# ARCHITECTURES FOR SECOND GENERATION KNOWLEDGE BASED SYSTEMS

José Cuena
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 - Boadilla del Monte
Madrid

## ABSTRACT

This chapter deals with the concept and examples of structured knowledge based systems.
First, an evaluation of the first generation knowledge representations is summarized.
Second, the role of deep knowledge is commented together with the knowledge structuring approaches based on the knowledge level specification of an agent and the task based approach for symbolic representation.
Third, several knowledge architectures are commented since SOAR, for general problem solving to the physical behavior representation architectures.
Finally, two architectures developed by the author to deal with flood management problems are presented as examples of real world knowledge based systems with complex structure.

# 1 INTRODUCTION

As a first result of the work developed along several lines in Artificial Intelligence from the start in 1956 three main paradigms were proposed for knowledge representation:

- The production rules representation obtained as a result of the developments along the heuristics programming approach where a flexible version of the algorithmic computation was proposed initially where the rules may be considered as atoms of algorithm operating in a working memory to be integrated along a path leading to a solution after a state space search. This paradigm is the more similar to the traditional computer science approach.

- The Horn clause representation where a set of logical rules are used to obtain answers for a set of predefined goals by applying an automated deduction technique. This paradigm is the result of the research along the line of automated deduction developed since early sixties that produced the resolution rule [Robinson, 65] together with several complete strategies [Luckham, 68], [Loveland, 79]. These results obtained to be applied for automated deduction were shown to be useful for problem solving by addition of an answer extraction process by [Green, 69] in such a way that several classical problems were adequately solved. The PROLOG language was proposed by Colmerauer and Kowalski after these achievements.

- The semantic network and frames approach was initiated from the psychology field and produced an approach to representation based mainly on hierarchies of frames with inference based on pattern matching and inheritance of attribute values with exceptions that allowed a representation meaningful and efficient both for inference and explanation.

These three representation approaches where the basis of the Artificial Intelligence offer to the applications world together with the successful experiences produced during the seventies with a set of prototype systems such as the well known MYCIN, PROSPECTOR, INTERNIST, etc.

This first generation representation tools produced a boom on interest for learning these techniques that were also used in simple applications. Several shortcomings were identified by the middle eighties (Winston 87, Chandrasekaran 83, Steels 87):

- The bottleneck of knowledge acquisition by lack of clear methods to build a knowledge base for a given type of problems.
- The lack of depth for modeling the physical processes.
- The lack of constraints management capability.
- The lack of multilevel reasoning capability.

All these limitations together with the marketing practice that promised too easy results by using a tool with a two week course, produced some discouragement in the initial, maybe excessive, enthusiasm.

The Artificial Intelligence community has proposed several contributions to overcome these limitations. This chapter deals with the main aspects that configured the second generation expert systems: First, the main concepts for knowledge structuring and deepening in new, less uniform, architectures are proposed. Later, different architectures types are described together with some examples of applications.

# 2 THE SECOND GENERATION EXPERT SYSTEMS

There is not a clear agreement on the characteristics of these new systems even there is not an agreement on the name of second generation, so several characteristics are proposed to define these systems:

- Modularity in the knowledge structure by using the task concept proposed by [Chandrasekaran 83, 86, 87].

- Multilevel reasoning. Once the encapsulation of knowledge is organized based on tasks it is conceivable that inside a task other tasks may be used as parts of its knowledge base and inference procedures.

- Capacity for incorporate deep knowledge by including enhanced explanation capability for management and legal problems and physical or chemical processes models in the knowledge base structure.

Some comments are presented first on the role of deep knowledge and, after, the general aspects of knowledge architectures are commented.

## 2.1 The Role of Deep Knowledge

The concept of knowledge based systems leaves open as a design decision the quality of the knowledge to be introduced in the system. In some cases it is acceptable a knowledge that allows simply to obtain answers for the problems to be solved by the system. In these cases the knowledge based structure allows flexibility in the sense that the knowledge may be updated, but the quality of the knowledge may be improved, in the sense that deeper concepts may be used to meet the same goal as may be shown by the following example.

* Shallow knowledge:

  *if the amount of flow in the next two hours is more than 400 m3*
  *then*
  *the river reach will be flooded.*

  This rule simply states a cause effect relationship compiling general hydraulics criteria with specific domain characteristics in such a way that it cannot be criticized by an user when he/she is a civil engineer.

* Deep knowledge: Three rules:

*if the slope of the reach is 1% and the depth of flow is between 2-2,20 m*
*the length of the wet section is 30-35 m*
*then the drainage capacity of the river reach is 220-230 Hm3/h*

*if the river reach at the upper level has a breadth of 35-40 m and the length of*
*the river reach is ≈ 500 m and the present safe level is 0,40-0,50*
*then the amount of water that may be stored without flooding is 100 $Hm^3/h$*

*if the river reach has an amount of water that may be stored without flooding*
*greater than the storage capacity*
*then the reach will be flooded.*

This level of representation uses elements of knowledge and structure explicit enough to produce a better explanation for a civil engineer in such a way that he/she can make judgements about its adequacy.

A deeper level may be defined if some of the concepts formulated by constant quantities are formulated by using variables.

*if the reach slope is x and the depth of flow is z and*
*the length of the wet section is u*
*then*
*the drainage capacity is model 1 (x,u,z)*

*if the river reach upper level breadth is x and the length is y and the safe level is z*
*then the amount of water that may be stored without flooding in one hour is*
*model 2 (x,y,z)*

These two expressions are a more general formulation from which it is possible the modeling of the previous case by deducing the values for instantiation of $x,y,z,u$ in previous knowledge units or by its introduction as facts.

In the same way a block of rules may be represented at deeper level by a more complex model. Even the models 1, 2 may be formulated at different depths (qualitative and quantitative).

Deepening the knowledge is necessary because in many applications it is required an improved explanation capability by two main reasons:

- Education applications, where the student must know the reasons of an answer to understand both the right knowledge to be used and the right way to use it. In these cases the quality of the knowledge presented in the explanations is the real quality of the system.

- User acceptance of the system recommendations: this is the case when responsibilities are associated to the decisions of the system (i.e. in real time emergency systems where important decisions are to be taken with significant socio-economic impacts as opening the valves on a dam or closing a four in the steel industry), the user needs to know the technical reasons of these recommendations, in these cases black box rules compiling technical reasons and expert criteria are not explanatory enough so models of physical behavior must be included in the knowledge base.

Also deepening the knowledge through model introduction allows the additional advantages of:

- Generality, because the models to be used may be pre-established in existing libraries in such a way that the same model may be used for different systems applications.

- Better knowledge acquisition because the knowledge acquisition activity is guided by the models structure.

The problem of using deep knowledge is the loss of efficiency because more detailed knowledge bases are to be dealt with and several levels of inference may be required. However, the present state of the art in the hardware allows the operation of such complex systems in acceptable conditions.

Anyway, it is possible to use two versions of the knowledge when efficiency losses are not acceptable in such a way that only when explanations are to be required the deep version is used.

## 2.2 Knowledge Architecture

A first generation concept was proposed for knowledge architecture: the blackboard architecture. It is a personal view of the author that these architectures, because of its generality, are mostly a good solution for module integration at the implementation level than a model of an agent knowledge.

The architecture concept should be defined from the knowledge level perspective proposed by [Newell, 82]. Newell proposed an informal specification of the structure of a knowledge base by derivation of its characteristics from those to be defined for an ideal agent acting in a given environment. The operation of such an agent is defined by:

- The goals that the agent is capable to meet.
- The knowledge elements to be used to meet these goals.
- A general criterion to use this knowledge: Newell proposed the principle of rationality (i.e. the agent uses in every moment the knowledge required to meet the goal on course.).

Once a specification of this type is defined the design at symbolic level must be formulated in such a way that a maximum of analogy with the knowledge structure specified be obtained.

[Chandrasekaran 83, 86, 87] introduced the concept of generic task as a structure where it can be distinguished:

- The premises for the task reasoning interface.
- The goals of the task: the set of concepts that may be obtained as a symbolic result of the task reasoning.
- The knowledge bases of the task, where several of them may be used to meet the task goals.
- The inference engines that are capable of using the different knowledge bases to obtain the possible values of the goals.

This definition shows the task concept as a symbolic unit to be used for the implementation of the knowledge elements identified in the knowledge level.

