The Verification and Validation of Fuzzy Knowledge in Planning Systems

*VASILE MAZILESCU, **DANIELA SARPE, *CORNELIA NOVAC-UDUDEC
Department of Economic Informatics, **Department of Economics
Dunarea de Jos University of Galati, 47 Domneasca Street
ROMANIA
vasile.mazilescu@ugal.ro, daniela.sarpe@ugal.ro, cornelia.novac@ugal.ro

Abstract - This paper argues that verification and validation (V&V) techniques are an essential part of the knowledge management process. However, an examination of known studies on the effectiveness of existing KBS V&V techniques shows that the state of knowledge in this area is very restricted. It has recently grown more and more accepted that there is a need for incorporating aspects of time and imprecision into knowledge based systems, considering appropriate semantic foundations. Problem solving can be seen as a process consisting of problem space search and knowledge search. Our expert system is a special possibilistic expert system, developed in order to focus on fuzzy knowledge.

Key-words: Predictability, Fuzzy reasoning, Possibilistic expert control system, Validation, Verification

1 Introduction
Knowledge-based systems (KBS) have proven to be an effective technology for solving many kinds of problem in business and industry. KBS succeed in solving problems where solutions are derived from the application of a substantial body of knowledge, rather than by the application of an imperative algorithm. In the 1980s, KBS technology was widely applied to solve stand-alone problems. Classic examples of the successful use of the technology were in diagnostic problem-solving (in medicine or engineering), provision of advice (for example, in "help-desk" applications), and construction /configuration (for example, product manufacturing and transportation loading). In the 1990s, many organisations have identified their collective knowledge as their most important resource, and are applying KBS technology to capture and exploit these "knowledge assets" in a systematic manner.

The characteristic feature of problem domains where KBS technology is suitable is that the problems are ill-defined: they are not amenable to solution by algorithmic means; instead, the knowledge in the knowledge base of the KBS is used in some way to search for a solution. Often, the domain is such that there can be no guarantee that a solution will be found, or that found solutions will be optimal. Many KBS offer a "best effort" solution, which is good enough when the application requirements permit this (that is, the system is not safety or mission-critical). Therefore, there are three large groups of problems, which Artificial Intelligence (AI) should approach in terms of control: numerical-symbolic interface, qualitative modeling and time management. Such applications obviously require dated event operations the life time of which should be managed by the system which often works asynchronously with the acquisition and control system. Time restrictions are not excessive in usual industrial applications. Critical time reasoning problems may occur in case of faulty operations and overloading. For example, controlling the power systems is a complicated task. Operators are continually monitoring incoming system data and making decisions to ensure the production requirements. In most systems, operators get this raw data through embedded software. If a fixed threshold is crossed, the software triggers an alarm. Conventional expert system shells are too slow for real-time environments, and their inference process is unbounded. We need a reactive, interruptible system that can assimilate data and asynchronous events, and present the operator with a reasoned opinion in a timely manner. Speed alone is not enough. A real-time expert system shell must also represent imprecise, time and temporal data, encode temporal knowledge, and manage temporal/fuzzy reasoning.

We have developed an object-oriented fuzzy real-time expert system shell. We will investigate the relations between fuzzy rules and its temporal characteristics, the fuzzy condition/fact pattern matching, the fuzzy constants compatibility, the linking of variables, etc. The importance of verification and validation process has been specified in section 2. To illustrate the theoretical results we present in section 3 a Knowledge Management System (as a Planning System) that can be seen as a
The combination of methods and measurement: the methods used in requirements specification, knowledge acquisition, system design, and system implementation result in the production of a series of artifacts, each of which is amenable to some form of measurement (either individually or in combination). An example of fuzzy control model based on metalevel knowledge for load balancing problem in a specific structural definition is made. Section 4 present concluding remarks. Comparisons to relevant research are made throughout the paper.

