainwater outlet. The maximum distance between two downpipes from an eave of trough gutter is 30m),

etc...

#### 4.2 Control results during the installation phase

The aim of this step is to produce the final report. In order that no remaining data should be left with an unknown status, checking forms are produced by the system, and enable the operator to accomplish on site checking of the corresponding characteristics. First of all, a summary of the overall properties of the project is given. It could appear as the following report:

The roof is not accessible,

There is no slope,

The waterproof covering is a multilayer,

There are loadbearing insulating board supports,



A vapour barrier is foreseen,  
 A protection is foreseen,  
 The insulation is 12 cm thick,  
 Expansion joints are not foreseen,

etc...

Then verifications are suggested by the software and lead to on site checking. This phase can be repeated several times and indications have the following form:

*Verifications:*

Rainwater outlets are made up of two parts: the flashing and the spigot, which are assembled by welding or by any other method providing a permanent watertight joint. It should be verified that the distance between the external edge of the outlet hole and the outside edge of the plate is not less than 0.12 meter.,

etc...

## 5. CONCLUSIONS

Noé is an operational prototype, reproducing the step followed by a technical controller in the area of waterproofing work on flat roofs. The system's behaviour is mapped on the observed state of the art habits of human experts, and implements a succession of control procedures, being more and more refined, so as to finally produce an end of phase report when on-site checkings are successful. The software takes advantage of the powerful capabilities of the graphical toolbox of the Macintosh, so as to offer a high level interface, enabling a wide public to benefit from it. Specific functions had to be defined to handle the temporal consequences of modifications to the initial design, and the system is able to deal with real evolving projects. Nevertheless, improvements will be made in many respects, to provide self-explanation possibilities, to handle the regulation documents accurately, and to virtualize many concepts, allowing a machine independent system to be obtained.

## REFERENCES

1. Allen, J. F., 1981. An interval-based representation of temporal knowledge. Proc. 7th Intern. Joint Conf. on Artificial Intelligence, Vancouver, Canada, pp 221-226.
2. Turner R., 1986. Logiques pour l'Intelligence Artificielle. Masson Eds., Paris, 120 p.
3. Poyet, P., 1987. La structure de contrôle dans les systèmes experts de simulation. 6ème Congrès Reconnaissance des Formes et Intelligence Artificielle de l'AFCEP, pp. 723-738.
4. Montalban, M., 1987. Prise en compte de spécifications en ingénierie, application aux systèmes experts de conception. Thèse de Doctorat en Informatique, Université de Nice, Nov. 1987, 176p.
5. Montalban, M., Haren, P., Delcambre, B., 1988. Systèmes experts de conception fondés sur les spécifications. Les Huitièmes Journées Intern. sur les Systèmes Experts et leurs Applications, Avignon, Mai 1988, Vol 3, pp. 203-219.
6. Lawson, B., 1980. How designers think. Architectural Press, London, 1980.
7. Mc Allaster, D., 1978. A three-valued truth maintenance system, S.B. Thesis, Department of Electrical Engineering, MIT, Cambridge, MA.
8. Mc Allaster, D., 1980. An Outlook on Truth Maintenance., AI Memo No 51, August 1980, MIT-AI Lab., 44 p.
9. Doyle, J., 1979. A Truth Maintenance System. AI Journal 12, pp. 231-272.
10. de Kleer, J., 1986. Problem solving with the ATMS. AI Journal 28, pp. 197-224.
10. de Kleer, J., 1986. Extending the ATMS. AI Journal 28, pp. 163-196.
10. de Kleer, J., 1986. An Assumption based Truth Maintenance System. AI Journal 28, pp. 127-161.
11. Berlandier, P., 1988. Intégration d'outils pour l'expression et la satisfaction de contraintes dans un générateur de système experts. Rapport de Recherche INRIA, No 924, INRIA (Ed.), Nov. 1988, 43 p.

12. Haren, P., Neveu, B., Corby, O., Montalban, M., 1985. "MEPAR: Un moteur d'inférences pour la conception en ingénierie". 5<sup>ème</sup> Congrès Reconnaissance des Formes et Intelligence Artificielle, Grenoble, 27-29 Novembre 1985 - France, pp. 1273-1280.
13. Haren, P., Neveu, B., Giacometti, J.P., Montalban, M., Corby, O., 1985. "SMECI: Cooperating Expert Systems for Civil Engineering Design". SIGART Newsletter, April 1985, Number 92, pp. 67-69.
14. Poyet, P., De La Cruz, P., Miléo, T., Loiseau, J. N., 1989. Récentes Etudes en Matière de Simulations Tactiques Intelligentes. Les Neuvièmes Journées Intern. sur les Systèmes Experts et leurs Applications, Avignon, 29 Mai - 2 Juin, 38 p.
15. Poyet, P., Haren, P., 1989. A.I. Modelling of Complex Systems ; In: Modelling Techniques and Tools for Computer Performance Evaluation, Plenum Publishing Company, Plenum Press, Puigjaner, R., and Potier, D., (Eds.), 34 p.
16. Poyet, P., De La Cruz P., 1988. Une Nouvelle Classe de Simulateurs Destinée aux Aides Tactiques et aux Systèmes d'Armes. Conf. Spécialisées (Science et Défense), Huitièmes Journées Intern. sur les Systèmes Experts et leurs Applications - Avignon 1988, pp. 89-99.
17. Poyet, P., Haren, P., 1988. A.I. Modelling of Complex Systems. Invited Paper to the ACM and IFIP 4th Intern. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, Sept. 88, Palma de Majorque, 34 p.
18. Poyet, P., De La Cruz, P., 1987. Simulation Navale en Environnement Hostile. Actes de l'Ecole d'Automne d'Intelligence Artificielle. Institut d'Expertise et de Prospective de l'Ecole Normale Supérieure d'Ulm - Société Thomson, 10 p.
19. Poyet, P., Haren, P., De la Cruz, P., 1987. Un système expert de simulation navale. 6<sup>ème</sup> Congrès Reconnaissance des Formes et Intelligence Artificielle de l'AFCEC, pp. 587-592.
20. Chailloux, J., Devin, M, Dupont, F., Hullot, J. M., Serpette, B., Vuillemin, J., 1987. Le\_Lisp de l'INRIA version 15.2. INRIA, France.
21. AIDA, 1988. AIDA a Powerful LISP Portable Interface Builder, Manuel de Référence v 1.2., ILOG S.A., Paris.
22. Hullot, J. M., 1985. CEYX - Version 15. Rapports de Recherche INRIA No 44 et No 45, 19 p. et 83 p.
23. Cox, B. J., 1986. Object-Oriented Programming: An Evolutionary Approach, Addison-Wesley, Reading (Mass.).
24. Meyer, B., 1988. Object-oriented Software Construction, Prentice Hall International Series in Computer Sciences, 534 pp.

Leere Seite  
Blank page  
Page vide