Chandrasekaran proposed these modules as the basic elements to be managed by an upper level inference procedure modeling the general problem solving strategy which may be considered a model of the principle of rationality.

A possible general approach to the design of knowledge based systems can be:

- First, a definition of the architecture to deal with a class of problems should be defined by using the steps of knowledge level specification and task definition.

- Second, an exercise of model formulation for different real world case studies should be used to test the conceptual capabilities of the architecture.

A feed back may result after the second step. Once an adequate architecture has been defined the implementation work may be started. This job may be developed on top of an existing environment for traditional knowledge representation or after an implementation work on top of UNIX.

This type of approach makes easier the development of the applications in the area of problems where the architecture is designed as may be shown in the figure 1, where the process of knowledge engineering is reduced to the specification of the different knowledge units proposed by the architecture. The traditional knowledge engineering approach requires the identification of these knowledge structures as intermediate steps as in the KADS [Wielinga et al, 88] approach until some units of the basic level of representation are attained.

This type of architectures allow an adequate definition of the knowledge depth for every component of expertise in such a way that acceptable efficiency and explanation quality be obtained.

Two types of architectures may be defined:

- Architectures for general problem solving (GPS) where an abstraction of any problem described by an initial state and a final goal may be processed, this is the approach of the SOAR system [Laird et al. 87], a version of the GPS approach supported by an architecture based on problem spaces, or the PRODIGY system [Minton, 88] an extension of the ideas of STRIPS [Fikes, Nilsson, 71].

- Architectures for specific problem solving (SPS) oriented towards the design of a class of problems by maintaining a level of generality with respect to the *ad hoc* representations. The classes of problems to be solved with this approach are derived from professional areas where a set of established concepts are available in such a way that several predefined tasks may be available and several methods of inference may be predesigned, all of them accepted by the professional experts.

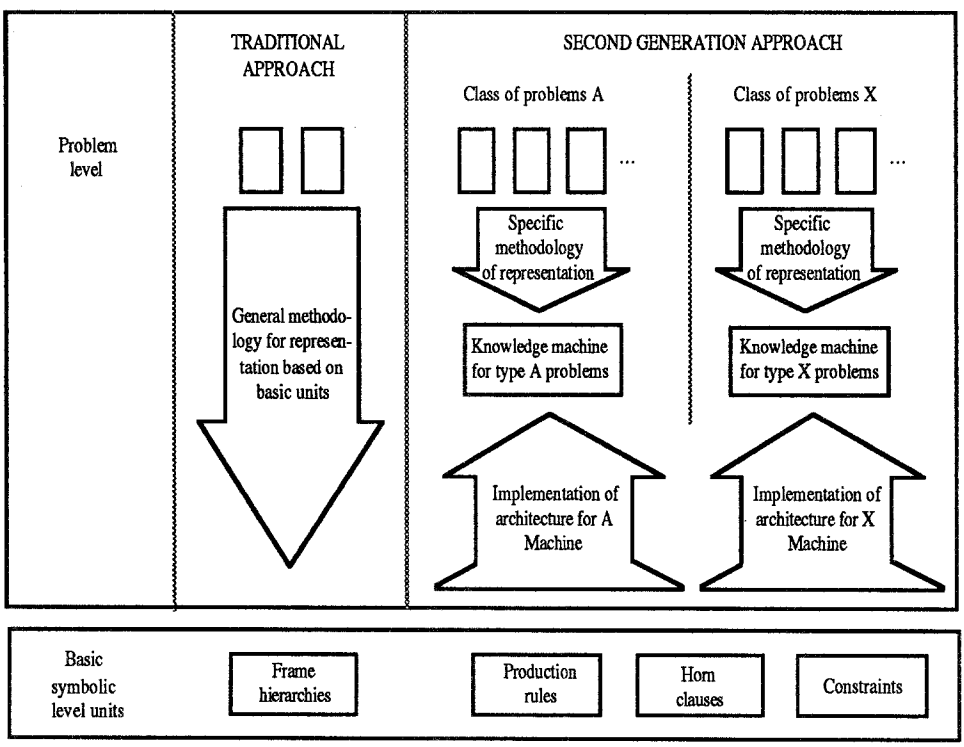


*Figure 1: The two approaches for knowledge acquisition*

In the next paragraphs, several architectures of both types are presented.

Given that an architecture in general is defined by its typical components and its structuring style the format that will be used for an architecture definition is based on:

- The set of knowledge units that may be used for the definition of a model.
- The set of basic associated procedures of inference and knowledge acquisition.

In such a way that a knowledge model may be built by using instances of the knowledge units complemented by additional ad hoc knowledge bases. The model will be operated by using the predefined inference procedures integrated in an *ad hoc* inference procedure.

The description of the different examples will be adapted to this format.

# 3 GENERAL PROBLEM SOLVING ARCHITECTURES: THE SOAR EXAMPLE

SOAR is a structured version of the GPS concept [Newell, Simon, 63] supported by production rules. The basic SOAR structuring concept is the problem space, defined by:

- A set of possible states characterized by the values of a predefined attribute set.
- A set of problem space operators formulated by production rules.

A problem space operates by receiving an initial state and a goal to be met. The problem space production rules are used by the general inference engine until an instance of a goal is obtained or an impasse situation is found.

The integration of the different problem spaces operation is produced by an inference procedure of subgoaling fired by an impasse situation in the problem solving process internal to a problem space. The inference procedure may be described in a general way by the following steps:

- Given a problem, a problem space is searched to solve it.

- Once a problem space is found its operation is fired with the parameters of the problem.

- If no solution is found in the problem space because the inference procedure is unable to continue a process by lack of discernment criteria (impasse situation), then subproblems are generated whose solutions may help to improve the impasse situation. These subproblems require to be solved in another problem spaces that the control reasoning must find.

- Once an auxiliary subproblem is solved, the solution is returned to the upper problem space. The search for a solution in this subproblem may have impasse situations that may require the use of other problem spaces. In this tree of problem spaces for different subproblems (see figure 2) it may happen that different instances of the same problem space be used.

- The process of impasse resolution by using other problem space may be synthesized and abstracted through the chunking procedure to give a new production rule to be included in the problem space originating the impasse in such a way that a future similar impasse situation may be solved without subgoaling.

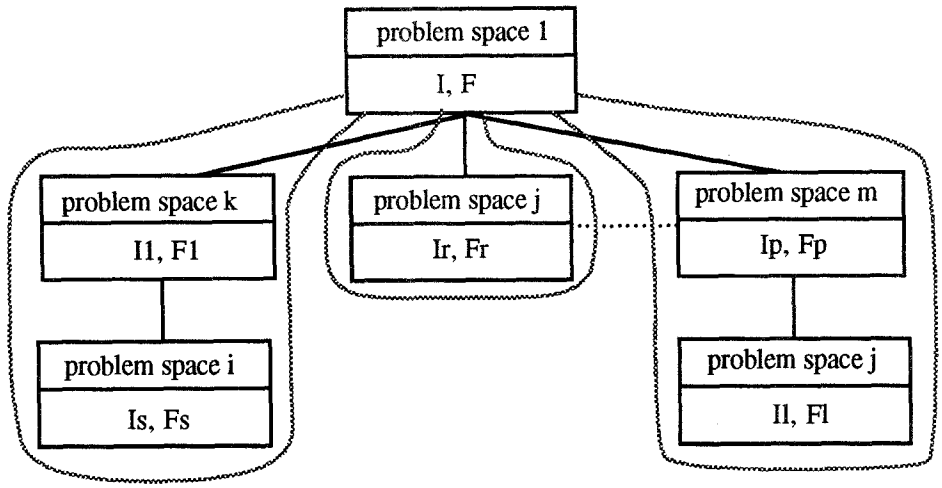


*Figure 2: Problems space management in SOAR*

To support this general reasoning concept two types of problem spaces are required for problem solving:

- Domain problem solving to deal with subproblems that may be integrated to produce a solution.

- Control decisions: problem spaces to support the reasoning on problem space and goal selection.

This reasoning concept is formulated in SOAR in an uniform way by using the context concept where control concepts and domain concepts are integrated (goal, current problem space, current state and current operator). The uniformity of the symbolic representation approach must not conceal the differences in the reasoning concept at the knowledge level.

In summary, the concept of the SOAR architecture at symbolic level may be described by:

- A knowledge base formulated by production rules to deal with domain problem spaces and control problem spaces.