2 The Importance of Validation-Verification and Predictability

Verification can be viewed as a part of validation: it is unlikely that a system that is not "built right" to be the "right system". However, verification is unlikely to be the whole of validation, due to the difficulty of capturing specifying user requirements. As noted above, this is a particularly important distinction in knowledge management. Of course, the goal in software/knowledge engineering is to try to ensure that the system is both "built right" and the "right system"; that is, the goal is to build "the right system, right". In software engineering, efforts have been made to formalise the development process so that user requirements may be stated as a fully-formal specification, from which it can be proven that the implemented software system meets the requirements. While formal methods are desirable - even essential - in some cases (notably safety and mission-critical systems), these methods are unsuitable in large classes of software applications:

1. Where requirements are amenable to formal specification, it may be too difficult to create the specification within project time and budgetary constraints.

2. There are many kinds of requirement that are not amenable to formal specification (for example, the "usability" of a graphical user interface).

The extent to which formal methods can be applied in knowledge management is debatable, but it is certainly unrealistic to expect formal verification to serve as the only V&V technique in a KBS development project, because it will rarely be possible to ensure that the formal specification is a complete and correct statement of the users’ requirements. Therefore, KBS V&V will typically need to involve multiple techniques, including formal verification against formal specifications (where possible), and empirical validation (including running test cases and evaluating the system in the operational environment). Given that knowledge management is an inexact art, the most fundamental measures of the success of a KBS project would seem to be: Did we get it right? That is, does it meet the users' actual requirements. Can we keep it right? That is, is it sufficiently maintainable for anticipated future changes. Can we do it again? That is, is the process repeatable to ensure success with future projects. The final point refers to the capability of the knowledge engineers, and reflects the modern view of software quality being determined primarily by the quality of the development process. While verification and validation are only part of the overall development process, they are extremely important because they are the only way to produce an answer to the first of the three questions above ("Did we get it right?") and, provide partial answers to the other two questions (V&V techniques assist in measuring maintainability, and a repeatable V&V capability is a prerequisite for success in knowledge management). Consideration of the importance of V&V to successful knowledge management raises another question: how effective are the KBS V&V techniques in current use? Obviously, if the techniques are incomplete or unsound, then they cannot be trusted to provide measurement of software quality and project success. The goal of this paper is to reflect upon studies which have been done to assess the effectiveness of current KBS V&V techniques, and to:

- summarise what the studies tell us about the current state-of-the-practice in KBS V&V;
- identify ways to improve the state of knowledge engineers’ own knowledge about available KBS V&V techniques.

Real-time systems span a broad spectrum of complexity from very simple microcontrollers to highly complex and distributed systems [9]. These complex future systems include the space station, integrated vision/robotics/AI systems, collections of human/robots coordinating to achieve common objectives (usually in hazardous environments), and various command and control applications. To further complicate the problem there is many dimensions along which real-time systems can be categorized. The main one includes: the granularity and the strictness of the deadlines, reliability requirements of the system, the characteristics of the environment in which the system operate. The characteristics of the environment, in turn, seem to give rise to how static or dynamic the system has to be. However, one common denominator seems to be that all designers want their real-time system to be predictable. It means that it should be possible to show, demonstrate, or prove that requirements are met subject to any assumptions made, for example, concerning failures and workloads. In other words, predictability is always subject to the underlying assumptions being made.

The previous aspects emphasize the importance of V&V as measurement techniques for the knowledge
management process. Knowledge management (and software engineering) can be seen as a combination of methods and measurement: the methods used in requirements specification, knowledge acquisition, system design, and system implementation result in the production of a series of artifacts, each of which is amenable to some form of measurement (either individually or in combination). V&V techniques provide the means of obtaining the measurements. The following artifacts are of particular importance in the KBS development process [1,8,10]:

Requirements Specification. The requirements specification document states the minimum and desired user requirements, typically in natural language (or, less usually, in some restricted or semi-structured natural language subset). As a natural language document, the requirements specification is not amenable to analysis by V&V techniques - instead, it is used to establish the needs for V&V.

