Knowledge-Based Process Specification Language

A Specification and Validation Technique for Knowledge-Based Systems

by

G.B. Prabhat and P. Srinivasan

Sundram Information Systems
(A Division of Sundram Fasteners Limited)
98-A, III Floor, Dr. Radhakrishnan Salai, Madras 600 004.
Tel: (+91)-44-844350
Fax: (+91)-44-844699

Abstract

This paper is the third in a series of papers to be presented on the KITS Methodology™. In this paper, we focus on the verification and validation issues of Knowledge-Based Systems with particular reference to *Knowledge-Based Process Specification Language*, a shell-independent technique we had developed for the formal specification and subsequent validation of Knowledge-Based Processes.

Introduction

As a professional organization involved in the development of Knowledge-Based intelligent systems for clients from various industry sectors, we observed the need for a shell-independent, standard scheme for specifying and validating Knowledge-Based Systems. We observed that specifying Knowledge-Based Systems using conventional specification techniques like Structured English and Flowcharts was inadequate. Therefore, as part of the KITS Methodology - the methodology we use to develop our brand of intelligent systems, namely Knowledge-integrated Information Technology Solutions (KITS™) - we have developed a tool-independent formalism to specify and subsequently validate Knowledge-Based Processes.

Knowledge-integrated Information Technology Solutions

The term Knowledge-integrated Information Technology Solutions (KITS) indicates an integrated Knowledge-Based System architecture as shown in Figure 1. The KITS architecture integrates frontier technologies like Knowledge-Based Systems and Hypertext with conventional ones like Database Management Systems, Procedural Computing, Spreadsheets and Graphics. Since Knowledge-Based Systems form a major part of the KITS architecture, Verification and Validation are vital issues that are addressed in the KITS Methodology.

KITS Methodology - An Overview

The KITS Methodology is a comprehensive approach to building KITS. It is built on the basic premise that existing models for project management like the Waterfall Model and techniques like Data Flow Modelling are inadequate for the professional management of building intelligent information systems. The KITS Methodology blends the tenets of the Waterfall Model and the Prototyping paradigm providing a unique approach to building intelligent Information Systems.

According to the KITS Methodology, an application based on the KITS architecture is built in four phases, viz., Project Inception, Application Design, Prototyping and Project Completion as shown in Figure 2.

Project Inception

Project Inception consists in evaluating the problem through a feasibility study employing special methods for domain evaluation. Apart from evaluating the domain, one of the major problems associated with building intelligent systems of this type which incorporate expert knowledge is determining the complexity of the project in terms of time, people and cost. Project Inception helps tackle these issues. The end products of Project Inception are the *Functional*