- A working memory built by objects of different types:
  * Objects used in the different domain problem spaces.
  * Context objects representing intermediate states of control.
  * Preference objects modeling the assignment of preference levels to context elements.

- An inference engine for control problem spaces reasoning operating on preferences and context objects in such a way that using the preferences modifies the context.

- An inference engine to deal with inference in domain problem spaces dealing with the production rules modeling the operators and the domain state objects.

- A working memory management system.

- A procedure for learning by Chunking after a subgoaling process.

The SOAR reasoning concepts are implemented by the operation of the set of production rules in two phases:

- Elaboration phase.
- Decision phase.

The elaboration phase is a parallel firing of production rules until the quietness situation is met. The rules are formulated as:

$$C_1, C_2, ..., C_n \text{ --> } a_1, a_2, ..., a_m$$

where $a_i$ are actions generating values for attributes of the different object types and $C_i$ preconditions.

In the decision phase the contents of the open contexts are reviewed to decide which objects are to be introduced as values of the context attributes not yet defined. This reasoning step uses an internal knowledge base not to be modified by the user to evaluate the final choices using as premises the preference values generated in the elaboration phase.

This decision phase may find an impasse situation when (1) no clear choice may be done because too many alternatives may be selected, (2) contradictory options are recommended with the same level of preference, (3) no changes are produced in the context situation after an elaboration-decision phase, (4) the selected choices were unable in these cases to find a solution. As has been commented before the subgoaling procedure will propose a problem to be solved in other problem space.

SOAR may be understood using the general format of architectures definition commented before as an environment where models for dealing with problems by subgoaling may be formulated based on a set of predefined entities:

- A pattern for problem spaces definition based on:
  * The type of problems to solve defined by the type of objects that may be used as premises intermediate goals and final goals.
  * The knowledge to be used for problem solving at domain level and control decision level. This knowledge may be defined by production rules.

- A library of predefined problem space patterns.

- A set of auxiliary procedures for:
  * Inference at domain level (it is proposed the production rules inference engine).
  * Inference at control level (the same inference engine for elaboration + preference evaluation in the decision phase).

* Knowledge acquisition:
  .- Production memory management.
  .- Learning by Chunking.
* General subgoaling inference engine (which may call the others inference engines).

It could be defined also a pattern for preference reasoning to be adapted by the user.

This environment may produce knowledge models structured by problem spaces that may be interpreted by the general inference procedure. This is summarized in the figure 3.

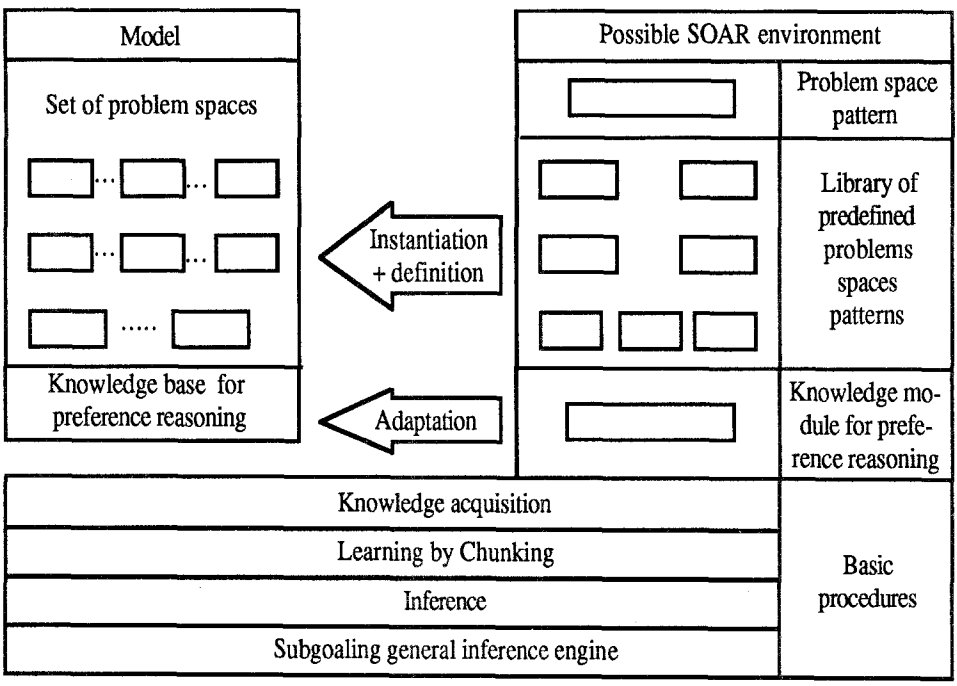


*Figure 3: A possible version of the SOAR architecture*

The authors of SOAR proposed several examples one of them a version of R1 where it was defined a model based on several problem spaces every one with knowledge about different subproblems of configuration.

## 4 GENERIC TASK ARCHITECTURES

[Chandrasekaran 83, 86, 87] proposed a knowledge structuring criterion more adapted to every problem type characteristics in such a way that in many cases the problems may be solved by integration in a general search strategy of specialized problem solving modules.

An example proposed for this approach was a hierarchical diagnosis reasoning with the following features:

- A first module reasons about the feasibility of a problem producing as possible conclusions:
  * There is no problem.
  * Maybe, there is a problem of possible types A, R, S.

- Next level modules reason about the feasibility of A, R, S problems using the same data as the upper level module together with additional ones obtained for every problem type. The possible conclusion at this level is evidence in favour or against the existence of the type of problem analyzed proposing, in the positive case, the subtypes of problem to be analyzed.
  This process continues until some possible problems are identified with sufficient details (see figure 4) at the bottom level.

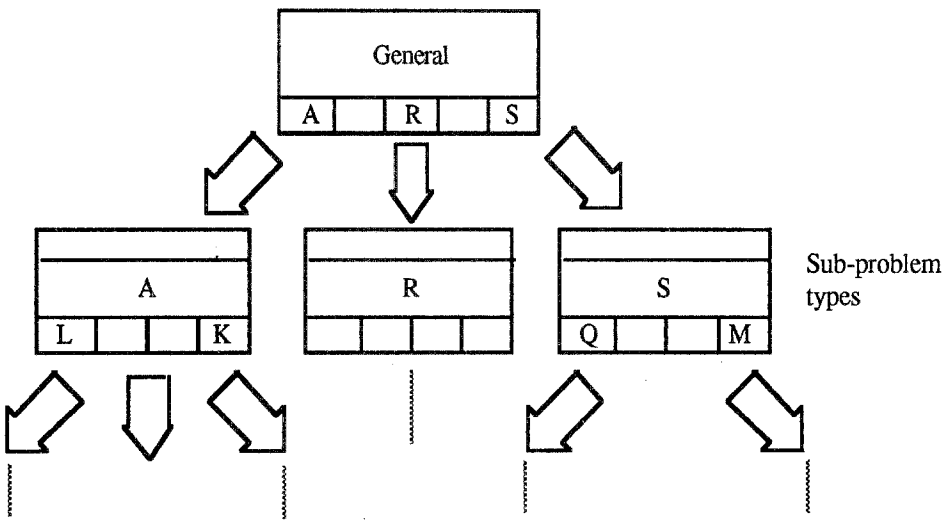


*Figure 4: The diagnostic tasks tree*

This top down search strategy for classification integrates reasoning modules for every problem type. Every module may have different knowledge representations and inference procedures. Even, internally, a module may be organized in a similar way by a set of internal modules managed by another control strategy.

According with the previous description a task based symbolic representation may be described by:

- A set of generic tasks modeling the knowledge to deal with the different analysis modules.
- An inference procedure supporting the general problem solving strategy.

A generic task is a knowledge unit defined by:

- Premise concepts attributes.
- Goal concepts attributes.
- Knowledge bases required to meet the different goals.
- inference procedures, using the knowledge bases to infer the possible goal values.

This structuring concept may be used for the symbolic representation design of an agent knowledge level specification. The general strategy to be implemented must be a model of the principle of rationality and the different tasks are the symbolic representation of the agent knowledge to meet its goals.

The generic task approach may be used for modeling general problem solving and specific problem solving, i.e. the SOAR model may be implemented by a general subgoaling inference engine controlling the operation of generic tasks modeling the different problem spaces. Also specific systems for diagnostic such as the one commented before may be formulated based on the generic task approach (in fact the Ohio State University Laboratory of Artificial Intelligence Research developed several diagnostic systems such as MDX for medical diagnoses, AUTOMEC for fuel problems in cars or WELDEX for welding failure analysis).