Conceptual Model. The conceptual model describes the knowledge content of the KBS in terms of real-world entities and relations. This description is entirely independent of the ways in which the KBS may be designed or implemented: the idea is to allow the knowledge engineer to perform a knowledge-level (epistemological) analysis of the required system before making any design or implementation choices. The best-known framework for defining KBS conceptual models is KADS, in which models may be initially defined using a semi-formal, largely diagrammatic representation, from which a refined, formal model can be derived. The conceptual model forms the basis of the design model.

Design Model. The design model serves to "operationalise" the conceptual model into an executable KBS; it describes the required system in terms of computational entities: data structures, processes, and so forth. For example, the design model may specify that a particular conceptual task is to be performed by a backward-chaining search, or that a concept taxonomy is to be represented using a frame heirarchy. The KBS specification language KADS (Wielinga, et. al., 1992) is particularly well-suited to the representation of design models. The design model dictates the form of the implemented system.

Implemented System. This is the final product of the development process: the KBS itself. Once the design issues have been explored in the design model, the system may be implemented in any programming language, although typically a special-purpose KBS language is used.

There are many V&V techniques that have been developed for use on KBS. Five of the most common approaches are listed below.

Inspection. According to a survey of developers of KBS in business applications, inspection is the most commonly-employed V&V technique. Arguably, it is also the least reliable, as it essentially involves nothing more than human proof-reading the text of the various artifacts. Typically, a domain expert is asked to check the statements in the knowledge base; since the formal languages used in the design model and implemented system will be unfamiliar to domain experts, this technique is better-suited to use with the semi-formal conceptual model (which will typically use a more "reader-friendly" graphical representation).

Static Verification. Static verification consists of checking the knowledge base of the KBS for logical anomalies. Frameworks for anomalies in rule-based KBS have been well-explored, and software tools exist to detect them. The most commonly-identified anomalies - and the ones detected by most of the available tools - are redundancy and conflict. Redundancy occurs when a knowledge base contains logical statements that play no purpose in the problem-solving behaviour of the system; this typically indicates that the system is incomplete in some way.

Conflict occurs when there are logical statements that are mutually inconsistent, and would therefore cause the system to exhibit erroneous behaviour. Anomalies may exist in any of the formal artifacts: the implemented system, the design model, and - if it is defined formally - the conceptual model.

Formal Proof. Formal proof is a more thorough form of logical analysis of the (formal) artifacts in the development process than that provided by static verification. Proof techniques can be employed to verify that the formal artifact meets the specified requirements. In practice, however, while there are many formal specification languages for KBS, there are few documented examples of the use of proof techniques to very user requirements.

Cross-Reference Verification. When there exists descriptions of the KBS at different "levels", it is desirable to perform cross-checking between these, to ensure consistency and completeness. For example, we would expect the concepts that are specified as being required at the conceptual level to be realised in terms of concrete entities at the design level, and in terms of concrete data structures in the implemented system. The most appropriate uses of cross-reference verification are to check correspondence between: conceptual model and design model; design model and implemented system.

Empirical Testing. All software testing involves running the system with test cases, and analysing the results. The software testing literature distinguishes between function-based testing and structure-based testing. Function-based testing bases the selection of test cases upon the functional requirements of the system, without regard for how the system is implemented.
The success of function-based testing is dependent upon the existence of a "representative" set of test cases. In structure-based testing, test cases are selected on the basis of which structural components of the system they are expected to exercise; the objective is to show that the system produces acceptable results for a set of test cases that exercise all structural components of the system. Testing can be applied only to the executable artefacts: typically only the implemented system. In particular, little is known about the relative effectiveness of V&V techniques in object-oriented software development.