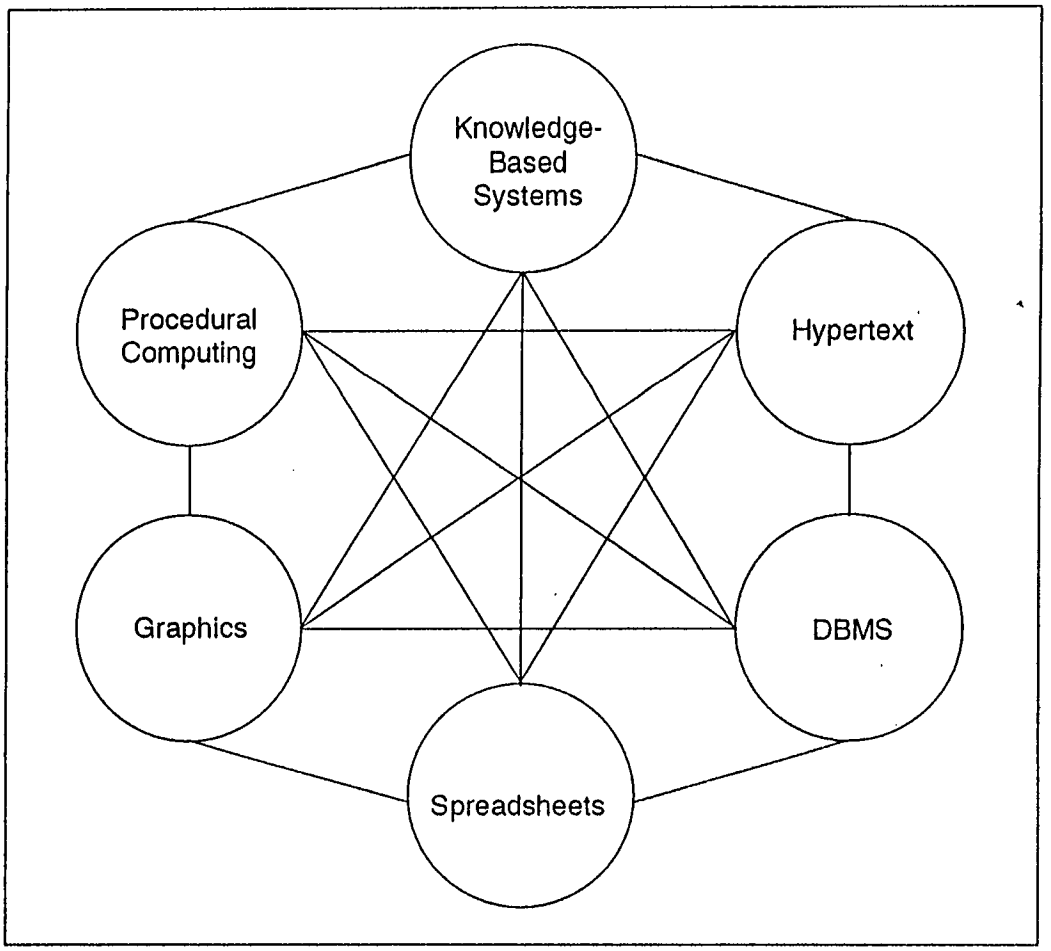

Figure 1. The KITS Architecture.

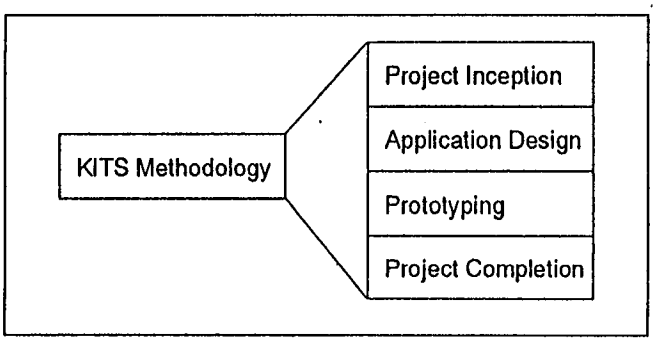

Figure 2. The KITS Methodology.

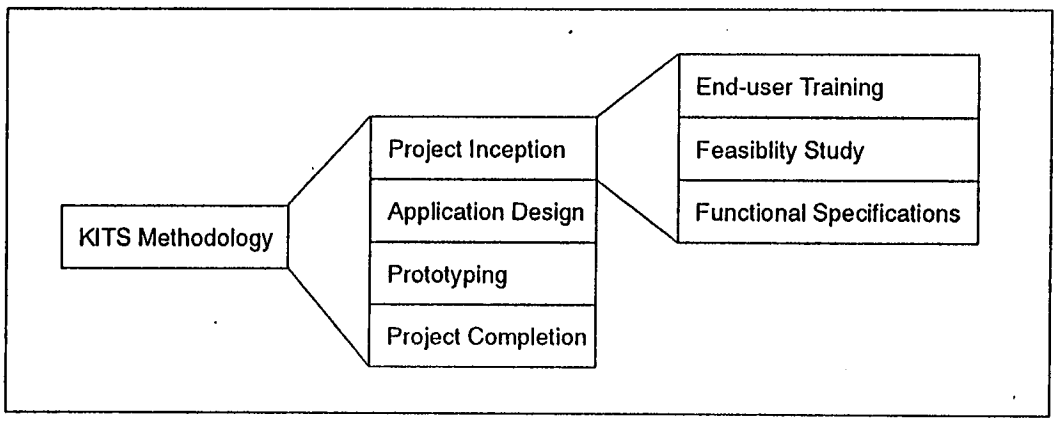

Figure 3. The KITS Methodology - Project Inception.