Other general reasoning models proposed by [Chandrasekaran, 83] were the "what will happen if" (WWHI) and the routine design.

The WWHI pattern deals with the propagation of impacts through a structure in such a way that, if an initial event happens in a module, possible impacts are deduced to be transmitted to other elements of the structure. In this type of models the generic tasks are knowledge units representing physical entities models using as premises the possible effects transmitted from the environment and as possible goals the resulting effects to be propagated by the environment. The knowledge base represents the cause-effect behavior of the physical entity and the inference engine identifies possible behaviors consistent with the knowledge base and the premises.

The general strategy is an exploration algorithm along the network of possible connections among physical entities guided by the answers of every task.

A routine design model was proposed by [Brown, Chandrasekaran, 89] based as main concepts on generic tasks modeling specialist knowledge at different levels. The basic elements of this approach are:

- The basic operators modeling the elementary actions on the design development (they may be represented by preconditions, add list and delete list).

- Basic task: a fixed sequence of operators, this concept is analog to the macroperators of STRIPS [Fikes et al, 71].

- Plan: sequence of actions, tasks and specialist calls.

- Specialist: a generic task organized by:
  * A knowledge base formed by: (1) a library of predefined plans as primary facts, (2) a rule or constraint base for making choices in the library of plans according to the questions formulated as premises for the task, (3) a consistence knowledge base to test if the problems are acceptable for the task.
  * An inference engine test the adequacy of a problem for the specialist and proposes a set of feasible plans to solve the problem.

- A model for a design problem is defined as a set of consistent specialists (i.e. such that the plans proposed by every one include existing tasks, operators and specialists in the model).

- An inference engine for hierarchical exploration of the model:
  * The first level specialist defines the possible general plans for design proposing a first plan.
  * The second level specialist integrated in a selected plan in the upper level propose possible plans for the next level. It may happen that some specialist cannot accept the task proposed in a plan in this case backtracking is produced to the upper level specialist next plan.

This is summarized in the figure 5:

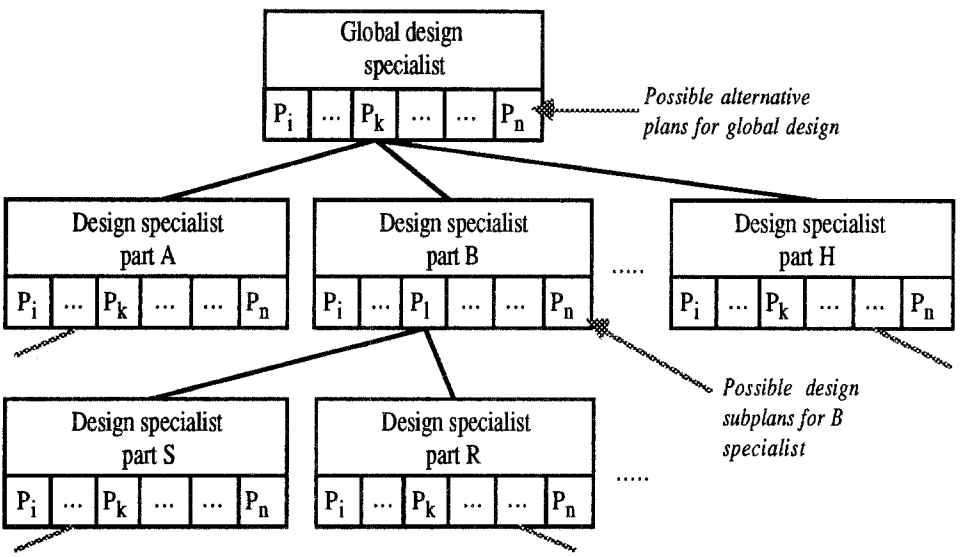


*Figure 5: A reasoning tree for routine design*

This modular concept may be improved because it allows the integration in the model of conventional planning tasks that may be considered as specialists incorporating a set of basic operators and basic tasks with a planning procedure as inference engine similar to STRIPS, ABSTRIPS, etc., that produces as a result a set of feasible plans or an answer of infeasibility.

Also the concept of specialist may be improved by using the Case Based Reasoning (CBR) approach. In fact, if the number of possible plans included in a specialist is not too large it may be reasonable the proposed approach based on the choice using a classification rule base; however if the number of plans is too large it may be reasonable to use a reduced sample of plans representative enough. This reduced sample may be used by an inference procedure using the rule base to choose a reasonable plan near enough to the goals to be met and, using CBR, makes an adjustment of the plan selected to the specified goals.

In summary it is possible the definition of a specific architecture oriented towards routine design by definition of a set of specialists at the different levels and of different types (rule based, plan selection, planning and CBR).

This approach allows a better structuring of design systems such as R1 where a linear sequence of specialists modeled by contexts of production rules was applied by using the, not too cognitive, artifact of a context transition formulation based on a production rules standard control strategy.

A group directed by the author of this paper has developed an architecture for failure diagnostic and repair to be used for education where a set of diagnostic tasks and planning specialists are used for diagnostic and repair plan design. This system will be used for education where a set of diagnostic tasks and planning specialist are used for diagnostic of a pneumatic circuit of a bus (for demonstration purposes). The main control strategy developes first a typical diagnostic tree as such presented before; once the final leaves of the tree propose a set of failure hypothesis several planning trees are fired based on these failure scenarios, those planning trees are similar to the design ones where a high level specialist proposes a set of plans for repairing the failure that may be detailed by specialist at the lower level (see figure 6).

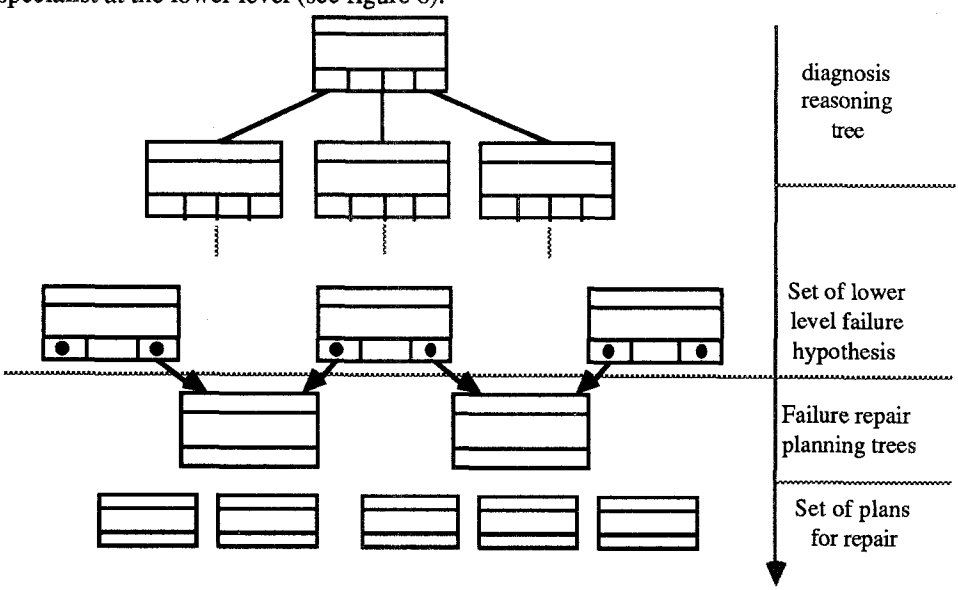


*Figure 6: Integration of reasoning and failure repair*

This project is being developed to be illustrated by video disk images showing the operations for diagnosis and the repairing plans.

The problem solving strategy develops the diagnosis tree and incorporates models of the criteria for choosing the initial specialists of the repairing trees, for finally developing the planning trees.

This application may be considered as an example of integral modeling of an agent for specific problem solving in the area of the pneumatic circuit of buses, by using a second generation approach:

- First, the knowledge level specification establishes:
  * The goals of the agent: identification of the different possible failure scenarios. Identification of the possible plans for repair.
  * The knowledge of the agent conceived at different levels of abstraction for diagnostic and different levels of capacity for failure repair.
- After, the symbolic representation is developed by using the task approach to model the different knowledge elements together with the general problem solving strategy.