3 A Fuzzy Planning System
In this section we concentrate on predictability with respect to the timing requirements, the basis of validation and management the fuzzy control model. The closed-loop control expert system can be modelled like a nondeterministic state machine, whose outputs are the process outputs. A major obstacle to the widespread use of (possibilistic) expert systems in real-time domains is the non-predictability of rule execution time. Systematic analysis methods must be used so that the possibilistic expert system behaviour can be studied quantitatively within the developed modelling framework. The relationships and the analogy between expert and control system architectures are important problems for intelligent control [7]. This is possible because both are problem solving systems with different problem domain (environment) the expert system reasons about and takes actions on. The problem domain must be defined as a collection of problems that the expert system desires to solve. In conventional control, the plant is a dynamical system, described with linear or non-linear differential / difference equations. An artificial intelligent expert system consists of the planner or the inference engine, the problem domain, the exogenous inputs, and their interconnections. The outputs of the expert system are the inputs (control actions) to the problem domain. There are unmeasured exogenous inputs to the problem domain (disturbances) that represent specific uncertainty. The measured exogenous input to the expert system is the goal. An expert system can be modelled using predicate or temporal logic or other symbolic techniques such as finite state machine. Artificial intelligent feedback expert systems are analogous to conventional feedback control systems that do not use state estimation (they do not use situation assessment). A fuzzy real-time expert systems must represent imprecise, time and temporal data, encode temporal knowledge and manage temporal/fuzzy reasoning. Following a conventional control-theoretic approach, we can introduce a mathematical model for the plant P and the possibilistic expert control system (PECS), which consists of the possibilistic expert system (PES) and the plant [5,6]. The PES must be designed so that it can coordinate the use of the plant outputs and reference (user) inputs, to decide what plant command inputs (or hypothesis/conclusions) to generate so that the closed-loop specifications are met. Although the PES (viewed as an expert system) are frequently being used to perform complex control functions, most often it is the case that no formal analysis of the dynamics is conducted because mathematical analysis of such systems is often considered to be beyond the scope of conventional control theory. It is assumed that the plant can be represented with the following model:

\[ P=(X, E, f, \delta, g, E_o) \]

that can represent certain class of discrete event system, where \( X \) is the set of plant states denoted by \( x \), \( E \) is the set of all events, \( f \) are the state transition map, \( f: X \rightarrow X, c \in P(E), k \in T, \delta \) are the output maps, \( g \) is the enable function, \( g: X \rightarrow P(E) \), and \( E_o \) is the set of all valid event trajectories (that are physically possible). Note that \( E \) is the union of the command-input events (\( E_c \)), the disturbance input events (\( E_d \)) and the output events (\( E_o \)) of the plant. When discussing the states and events at time \( k \), \( k \in T \) or \( k \) is a fuzzy instant or a fuzzy time interval, \( x_k \in X \) is the plant state, \( c_k \in E_c \) is a command input event of the plant, \( c_k \in E_d \) is a disturbance input event of the plant, \( c_k \in E_o \) is an output event of the plant, that is equal to input event \( c_k \in E_p \) for PES. Each \( c_k \subseteq g(x_k) \) is an event that is enabled at time \( k \), and it represents a set of command and disturbance input events of the plant. If an event \( c_k \in E \) occurs at time \( k \) and the current state of plant is \( x_k \), then the next state is \( x_{k+1} = f(x_k) \) and the output is \( c_k \delta = c_k \delta = \delta_k(x_k) \). Any sequence \{\( x_k \)\} such that for all \( k, x_{k+1} = f(x_k) \), where \( c_k \subseteq g(x_k) \) is called a state trajectory. The PES has two inputs: the reference input events \( c_k \in E_p \) (user inputs) and the output events of the plant \( c_k \in E_p \). Based on its fuzzy state and these inputs, the PES generates enable command input events to the plant \( c_k \in E_p \). This inference loop constitutes the core of the PES where the knowledge is interpreted by the inference engine, actions are taken, the fuzzy factbase is updated and the process repeats. Usually, the fuzzifier may transform the measured value \( c_k \) of the sensor measurement into a corresponding universe of discourse for each input variable, as an input fuzzy fact. Fuzzy rules \( R \in R \), are used to express knowledge. Three kinds of variables are used: input, output and intermediate variables. The defuzzification process decides for each output variable a single value. The PES is modeled by:
Plant feedback and the reference input variables