Specifications and the *Project Plan* documents. The functional specifications of the system are captured using an Information Flow Diagram (IFD™) based on a technique called *Information Flow Modelling* [Prabhat et al, 1991]. Figure 3 sums up the activity in Project Inception.

Application Design

Application Design is the process of producing a logical and physical design of the KITS architecture. The primary input to this phase is the Information Flow Diagram produced in the Functional Specifications of the system. This abstract Information Flow Diagram is broken down into many *Steps* to fundamental levels of detail. The details of the elements of the *Final Step IFD* are documented in the Information Flow Components Dictionary (IFCD™). The logical view in the IFD is now mapped onto a physical view in the Systems Architecture Diagram (SAD™) using a technique called Systems Architecture Specifications. The details of the elements in the SAD are then documented in a Systems Architecture Components Dictionary (SACD™). The Application Design phase concludes with a prescription of testing strategies as shown in Figure 4.

Prototyping

Prototyping is the process of building a microcosmic version of the ultimate system. It begins with Knowledge Acquisition where the domain knowledge is acquired from the expert in the form of rules, heuristics, procedures and facts. The knowledge acquired from the expert is documented in a shell-independent formalism called the *Structured English Representation of Knowledge* [Prabhat et al, 1992]. The output document at the end of this sub-phase is the *Knowledge Document*. The Knowledge Base is then designed and specified. The prototype is implemented and validated for conformance to its functional specifications. The *Test Report* is generated at the end of this sub-phase. We have developed structured methodologies for each of these steps detailed in Figure 5.

Project Completion

Project Completion phase converts the prototype system to the full-scale KITS architecture. It begins with the necessary redesign of the prototype and scaling up the prototype Knowledge Base to contain the complete knowledge. The prototype is then ported to the delivery platform (if different from the development platform) and integrated with data sources and sinks such as databases, spreadsheets, etc. Validation Testing is conducted to ensure that the system conforms to the customer's requirements. The *User Manual* is produced in the Documentation phase. This concludes the deployment of the system. Change Management modules assist in the maintenance of the system thereafter. Figure 6 summarizes the activity in this phase.

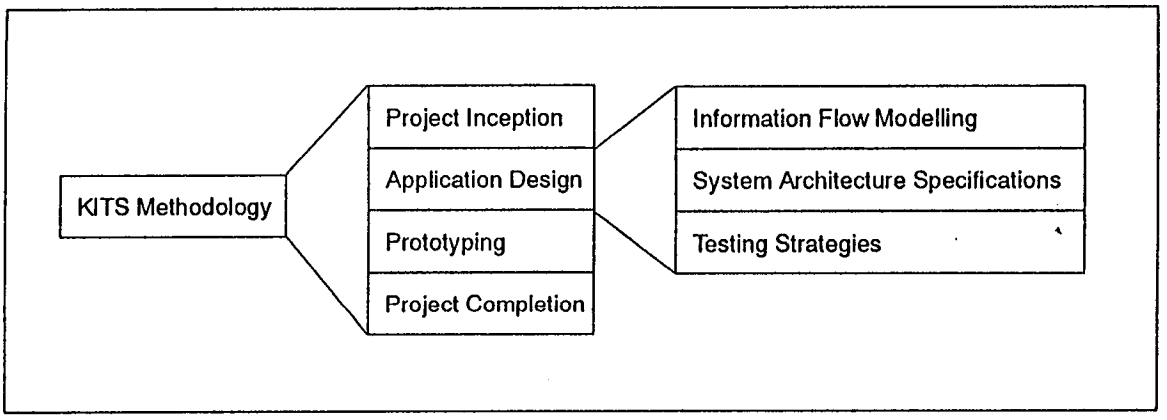

Figure 4. The KITS Methodology - Application Design.