In the first version of the implementation the knowledge contents of every task were shallow rule bases obtained from expert interviews but it may happen that in some of the tasks a deeper modeling be applied representing aspects of behavior, this is discussed in the next paragraph.

### 4.1 Architecture for Physical Behavior Reasoning

The need for commonsense reasoning models about the physical world was proposed by [Hayes, 79] using first order logic language. A deeper approach was proposed by [De Kleer, Brown, 84], [Forbus, 84], [Kuipers, 86] where qualitative versions of the usual quantitative models were proposed as main bases of the formulation using constraints.

De Kleer, Brown proposed a formulation based on a way of understanding the physical structure of an industrial installation. Forbus, alternatively proposed a formulation based on a way of understanding the behavior of an installation as a set of processes. Kuipers proposed a model of understanding based on a set of equations.

The three approaches aim to the identification, given an initial state of the feasible transitions to other states by computing the consistent sets of changes of the state parameters. This leads to build an envisionment graph: a directed graph with its nodes as possible states of the physical entity and its links as transitions between states. The graph could be used for shallow modeling of behavior. However, when the physical system is a complex one, building the envisionment graph is not feasible. When the model is used for real time monitoring of an installation this graph is not necessary, only a short term transition tree is required; even not a very detailed one because, when monitoring, only the transitions to significant situations for this purpose require to be predicted; using all these restrictive criteria it may be feasible the application of these modeling approaches, to real world problems.

As has been commented, the first two approaches have the advantage of being near to the physical phenomena, the Kuipers model is simply a qualitative approach to the computation of a mathematical formulation. Given that our goal is representation we shall comment the components and processes based approaches, the Kuipers approach may be used internally to both approaches for solving computing problems.

In the component based Approach, every component is described by a set of permanent parameters modeling aspects of its structure and a set of time dependent parameters modeling its behavior along time.

The behavior of the component is described by a set of equations relating the changes along time of state parameters. Given a set of equations two approaches are possible:

- The De Kleer, Brown approach oriented towards a qualitative description of the equations. To do that it is assumed (1) a qualitative value domain for every state variable, (2) a qualitative calculus representing the operation of the equation in the new domain, (3) a new expression of the equation is obtained according to the previous definition. These qualitative versions of the equations are named confluences. The ENVISION algorithm is defined to compute consistent sets of changes of the state parameters and its time *derivative*.

- The Kuipers approach oriented to "understand" qualitatively the equation expressions, in such a way that a language is defined to describe the equations, a qualitative space is defined for the variables and an algorithm QSIM is formulated for inference using as constraints the equations expression in the predefined language.

QSIM as ENVISION propose a set of feasible changes from a given state. Both inference engines may be applied at component level or at installation level.

An installation may be described by the integration of the component models via the models of the conduits that physically connect the different components (usually the conduits are represented by equalities among variables from different components).

Inference may be organized in one step if the whole constraint system is solved or in several iterative steps if local component solutions are proposed to be filtered in the other components constraints through the conduits conditions.

Once a consistent set of transition alternatives is defined. The possible transitions may be identified. For the new states the process is repeated until no more new states are produced. In this case the envision graph is obtained. As has been commented before this is not the usual case when monitoring an installation in real time where it is acceptable a differential envisionment tree where several levels (around two) of feasible future states are generated from the present state.

Even with these reductions it may happen that the size of the detailed constraints model be too large. Two approaches may be considered to deal with this problem:

- The introduction of goal oriented filtering criteria in such a way that only meaningful states for the goals be generated (i.e. if an emergency management system is using the model only the short term problematic situations are to be considered). This may be done by using shallow knowledge bases to regress the significant goal values to the current state concepts values to be used for constraining the current state decision.

- To use a set of models with different scopes of analysis in the physical structure, in such a way that at an aggregate level parts of the installation be considered as components with more simple equations. If a manageable size of a model is defined by a maximum number of constraints a focusing knowledge may allow the navigation on a hierarchy of representations with different level of detail in different areas of the installation.

In summary, qualitative modeling does not require the simple integration of confluence sets but the definition of a knowledge structure where the operations of knowledge acquisition and inference to solve problems be feasible and efficient enough.

At the knowledge level an agent may be specified capable of predicting pertinent behavior scenarios of a given physical object. To meet this goal the agent may have knowledge for understanding the behavior of the different parts of an installation but also to reason about the modeling approaches to compute this behavior.

A symbolic model for knowledge representation of an installation will be defined by:

- A set of qualitative models of the installation components with different levels of detail: from the basic components to aggregate artificial components modeling as single components the parts of an installation.

- A knowledge base for focusing by definition of a tree of representations with different levels of detail oriented towards the progressive definition of the goals.

- A knowledge base for goal regression. For example in a flow system this base should be a shallow version (rule based) of the behavior model of the system relating the flows at the end of the flow circuit with the flows at intermediate points, in such a way that intermediate value intervals may be identified corresponding to a given goal to be explored (this is a case that will be commented with more detail in the SIRAH system example).

According to the previous considerations a class of task based architectures could be considered to deal with qualitative physics problems using the component approach with the following features:

- A library of predefined component models. This is important because in a given branch of a professional field it is possible the identification of a set of basic submodels that may be presented often in the type of installations in the branch. These submodels may be alternative approaches to the same component with different degrees of complexity and may be also models of aggregate installations to be considered as components for focusing analysis.

- A set of basic procedures for:
  * Knowledge acquisition.
  * Constraint based inference.

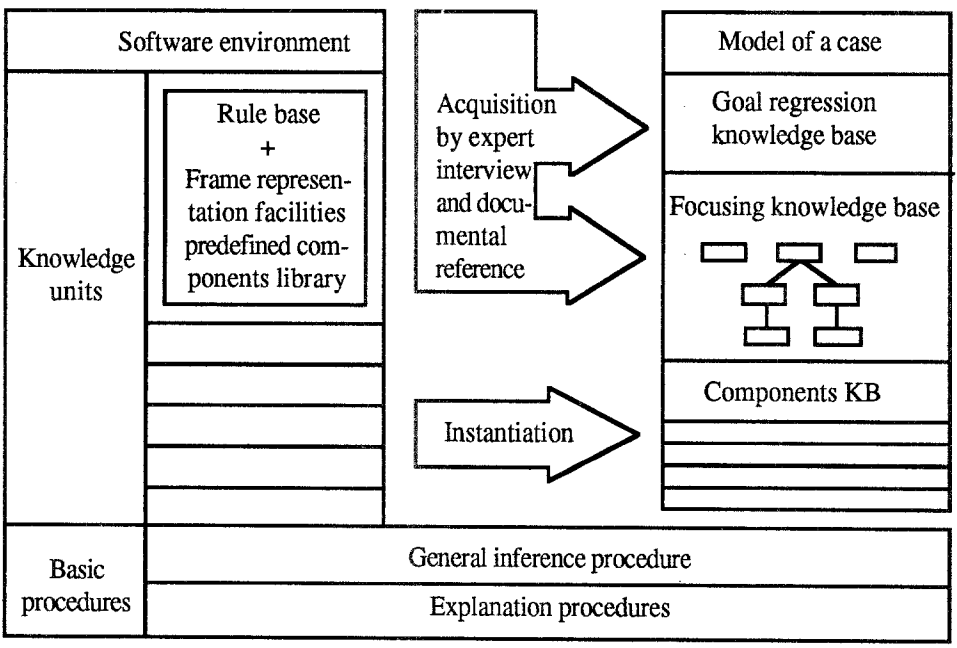


*Figure 7: A possible architecture for component based models design and operation*

This type of environment may allow building models such as the ones described before (see figure 7).

In the process based approach an agent for understanding the physical behavior has the same general characteristics defined for the component based approach (parameter with qualitative values with the same structural conditions) but in the Forbus approach the idea of focusing is already included since the first presentation of the model.

In fact, the world is many understood as a set of active individual views and a set of active processes.

An individual view or a process are activated if:

- A set of objects required to configure the view exist.
- A set of predicative conditions about the world is true.
- A set of quantity conditions is satisfied by the parameter values describing the states requiring quantity for its definition.

Once a process or an individual view are activated it may be concluded in both knowledge units a set of complementary conditions representing the view of the world relevant enough for the unit purposes. The process model units have as an additional feature the inference of the set of influences in the parameters describing the state of the world along time that are affected by the process (i.e. a process of heating may produce positive influence in the temperature). The influences are described as qualitative equations relating two or more variables.