where $X_{\text{PES}} = X^b \times X^\text{int}$ is a set of fuzzy PES states $X_k$ ($X^b$ is the set of fuzzy factbase states and $X^\text{int}$ is the set of possibilistic inference engine fuzzy states), $E_{\text{PES}}$ is the set of events of the PES (reference inputs $E_{\text{r}}$, user inputs, output PES events $E_{\text{PES}}^0$, the set of fuzzy rules $R$ and the plant output events $E_{\text{PES}}^p$), so that:

$$E_{\text{PES}} \subset \mathbb{P} \quad (E_{\text{PES}}^f \cup E_{\text{PES}}^\text{UI} \cup U \cup E_{\text{PES}}^0 \cup HC)$$

are the state transition maps, $\delta_{\text{PES}}$ is the output map of PES and $E_{\text{PES}}^v$ is the set of all valid inference loop trajectories that are possible. It is assumed that an occurrence of a command input event to the PES, $e_{\text{PES}}^0 \in E_{\text{PES}}^r$ is always accompanied by a firing of enabled instantions rules $R_i \in R$, $i=1,...,n$, so that the fuzzy inference loop can be updated accordingly. Similarly, the firing process of fuzzy rules cannot be active alone, because the inference loop is updated only if there is a change in the plant reflected via its outputs, or a change in the reference input event $e_{\text{r}} \in E_{\text{PES}}^r$, or user inputs. It can control the hypothesis/conclusions for the user decision or the enabling of the command input events of the plant ($c_{ak} = c_{ak}$). The input events inclusion in the fuzzy rulebase $R$ allows the PES designer to incorporate the plant feedback and the reference input variables directly as parts of the fuzzy rulebase. This is analogous to the use of variables in conventional rule-based expert systems.

It is important to note here that the consequent formulas of the rules represent how the fuzzy state $x^b$ in the fuzzy factbase changes, based on the occurrence of input events, and they can be defined in a recursive manner. We can define the conclusions on $X^b \times E_{\text{PES}}$ or on $X^b \times E_{\text{PES}}^f \times X^\text{int}$ so that the fuzzy rules could characterize changes made to the inference strategy. An event $e_k \in \{R_i \in E_{\text{PES}}^r\} \subset g_{\text{PES}} \quad (x^b_k) \quad \text{can possibly occur}$ if the full premise of $R_i$ evaluates satisfactorily at time $k$, for the given state $x^b_k \in X^b$ and the command input event $c_{ak}$. Then, after the event's occurrence, the next state $x^b_{k+1} = f_{e_k} \quad (x^b_k) \quad \text{is given by the application of the conclusion (taking into account their time of truth)}$ to the fuzzy state $x^b_k \in X^b$ to produce $x^b_{k+1}$ and by updating the inference engine state $x^\text{int}$. In this way it is evident that fuzzy decision-making capabilities of the PES are more sophisticated than those of the standard fuzzy controllers. The PECs has to be designed so that it can eliminate the undesirable closed-loop system behaviours. There is a need to specify the initial state of the closed-loop system to reduce the insignificant state combinations that may unnecessarily complicate the model. The fuzzy knowledge-based system requires to adapt the representation of a knowledge in order to operate it and to improve the efficacy of its operating using the compilation technique. The accepted data are: variables, atomic and fuzzy constants. The fuzzy constants may appear both in facts and in rules and are always associated with a fuzzy set through the constfaz function. The possibility distribution modeling allows a unified framework for the representation of imprecision and uncertainty [2].

The operation of the fuzzy expert system proceeds by the following steps:

1. Acquiring the plant output and reference input events at time $k$;
2. Forming the conflict set in the fuzzy match phase from the compiled set of rules in the fuzzy knowledge-base and based on $c_{ak}$, the current status of the truth of various fuzzy facts, and the current values of variables in the knowledge-base;
3. Using conflict resolution strategies (refraction, recency, distinctiveness, priority, and arbitrary) in the select phase to find one rule $r$ to fire;
4. Executing the actions characterized by the consequent of rule $r$ in the act phase.