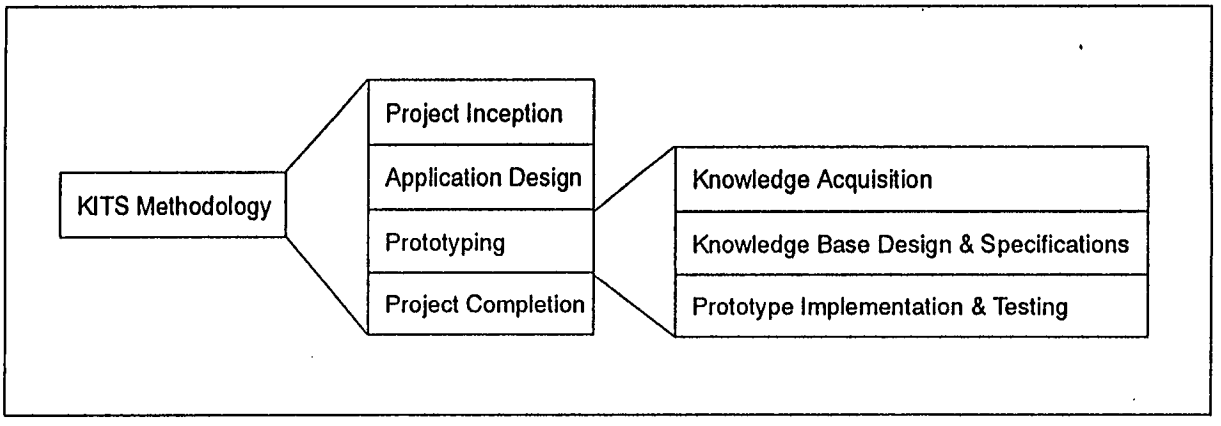

Figure 5. The KITS Methodology - Prototyping.