The inference strategy has three main steps:

- Activation of individual views and processes after the introduction of the initial state (this is a process that requires chaining because, with the initial state, a set of individual views may be activated. These active views may generate as conclusions complementary facts that may allow the activation of additional views etc.

- Activation of processes based on the set of already active individual views. This activation may produce additional facts that may fire others views and processes.

- Once a quietness state is obtained the resolution of influences as a set of constraints generates a set of possible alternative state changes that give the basis for an envisionment tree definition. [Forbus, 90] stratified the influence resolution in two phases: the unambiguous influence satisfaction and ambiguous influences satisfaction based on a previous step where reasonable criteria for taking out the ambiguities are introduced. In this phase a goal regression filtering may be required to prune the envisionment paths not relevant enough for the required goals to predict.

A software environment may be designed to process such type of models allowing facilities for views and processes specification and for the addition of knowledge bases for goal regression together with the procedure for knowledge acquisition and inference engine; [Forbus, 90] specifies a version of this type of inference engines using ATMS.

In the process based approach is more difficult the previous definition of a library of processes given that the preconditions of a process may reference specific aspects of an installation.

## 4.2 Architectures for Real Time Decision Support in Physical Systems

In this paragraph two specific cases of knowledge structure are discussed oriented to decision support implemented on top of real time information systems. Both models deal with the same problem: flood management. This problem was formulated with an outline of its possible solutions in [Cuena, 83].

A possible knowledge level description of an agent for decision support in flood management is:

- The agent must recommend as final goals which actions should be taken for dam control and civil defense.

- The agent must be able to meet as subsidiary goals the short term problems identification.

- To meet these goals the agent knowledge must be able to understand (1) the emergency operation: which resources to be used for which problems, and (2) the physical environment behavior.

Two approaches have been developed for symbolic representation of this agent [Alonso et al, 90], [Cuena et al, 91], both have the same modeling approach for the emergency operation knowledge: an ad hoc rule base or frame base modeling the specific criteria of the authorities. The knowledge for understanding behavior is represented in one case by quantitative models embedded in a set of rule bases (the CYRAH architecture (Spanish acronym for Calculus and Reasoning in Hydrology)) and in the other case by qualitative models organized by tasks (the SIRAH architecture (Spanish acronym for Intelligent System for Hydrology Reasoning).

Both models aim to understand the behavior of a watershed under rainfall.

A watershed surface is composed of reception areas that are disjoint surface units with a single outlet for water flow. During rainstorms the outflows drained from reception areas enter the drainage channel network where it can be distinguished the upper basin networks with steep slopes in channels and high velocities and the final lower river reaches where milder slopes produce slow velocities and possible risk of overflow when storms are present.

Several watersheds may be involved in a flood management problem if they drain to the same floodable river reach.

The general inference procedure is analog to the reasoning procedure in an office for prediction when time is available:

- Interpretation of present and recent data received from the real time data base in qualitative form.

- Minimal prediction of the near future assuming the external actions limited to the registered rainfall until the present moment. To meet this goal a simulation run must be done with a numerical model of watershed behavior (named SAVEL).

- Potential problems identification based on the present situation and the minimal prediction defined before. This allows the choice of which of the component basins are to be simulated with future rainfall scenarios: those where present and incipient problems are detected with the minimal prediction. This step may then generate an economy in computational effort.

- Simulation of the different selected watersheds using the SAVEL model with a previous generation of scenarios based on meteorological predictions and the predicted outflows from upstream basins. This step produces as a result a first set of relevant qualitative behaviors of the different watersheds that may be used for problem evaluation or as input for floodable river reaches.

- Numerical simulation of the floodable final river reaches preceded by a reasoning step about relevant watershed inflow combinations (scenarios) generation. As a result of this step the possible behavior in terms of water levels of the floodable river reaches are obtained.

- Problem evaluation, given the present and predicted states of flows and water levels, the knowledge bases for the different problem types are fired in such a way that possible problems in different time horizons are identified.

The figure 8 shows a summary of components of the CYRAH environment and model structure based on:

- Predefined knowledge and data units to deal with the knowledge representation and data formulation tasks for model building.
- A set of procedures for knowledge acquisition and inference.

To build a model in CYRAH interpretation tasks and simulation tasks have been defined together with control tasks to be used for guiding the general inference procedure. Also the watershed structure based on reception areas, river channels and reservoirs, integrated in a tree-like network and must be defined the final floodable river reach structure to be used by the simulation task. Finally, frames for characterizing typical problems are included.

Basic tools for knowledge representation internal to these structures are:

- A subsystem for frame hierarchies management embedding rule bases in the frames for some attributes value definition. The rules are used with backward reasoning with uncertainty management. Every frame may inherit rule bases for attribute value deduction by traversing the hierarchy.

- A relational data base supporting the parameter and structure definition for the simulation tasks.

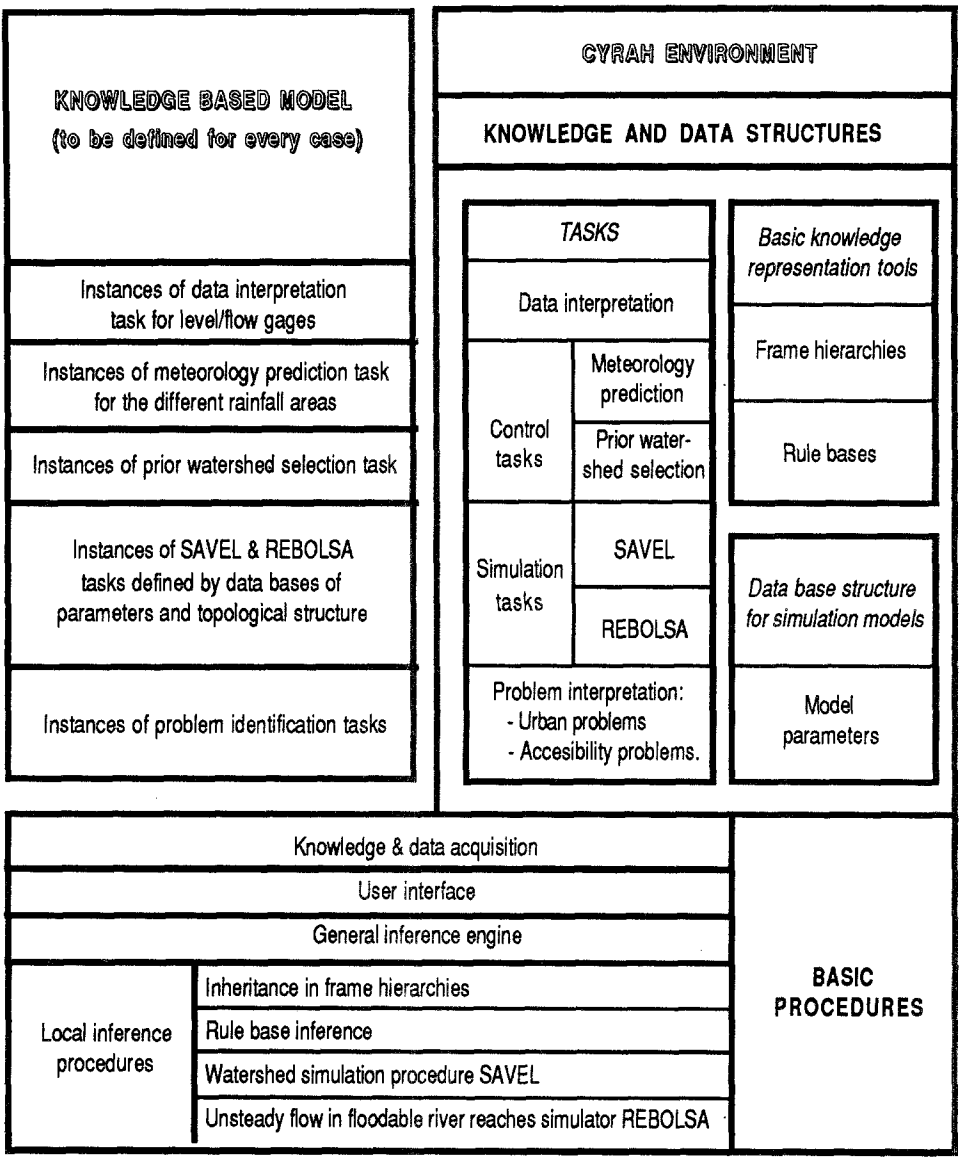


*Figure 8: The CYRAH environment and user model structure*

Also in this figure the general structure of a user model is presented based on several knowledge unit instantiations and databases representing respectively the criteria for simulation management and problem identification and the data for simulation models.

In this paragraph the general reasoning procedure is described first and, later, the different knowledge components are presented.

As it is shown in figure 8 two simulation tasks are provided by the environment (SAVEL and REBOLSA) with a quantitative procedure for simulation of behavior that may be considered an inference engine using a knowledge base describing the physical structure of the watershed or the river channel with premises the initial state and the scenario of external actions (rainfall time series in a watershed for SAVEL and flows form watersheds for REBOLSA). SAVEL is a model based on the packages HEC-1 of the U.S. Corps of Engineers and the SWMM of the Environment Protection Agency and REBOLSA is based on a finite difference scheme for numerical integration of the Saint Venant equations for unsteady flow in open channels. Also special equations are developed for floodable side packets behavior simulation.

It may be remarked also the inclusion of two control tasks where criteria are represented for defining reasonable scenarios for meteorology short term prediction and for selection of the problematic watersheds after the first step analysis with the minimal prediction. These knowledge bases, formulated with rules, model the expertise of the professional civil engineers to make reasonable assumptions during the analysis of a problem. Its use integrated with the quantitative models is an adequate representation of the professional knowledge for dealing with flood management problems.

The inference engine includes also criteria for synthesis of the results of the sets of simulation runs produced by the models.

In the SIRAH architecture a flow system may be conceived to describe the behavior of a watershed under a storm: the rain acts as a flow along time on the reception areas that produce outflows to enter in the upper basin networks which drain flows entering the floodable final river reaches.

Two types of tasks are considered for this type of systems: the *scenario generation task* for defining reasonable scenarios of external actions and the *basic simulation task* producing possible behavior for a given scenario of external actions.

Three qualitative basic simulation tasks (BST) are defined in SIRAH according to these concepts:

- *Reception area (RA)* to model the behavior of an area submitted to rainfall and producing as a result flows to be drained by the transport network.

- *Upper basin network (UBN)* to model the fast flow river network behavior in the upper basins where the flows generated in the tributary reception areas are concentrated producing surface flows. These flows produce impacts in the low basin floodable river reaches.

- *Floodable river reach (FRR)* to model the water levels behavior in the low basin floodable river reaches submitted to different flows produced by the upper basin networks.

Every BST module is defined by a frame with:

- The attribute premises for the internal reasoning process (permanent attributes and time variables attributes).

- The attribute conclusions, set of possible values of state variables to be used for building reasonable scenarios for the next BSTs.

- A knowledge base defined by rules and constraint confluences for qualitative modeling of behavior.

- An inference engine to reason, using the knowledge base, from premises to conclusions.
  There may be a knowledge base to help in the creation phase to infer also advanced characteristics in attributes from more primary attributes.

Three SGT types have been defined:

- The *meteorological scenario generation task* to infer possible significant goal combinations of local rainfall predictions acting upon the set of reception areas.

- The *flow transport scenario generation task* to infer the relevant combinations of input flows to the upper basin networks.

- The *floodable river reach scenario generation task* to infer the combination of output flows from the upper basin river networks draining to the floodable river reach.

These SGT's have a general structure based on:

- A frame with: (1) Attribute premises: possible values of every scenario component and significant values of every pattern of final behavior to be regressed. (2) Attribute results: lists of every feasible combination that makes a scenario.

- Two knowledge bases: (1) Modeling conditions to constrain the scenarios to be relevant for the final behavior prefixed patterns (i.e. if it is possible to regress the problem patterns to be explored, they may be used for constraining the scenario generation). (2) Modeling conditions for spatio-temporal consistency of the predictions belonging to the same scenario.

- An inference engine: for reasoning in two steps: (1) to generate, using both knowledge bases, consistent sets of possible values of every scenario variable. (2) to generate scenarios defined by combinations of the inferred possible values of scenario variables that are consistent with both knowledge bases.

These general tasks have to be complemented by an ad hoc knowledge for potential problems identification as described below.

The general reasoning procedure for using these basic knowledge units has as main steps:

a) Use of an ad hoc knowledge base for real time data interpretation to identify the present problems and the potential future problems. To explore the feasibility of these future problems this knowledge base defines patterns of final behavior to be explored by the predictive reasoning (i.e. potential problems to be explored may be defined if in some places levels are high but still not problematic levels or they are low but the rate of level change is important etc.)

b) Use of the meteorological scenario generation task to propose possible future rainfall scenarios on the surface of the basin. These scenarios are generated in order of proximity to the existing registered trend and only the first is retained.

c) A depth first search process of qualitative simulations along the different physical elements is produced:
   * The first meteorological scenario is introduced as input to the set of reception area task modules.
   * A first scenario of input flows is generated from the predicted flows drained from the reception areas for the UBN's.
     -. The scenario of upper basin network inflows is introduced as input to the set of river networks in upper basins.
     .- A set of upper basins outflow scenarios is generated to be introduced as input to floodable river reach tasks.
   * Simulation of scenarios by the floodable river reach task.

Once a set of behaviors (flows or levels) is obtained in this process the exploration of the next scenarios for rain or upper basin flows or outflows is decided based on a metacontrol reasoning step that uses as premises the resulting behaviors and the current patterns of final behavior to be explored. This control reasoning step is produced by a general control task that generates additional constraints to the SGT in such a way that, when these SGT are fired with the new constraints, they produce a first scenario which once simulated is expected to produce patterns of behavior different enough from the already inferred. With the scenarios generated in this way a new depth first traversal is done from b) or c).

This process is repeated until some SGT generate no feasible scenarios or a previously defined time period or number of iterations is met.

A shallow version of the concepts of the models may allow the definition of a model that by abduction may obtain the possible values of the premises consistent with the predefined problems to explore. In the figure 9 the relationships of this process at the different levels with the exploration tree of scenario generation and simulation, are shown.

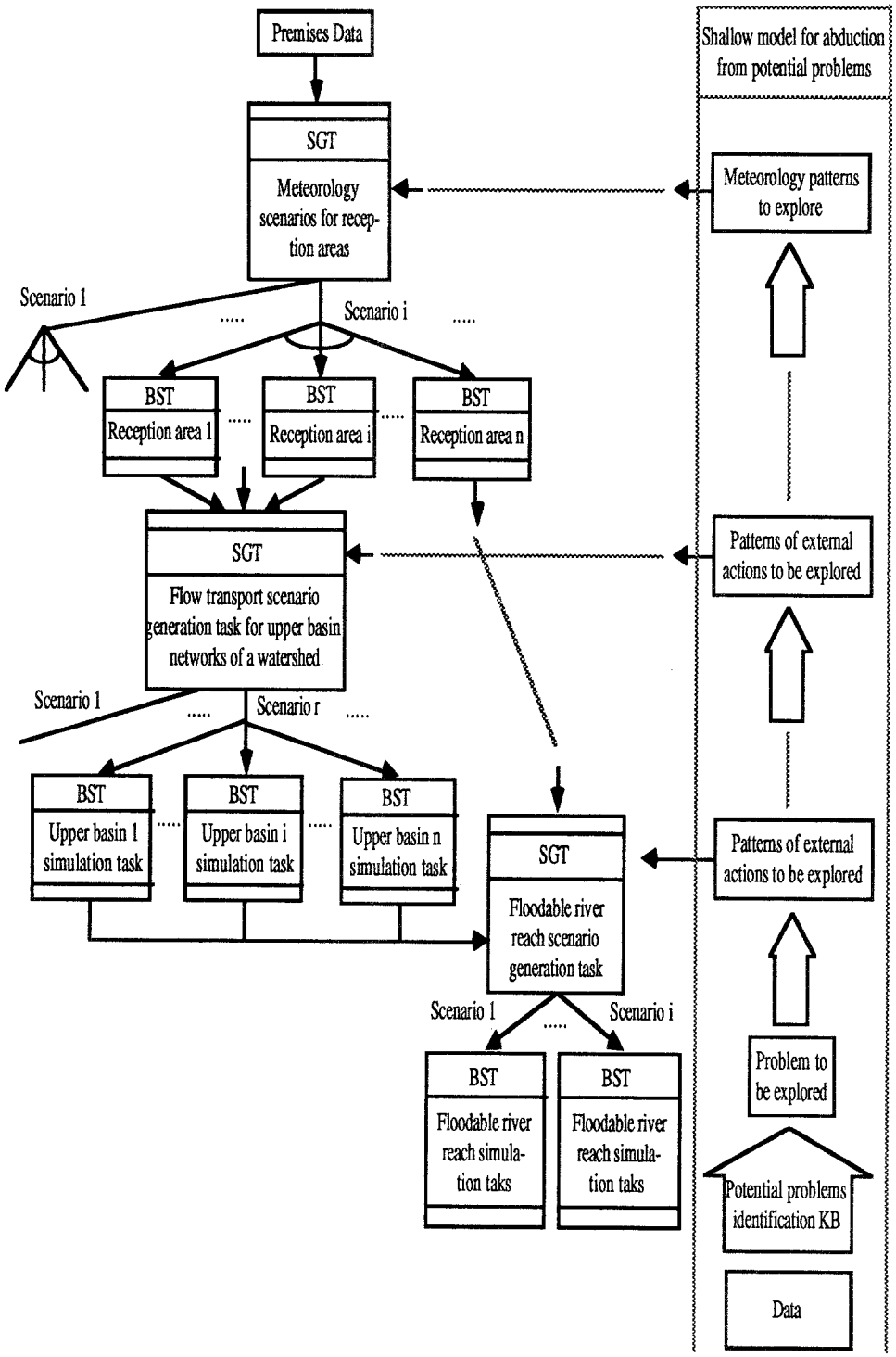


*Figure 9: Reasoning tree for SIRAH behavior prediction*

Building a knowledge model for qualitative simulation of behavior with this inference procedure requires the following steps:

- Formulation of the knowledge base for problem identification and for definition of the patterns of behavior to be explored.

- Instantiation of a set of BST modules representing the different physical elements.

- Formulation of the SGT's and control knowledge base.

- Formulation of the knowledge bases for inference of dams control criteria and civil defense recommendations.
  The SIRAH environment must provide predefined elements and procedures to support this building process. The elements provided by SIRAH are:
  * *A library of task modules for physical behavior modeling* of the three types presented before in such a way that it is possible to have different alternative tasks modules in the library to deal with the same concept. These task modules include formal knowledge bases defined by formal constraints and/or rules relating different attributes. The domains of attribute values and the instantiation of the formal rules to specific rules or constraints relating attribute values must be done by the user at the time of building a model for a case study. A more detailed account is given in section 5.
  * *A framework to build SGT's* with the specified format by the user of the possible types. The user specifies the patterns of final behavior to be considered and scenario variables concerning the task and the two knowledge bases modeling the goal regression and predictable behaviors.
  * *A framework to define the general control base* that produces dynamically modified patterns of future behavior to constraint the scenario generation tasks. The contents of this knowledge base are rules to propose possible new patterns of final behavior taking into account: (1) the patterns to explore proposed previously by the ad hoc knowledge base of problem identification, and (2) the existing predicted behaviors already generated by the inference process. The reasoning procedure proposes additional constraints to the patterns of future behavior in such a way that significantly different behavior with respect to existing ones be generated.
  * *A traditional knowledge representation environment* (frames and rules) to allow the definition of the two ad hoc knowledge bases for:
    .- Identification of present problems and patterns of final behavior to explore.
    .- Future problems and control decisions and civil defense recommendations.
  * *A knowledge acquisition procedure* to guide the user in the creation of a model.
  * *A General inference engine adapted* to the reasoning structure described before using the compiled knowledge structure.
  * *An explanation facility using a trace file* generated by the general inference engine if the trace mode execution is used.
  * *An user interface.*

A first version of this environment has been developed in C language on top of UNIX and tested to deal with the Júcar basin flood problems in Spain.

# 5 CONCLUSIONS

The preceding descriptions show that knowledge based systems are to be understood as complex structures built of knowledge units that may also be structured.

The identification of the more adequate structure and its content for a given type of problem is the challenge to the knowledge based systems designers.

This is a quite different perspective from the narrow approaches based on uniform paradigms built by too basic elements as rules. In this perspective knowledge based systems may be considered as a general paradigm to approach the software design.

# 6 REFERENCES

[Alonso et al. 90] Alonso M., Cuena J., Molina M.: "SIRAH: An Architecture for a Professional Intelligence". Proc.9th European Conference on Artificial Intelligence (ECAI'90). Pitman, 1990.

[Brown, Chandrasekaran, 89] Brown D.C., Chandrasekaran B.: "Design Problem Solving" Pitman. Morgan Kaufmann, 1989.

[Chandrasekaran, 83] Chandrasekaran B.: "Towards a Taxonomy of Problem Solving Types" A.I. Magazine 4 (1) 9-17, 1983.

[Chandrasekaran, 86] Chandrasekaran, B.: "Generic Tasks in Knowledge Based Reasoning: High Level Building Blocks for Expert Systems Design" IEEE Expert, 1986.

[Chandrasekaran, 87] Chandrasekaran B.: "Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks". Proc IJCAI-87. 1183-1192. Morgan Kaufmann, 1987.

[Cuena, 83] Cuena J.: "The Use of Simulation Models and Human Advice to Build an Expert System for the Defense and Control of River Floods". (IJCAI-83). Karlsruhe. Kaufmann, 1983.

[Cuena et al, 91] Cuena J., Molina M., Garrote L.: "An Architecture for Cooperation of Knowledge Bases and Quantitative Models: The CYRAH Environment". XI International Workshop on Expert Systems. Special conference on Second Generation Expert Systems. Avignon'91. EC2, 1991.

[De Kleer, Brown, 84] De Kleer, J., Brown, S.: "A Qualitative Physics Based on Confluences" Artificial Intelligence 24. Elsevier Science Publishers B.V. (North Holland) 1984, 7-83.

[Fikes, Nilsson, 71] Fikes R.E., Nilsson N.J.: "STRIPS: A New Approach to the Application of Theorem proving to Problem Solving" Artificial Intelligence 2 (3-4) 189-208, 1971.

[Forbus, 84] Forbus K.D.: "Qualitative Process Theory" Artificial Intelligence 24, 85-108, 1984.

[Forbus, 90] Forbus K.D.: "The Qualitative Process Engine" in "Readings in Qualitative Reasoning about Physical Systems" D.S. Weld and J. de Kleer eds. Morgan Kaufmann, 1990.

[Hayes, 79] Hayes P.J.: "The Naive Physics Manifesto" Formando parte de: Michie D. (ed) "Expert Systems in the Microelectronic Age". Edimburgh University Press, 1979.

[Kuipers, 86] Kuipers B.J.: "Qualitative Simulation" Artificial Intelligence 29, 289-338.

[Laird et al., 87] Laird J.E., Rosenbloom P., Newell A.: "SOAR: An Architecture for General Intelligence" Artificial Intelligence 33, 1987.

[Loveland, 79] Loveland D.W.: "Automated Theorem Proving a Logical Basis". Elseviere North Holland, 1979.

[Luckham, 68] Luckham D.: "Refinement Theorems in Resolution Theory" Symposium on Automatic Demostration. Versalles, 1968.

[Newell, 82] Newell A.: "The Knowledge Level" In *Artificial Intelligence* Vol 18 pp 87-127.

[Newell, Simon, 63] Newell A., Simon H.: "GPS: A Program that Simulates Human Thought". en "Computers and Thought" Feigenbaum y Feldman (eds). Mc Graw Hill, 1963.

[Robinson, 65] Robinson J.A. "A Machine Oriented Logic Based on the Resolution Principle" Journal ACM (12, 1), 1965.

[Steels, 87] Steels, L.: "The Deepening of Expert Systems" AI communications, vol 0, no.1. 1987.

[Wielinga et al, 88] Wielinga B.J., Breuker J.A., Bredeweg B.: "Knowledge Acquisition for Expert Systems" chapter in "Advanced Topics in Artificial Intelligence" R.T. Nossum ed. Springer Verlag, 1988.

[Winston, 87] Winston P.H.: "The commercial Debut of Artificial Intelligence" in "Applications of Expert Systems" Quinlan J.R. (ed) Addison Wesley, 1987.