The timing of the event occurrences is such that the PES is synchronous with the plant. Although every occurrence of an input event of the plant always affects the expert system state, the occurrence of an input event of the expert system does not necessarily immediately affect the plant state. In qualitative analysis of our fuzzy expert system, i.e. the system validation, the focus is on testing if the plant, expert system, and especially the closed-loop PECS satisfy certain properties, as follows: reachability, ciclic properties and stability. We can also analyze the properties of the isolated fuzzy expert system (i.e., without the plant). In our case the "plant" is fuzzy compiled knowledge-base, the "fuzzy expert system" is the fuzzy inference engine, the "command inputs" are the changes that the inference engine makes to the knowledge-base, and the "outputs" of the closed-loop system are fuzzy facts or variables in working memory (that the inference engine uses in its decision-making process).

4 Conclusions

We have shown that conventional knowledge-based debugging tools can ignore important dynamic behavior that can result from connecting the full fuzzy expert system (i.e., with an inference engine) to user inputs and a dynamical process. The results of
this paper shows that fuzzy expert control system are a class of (heuristically constructed) nonlinear control systems that can be studied with the analytical tools available from conventional control theory. Current research in real-time artificial intelligence is driven by a need to make knowledge-based systems function in real-time, to be predictable, and a need to integrate approaches to handle non-linearities. Response time analysis is in general undecidable, and is PSPACE-hard in the case where all the variables have finite domain [3, 4]. It is hoped that the work reported in this paper serves to promote the development of a firm mathematical foundation on which to perform careful analysis for the verification and validation of the dynamics of expert control systems that operate in critical environments. The overall conclusion from the studies is that the collective knowledge on the effectiveness of KBS V&V techniques is very limited. There is some evidence that different techniques have complementary effectiveness, and no technique has been shown to be so weak as to be not worth employing. However, the data that is available is sparse, being limited to a few instances of KBS and specific applications of tools or techniques. In order to improve the collective state of knowledge on the effectiveness of KBS V&V techniques, it is necessary to perform a considerably larger set of studies. In order to gather a sufficiently complete data set, the following process would need to be followed:

1. Create a sufficiently complete enumeration of the types of KBS requiring V&V. For each type of KBS, create instance artefacts at each stage of development (conceptual model, design model, and implementation), and for each development method. For example, instances of KBS with various distinct problem-solving methods would be required, and artefact instances would need to be created using different methods (and representation languages).

2. Define reference implementations for each V&V technique, either in the form of well-defined manual procedures, software tool specifications/implementation, or a combination of the two. Where necessary, variations on the V&V techniques will need to be defined for different representations used in the reference KBS artefacts produced in Step 1.

3. Define good fault models, based on observed error phenomena from actual experience in KBS projects.

4. Mutate the KBS artefacts from Step 1 using the fault models from Step 3 (ideally, this would be done automatically); then apply each of the V&V techniques defined in Step 2 to each mutated artifact; repeat for a statistically-significant set of mutated artefacts.

Such a study would be very ambitious but extremely valuable: it would provide conclusive evidence as to the effectiveness of each V&V technique for each type of KBS and development method, individually and in combination. Furthermore, it would support further research and development of KBS V&V techniques. Of course, such a study would be very difficult: Step 1 and Step 3 in particular are made hard by the fact that KBS technology is moving constantly forward: new kinds of KBS are always emerging - for example, witness the current interest in multiple-agent KBS and reliable information on actual error phenomena is had to come by (partly because knowledge engineers do not wish to advertise failures). It is worth noting, however, that the artefacts created in Step 1 would be of wider use that merely in a study of V&V techniques - they could facilitate complementary studies on the effectiveness of knowledge acquisition and design methods.

References: