ELSEVIER

Knowledge-Based Systems 16 (2003) 165–171

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Knowledge base management systems-tools for creating verified intelligent systems

Richard C. Hicks*

Department of MIS and Decision Science, Texas A&M International University, Texas, USA

Received 8 December 2000; revised 8 April 2002; accepted 17 October 2002

Abstract

As automation of business processes becomes more complex and encompasses less-structured domains, it becomes even more essential that the knowledge used by these processes is verified and accurate. Most application development is facilitated with software tools, but most business rules and expert systems are developed in environments that provide inadequate verification testing.

This paper describes an emerging class of applications we refer to as Knowledge Base Management Systems (KBMS). The KMBS provides a full life-cycle environment for the development and verification of business rule and expert systems. We will present an overview of knowledge base verification, the KBMS life-cycle, and the architecture for a KBMS. We then describe building a small expert system in the KBMS, with emphasis on the verification testing at each stage. We conclude with a summary of the benefits of a KBMS.

Keywords: Verification; Expert system; Computer aided software development; Rapid application development

1. Introduction

In many industries, the key to efficiency is automation. The first targets for automation were the most structured problems, such as accounting. Our ability to automate less structured domains is constrained by our ability to verify the knowledge used in the automation. Automation of less-structured domains is achieved with active intelligent components such as expert systems or business rule systems. We will refer to these systems in general as knowledge-based systems.

There are many difficulties in building a useful knowledge-based system, including difficulty in capturing deep knowledge, lack of robustness and flexibility, inability to provide deep explanations, difficulties in verification, little learning from experience [7], and computational efficiency. Knowledge-based systems are harder to build than most people perceive them to be because of the dependencies between rules in the system and the difficulty of verifying them.

The importance of verification in knowledge-based systems cannot be overstated. A single bad rule in a medical expert system could kill a patient, just as a single bad rule in a business system could put the company out of business. As we automate more and more processes, the need for verification becomes even more critical. Many automated process can perform incorrectly for a long time, as no person is responsible for checking the process.

In a survey of 40 knowledge-based system tools conducted in 1997, Murrell [9] concludes "The paper provides… areas in which the researcher can provide practitioners with valuable tools for the verification and validation of knowledge-based systems, where currently there are none…" It should also be noted that the systems surveyed were all research systems, and that none are available to the general public.

Business rules are usually created in a text editor as program code or database triggers, making the programmer responsible for verifying the program logic. Expert systems are usually built in a vendor-supplied tool that translate specifications into code, but may provide little or no verification testing of the specifications. At least one expert system development tool from a major manufacturer will allow duplicate and conflicting rules to be created, and none verify the application for all 23 verification criteria.

However, articles about early examples of Knowledge Base Management Systems (KBMS) software have begun

* Corresponding author. Tel.: +1-(888)-327-9397.
E-mail address: rick@ez-xpert.com (R.C. Hicks).

0950-7051/03/$ - see front matter © 2003 Elsevier Science B.V. All rights reserved.
PII: S0950-7051(02)00082-5

to appear. Aquinas, in production use at The Boeing Company, offers many of the functions desirable in a KBMS. It elicits knowledge directly from the expert into grids, which are analyzed for completeness and consistency. It refines the specifications and generates code for several expert system shells. Complete applications may be created in less than two hours [2].

EULE is a system developed by Swiss Life that has functionality 'in the triangle of Knowledge Representation, Business Process Modeling, and Knowledge Management.' This system is designed to automate office tasks in the insurance industry, and uses an extendable High Level Language (HLL) to model characteristics such as laws, regulations, and preconditions for activities. The resulting system is integrated into Swiss Life's Organizational Memory systems, and it is suitable for embedding in business process models [13].

The purpose of KBMS is to offer computerized assistance for building knowledge-based systems. A KBMS:

1. Provides full life-cycle support from knowledge acquisition to delivered code.
2. Guides the user through the development cycle.
3. Detects or prevents verification errors.
4. Algorithmically refines knowledge.
5. Generates code for the knowledge-based system.

We will first consider verification of the knowledge-based system because of its influence on KBMS life cycle and the KBMS architecture.

2. Verification of rule-based systems

One of the greatest challenges in building a substantial knowledge-based system is verification. XCON, a well-funded, strategically important knowledge-based solution, has a 95% reliability rating in a deterministic domain [1]. In an editorial in AI Expert, Eliot reported on an informal survey he made that examined delivered expert systems. These systems covered between 60% and 95% of the search space [3].

Previous efforts at expert systems verification, such as ONCOCIN [15], CHECK [10], and EVA [14], rely on a heuristic approach to verification; they perform tests on the completed code to determine if any symptoms of verification problems exist. As they consider the rule base to be a monolithic mass of knowledge, none of these systems can test for completeness because of combinatorial explosion. Preece's system, COVER, can test for completeness. However, it tests partitions of the rule base with test cases instead of testing at the rule cluster level [12].

The Two-Tier Verification (TTV) approach exhaustively verifies a knowledge-based system rule base for 23 criteria [4]. TTV manages computational complexity by partitioning both the rule base and the verification criteria. The rule base is partitioned into rule clusters, grouping them by their actions. Verification is partitioned into two levels of tasks. Global Verification verifies that the structure of the rule base, while Local Verification analyzes each individual rule cluster. In other words, we analyze the individual systems (the rule clusters) and the linkages between them. The classes of Global Verification Criteria are reachability and domain constraints. The classes of Local Verification Criteria are completeness, consistency, conciseness, and domain constraints. As heuristics, business rules are not subject to completeness testing, but all other verification criteria apply.

TTV supports exhaustive verification of all the major criteria, making it an appropriate foundation for the KBMS. In Section 5, we will demonstrate the life cycle with the prototype KBMS and show how verification is integrated into the life cycle. First, we will present the KBMS life cycle and an overview of the architecture.

3. KBMS life cycle

The KBMS is designed to complement existing and future delivery environments by providing a full life-cycle development environment that generates ready to run code for multiple implementation platforms. The KBMS life-cycle is presented in detail in Ref. [4], and consists of an iterative cycle of Definition, Rule Construction, Refinement, Testing, Delivery, Evaluation, and Maintenance.

3.1. Definition

The definition stage consists of two components. First, we must determine the scope of this iteration of the project in terms of what knowledge is to be added to the system. Secondly, we will define the structure of the new project to the KBMS and populate the Knowledge Dictionary.

A KBMS defines a rule base not as a monolithic mass of code or rules, but as a well-formed set of related rule clusters. A rule cluster consists of a well-formed set of conditions that logically imply a conclusion (or, if necessary, conclusions).

The structure of the rules must meet the following criteria. Any conditions that are instantiated by user input or other data sources are terminals. All other conditions must be instantiated by another rule cluster that contains the condition as an action. Referential integrity must hold. The rule base must contain a goal [4].

When a preliminary structure has been created, ideally in a knowledge map, the structure should be analyzed for opportunities for maintenance anomalies [5]. In a well-formed rule cluster, each condition is independent of the others, and every condition is required in at least one rule to determine each of the conclusions. If a condition is dependent on another condition, this relationship should