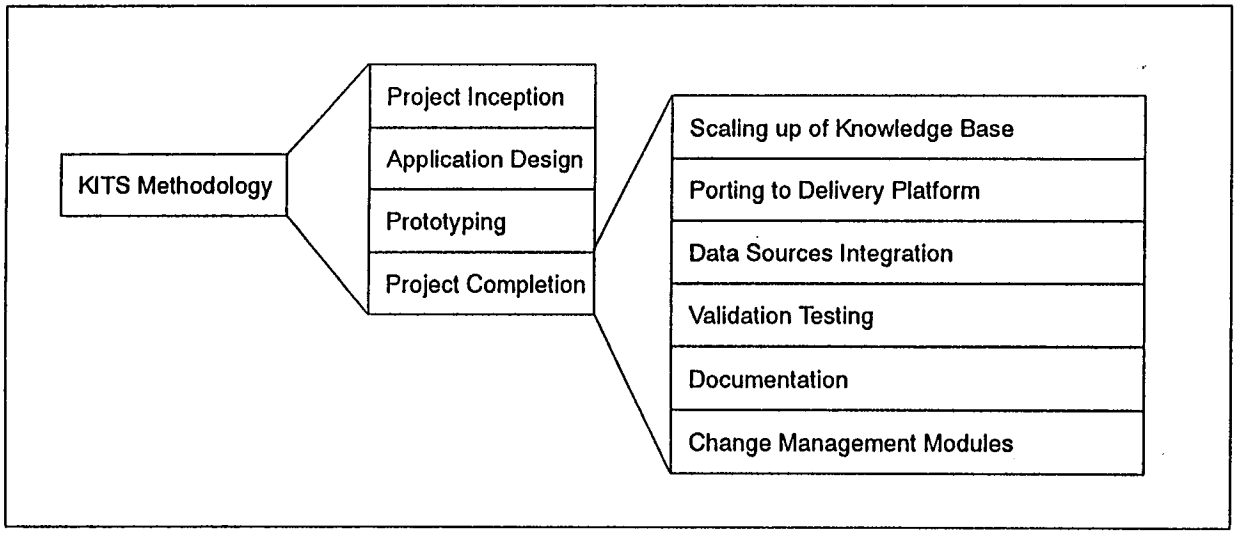

Figure 6. The KITS Methodology - Project Completion.

Verification and Validation - The Context

In the KITS Methodology, Verification and Validation are carried out during the phases of Prototyping and Project Completion. In order to ensure that all Knowledge-Based Processes in the KITS architecture conform to their functional requirements, they have to be specified and documented in an unambiguous and tool-independent formalism. In the KITS Methodology, the specification scheme employed is called *Knowledge-Based Process Specification Language (KPSL™)*. This language is the common denominator of the constructs and facilities found in most commercial Knowledge-Based System shells. However no commercial shell's representation language has been used.

Knowledge-Based Process Specification Language

At the end of Application Design and during the phase of Prototyping, the Knowledge Engineer encounters various Knowledge-Based Processes. A Knowledge-Based Process is one that requires reasoning and inferencing based on expert knowledge. These processes and their interrelationships are codified in a diagram called the *Information Flow Diagram (IFD)*. An IFD is a pictorial representation of the logical functionality of the KITS architecture.

As an example, let us consider a fictitious diagnostic Knowledge-Based System. The system is required to diagnose the cause of a problem in a machine and suggest some remedial action. The IFD for this system is shown in Figure 7. The Knowledge-Based Process "Start Diagnosis Activity" gets the "symptom" of the problem from the machine operator and performs some initialisation activities. It then passes the "symptom" to the Knowledge-Based Process "Find Cause of Problem" which locates the "cause" of the problem and passes it to the Knowledge-Based Process "Suggest Remedy for Problem". This process prescribes a "remedy" and passes it the Knowledge-Based Process "End Diagnosis Activity" which performs some winding up activity and returns the "remedy" to the operator. Each Knowledge-Based Process taps domain expertise from its associated Knowledge Repository.

All Knowledge-Based Processes in the IFD are now specified using KPSL constructs. A few general KPSL constructs are described below. Following that is the KPSL specification for the diagnostic Knowledge-Based System.

KPSL Constructs

The syntax used to describe the KPSL constructs is given below:

Symbol	Meaning
< >	replace with appropriate literal
[ ]	select from among alternate choices
|	separator for alternate choices

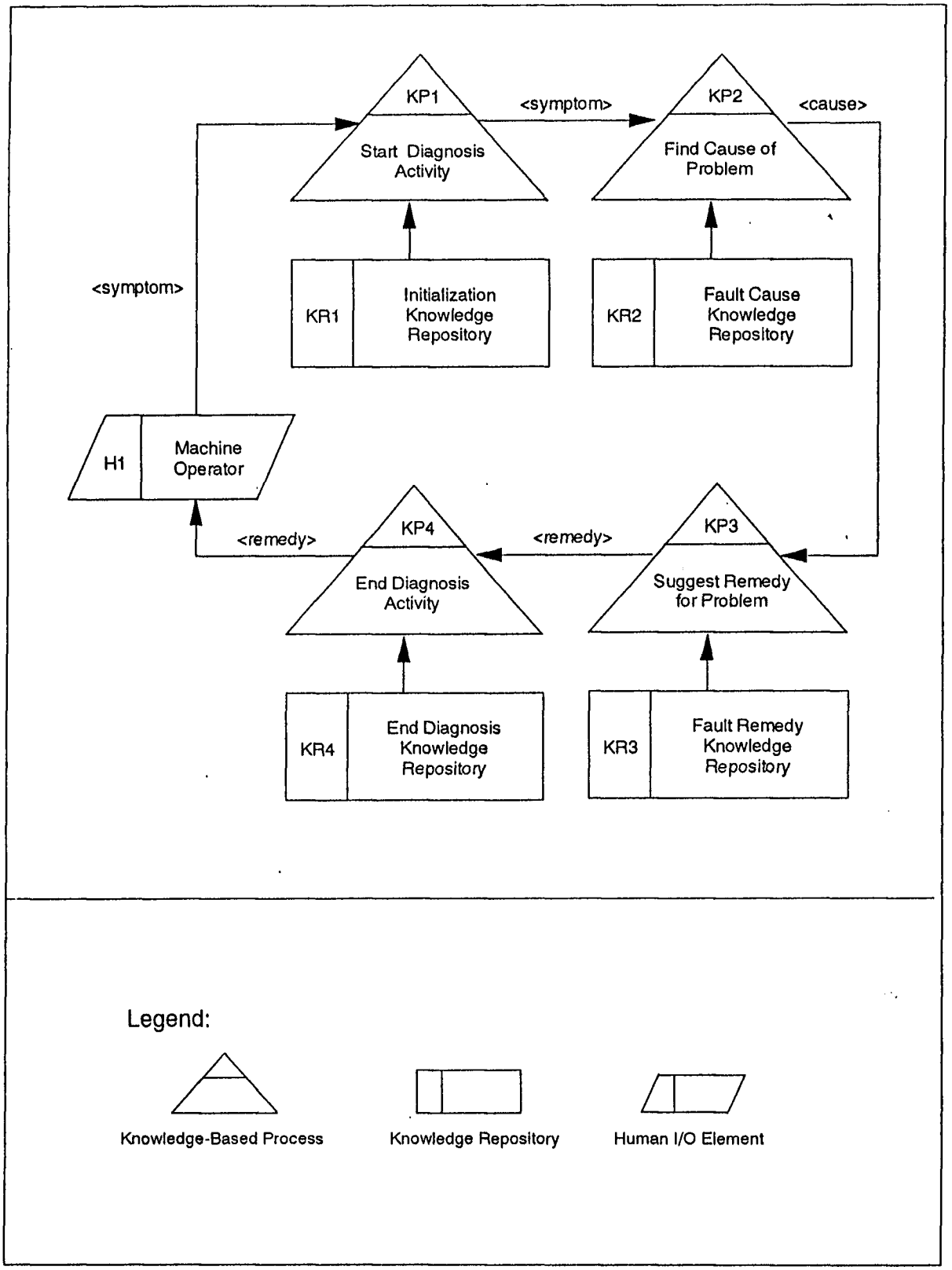

Figure 7. The Information Flow Diagram for the diagnostic Knowledge-Based System.

{ }	iteration

Operators

Arithmetic Operators	*	%	+	-	mod	div	
Relational Operators	<	>	=	<=	>=	<>	
Logical Operators	AND	OR	NOT				

Predicates

Predicates are used to test various conditions during the process of execution. The KPSL predicates are

- DEFINED(<Attribute>) - Returns TRUE if <Attribute> is defined. Otherwise, returns FALSE.
- NO_OF_VALUES(<Attribute>) - Returns the number of values that the attribute currently has. This is applicable for attributes that can take more than one value.

Problem-Solving Constructs

The expressive power of KPSL comes from the set of language statements that describe the problem-solving process. Some of them are explained below:

1. DETERMINE

Syntax:

DETERMINE <Attribute>
BY [FORWARD CHAINING | BACKWARD CHAINING ON <Attribute>]
USING <Knowledge Repository>

Description: Establish a value for the attribute by forward or backward chaining using the given Knowledge Repository.

2. ATTEMPT TO DETERMINE

Syntax:

ATTEMPT TO DETERMINE <Attribute> = { <Value> }
BY [FORWARD CHAINING | BACKWARD CHAINING ON <Attribute>]
USING <Knowledge Repository>

Description: By forward or backward chaining on the given Knowledge Repository, attempt to establish that the attribute takes the given value(s). In this case, establishing that the attribute takes the given value is of primary importance.

3. INITIALIZE

Syntax:

INITIALIZE BY [FORWARD CHAINING | BACKWARD CHAINING ON <Attribute>]
USING <Knowledge Repository>

Description: This is a startup mechanism to initialize the values of various attributes.

4. FIRE RULES

Syntax:

FIRE RULES BY [FORWARD CHAINING | BACKWARD-CHAINING ON <Attribute>]
USING <Knowledge Repository>

Description: Load the given knowledge repository and fire the rules in forward or backward chaining mode.

5. WINDUP

Syntax:

WINDUP BY [FORWARD CHAINING | BACKWARD CHAINING ON <Attribute>]
USING <Knowledge Repository>

Description: This is a shutdown mechanism to perform winding up operations.

6. SET CURRENT GOAL AS

Syntax:

SET CURRENT GOAL AS <Attribute>

Description: This is a means of reordering the sequence in which goals will be tried.

KPSL Specification for the Diagnostic Knowledge-Based System

Process Name Start Diagnosis Activity **Process Specification** INITIALIZE BY FORWARD CHAINING ON Initialization Knowledge Repository
Process Name Locate Cause of Problem ***Process Specification*** ATTEMPT TO DETERMINE CauseOfProblem = "Insufficient lubrication" BY BACKWARD CHAINING ON CauseOfProblem USING Fault Cause Knowledge Repository IF NOT DEFINED(CauseOfProblem) THEN ATTEMPT TO DETERMINE CauseOfProblem = "Inadequate power supply" BY BACKWARD CHAINING ON CauseOfProblem USING Fault Cause Knowledge Repository ENDIF IF NOT DEFINED(CauseOfProblem) THEN DETERMINE CauseOfProblem BY BACKWARD CHAINING ON CauseOfProblem USING Fault Cause Knowledge Repository ENDIF

Process Name Suggest Remedy for Problem **Process Specification** DETERMINE RemedyOfProblem BY FORWARD CHAINING ON Fault Remedy Knowledge Repository
Process Name End Diagnosis Activity **Process Specification** WINDUP BY FORWARD CHAINING ON End Diagnosis Knowledge Repository

After these Knowledge-Based Processes have been implemented using a Knowledge-Based shell or a language, they can be verified and validated by ensuring that they conform to their KPSL specifications.

Knowledge-Based Process Specification Language - Advantages

The advantages of KPSL are many:

- The specification results in effective Verification and Validation of Knowledge-Based components.
- It is a structured language; therefore ambiguities in specifying Knowledge-Based Processes are significantly reduced resulting in more accurate validation.
- The specification is English-like; therefore comprehensibility of processes is enhanced.
- KPSL is shell-independent. A set of Knowledge-Based Processes specified using KPSL can be implemented in more than one Knowledge-Based tool. This helps in easy portability between shells. The KPSL specification forms the common ground from which all physical implementations arise. This implies that even if the tool is changed, the specification remains unaffected.

References

1. [Prabhat et al, 1991] - Prabhat G.B., Ramachandran N., Nagendra Prasad G., "Knowledge Flow Modelling and Beyond: Towards a Development Methodology for Knowledge-Based Systems", Proceedings of the AAAI-91 Workshop on Standards in Expert System, July 1991, pp 11-34.

2. [Prabhat et al, 1992] - Prabhat G.B., Rajmohan S., Anandhi I., Srinivasan P., "Structured English Representation of Knowledge - A Mediating Knowledge Representation Formalism", Proceedings of the AAAI-92 Workshop on Knowledge Representation Aspects of Knowledge Acquisition, July 1992, pp 129-146.

Trademarks