Modeling Frameworks for Knowledge Engineering Approaches

Daniel Ashlock

[1] Department of Visual Arts and Curatorial Studies, University in Milan, 20143 Milano MI, Italy.
[1]ashlockmilan@hotmail.com

Article Info
Journal of Computing and Natural Science (http://anapub.co.ke/journals/jcns/jcns.html)
Doi: https://doi.org/10.53759/181X/JCNS202202003
Received 18 May 2021; Revised form 25 June 2021; Accepted 30 September 2021.
Available online 05 January 2022.

Abstract - Human knowledge was regarded as a transfer process into an applied knowledge base in the early 1980s as the creation of a Knowledge-Based Systems (KBS). The premise behind this transfer was that the KBS-required information already existed and only needed to be gathered and applied. Most of the time, the necessary information was gleaned through talking to professionals about how they handle particular problems. This knowledge was usually put to use in production rules, which were then carried out by a rule interpreter linked to them. Here, we demonstrate a number of new ideas and approaches that have emerged during the last few years. This paper presents MIKE, PROTÉGÉ-II, and Common KADS as three different modeling frameworks that may be used together or separately.

Keywords - Knowledge-Based Systems (KBS), Knowledge Engineering (KE), Artificial Inttelligence (AI).

I. INTRODUCTION

Expert systems are created using Knowledge Engineering (KE), a method that allows them to help with problems in their specified area of knowledge. Expert systems combine a vast, extendable knowledge and understanding with a middleware that dictates how the expertise base's content should be applied to each circumstance. Machine Learning (ML) may be used in the networks so they may learn through experience in same way human beings do. Classification methods are utilized in a variety of sectors, including health coverage, customer support, finance, engineering, and the system of justice. Knowledge engineering attempts to tackle challenges and difficulties in the same way that a human expert would. It does so by using algorithms to mimic the thought processes of a subject matter expert. Knowledge engineering examines how a task or choice is attained by studying the properties of the task or resolution. To tackle the issue or topic, a library of problem-solving methodologies and a body of ancillary information are used. The amount of ancillary knowledge available might be enormous. The virtual specialist may aid with diagnosing, fixing difficulties, supporting a person, or functioning as a digital assistant, based on the tasks and the expertise that is called on.

Artificial Intelligence (AI) research used to be primarily concerned with developing formalizations, inference processes, and operationalization tools for Knowledge-Based Systems (KBS). To test the viability of alternative methods, development activities were often limited to the creation of smaller KBSs. The findings of these experiments were encouraging, but commercializing this technology to create big KBSs was a huge failure in many situations [1]. To put it another way, it was very much like the "software crisis" that occurred in the late 1960s when software development tools were not enough for huge, long-lived commercial systems like mainframe computers. The poor condition in the construction of KBSs made obvious the necessity for new methodical methods, similar to how the software crises led to the formation of the field of Software Engineering.

As a result, the new field of Knowledge Engineering (KE) has a similar objective to that of Computer Engineering: transforming the method of developing KBSs from an art form into a scientific one. For this to work, the installation and renovation process itself must be analyzed, and techniques, protocols, and technologies specialised in KBS growth must be created. After that, we will go through some key KE historical milestones, with a concentration on the paradigm change from conductive medium to modelling framework. Modeling models established in the last few years will be discussed, including MIKE, PROT-G II, and CommonKADS. The next section is a review of the available KE literature. This paper has been organized as follows: Section II evaluates the past literature. Section III presents a critical analysis section; while Section IV concludes the paper.

II. RELATED WORKS

A. Meena in [2] conducted a thorough examination of the numerous rule domain knowledge systems, on the other hand, revealed that the fairly basic representational framework of frequent patterns could not enable an acceptable representation of various forms of information. Strategic information regarding the sequence, in which objectives should be attained, for example, may be found in the MYCIN knowledge base. One example is the combination of domain-specific information about the causes of a certain disease with common illness causation. Because of the variety of information kinds and the

absence of appropriate explanation for the many rules, maintaining such knowledge bases is complex and time-consuming. As a result, this transfer technique only worked for tiny prototype systems and did not create substantial, dependable, and maintained knowledge bases.

J. WANG in [3] argue that because tacit knowledge plays such a significant part in an expert's problem-solving capacities, it was realized that the presumption of the transfer approach, that explicit knowledge is the gathering of previously existing knowledge pieces, was incorrect. The transfer technique was abandoned in favor of modeling because of these flaws. A. Habermann in [4] presented an analysis of Knowledge Engineering (KE) as a modeling procedure. Currently, the general opinion is that the KBS construction process may be considered a modeling operation.

R. Hicks in [5] argue that creating a KBS entails creating a computational model whose problem-solving abilities are on par with those of a domain expert. Instead of creating a cognitively sufficient paradigm, the goal is to build a model that gives equivalent outcomes in problem-solving for issues in the focus point. The practitioner may be cognizant of portions of his or her understanding, but he or she will not be cognizant of a substantial part of it since it is buried in his or her abilities. This information is not immediately available; instead, it must be gathered and organized throughout the knowledge acquisition process. In this way, the process of acquiring information has shifted from being viewed as a means of transferring knowledge to one of creating a model.

P. Moore and H. Pham in [6] reviewed the specific strategies in KE. Generic Tasks and Role-Limiting Methods were two major methods that emerged in the 1980s and had a considerable effect on the growth of modeling approaches in KE. One of the first initiatives to promote the creation of KBSs by utilizing the concept of a repeatable problem-solving technique was Role-Limiting Methods (RLM). The RLM method may be described as a shell method. Because such a shell includes an execution of a certain PSM, it can only be used to tackle tasks that the PSM is designed to handle. PSM also identifies the many functions that recognition may play in the problem-solving procedure. It also defines the domain ontology for the responsibilities entirely, requiring the professional to only reproduce particular generic ideas and connections described by these responsibilities e.g. PSM Heuristic Segmentation.

The professional is given role explanatory variables through an RLM predicated on Heuristic Segmentation. Using this role, the expert must first identify which domain-specific conception belongs to this role, such as "patient records," and then give domain examples of this conception, such as concrete personal health information. It is critical to realize that the RLM only uses a specific type of information. Domain-specific implementations may be purchased via (graphical) APIs that are specifically suited to a PSM. This paper will present an analysis of the modeling frameworks that address several aspects of model-centered KE methods.

III. CRITICAL ANALYSIS

Modeling Frameworks

In this part, we will go through three different modeling contexts for model-based KE methods. It is important to note that CommonKADS is notable for defining how the Expertise Prototype should be structured. Meanwhile, MIKE focuses on how the Expertise Model should be formalized and operational as a function of the knowledge acquisition process. It should be obvious that there are other techniques that are well recognized in the KE society, such as VITAL, Commet, and EXPECT. The scope of this article does not allow for an in-depth examination of all of these techniques.

CommonKADS Method

CommonKADS and KADS are well-known knowledge engineering approaches. A fundamental feature of KADS is the creation of a set of models, where each model represents certain elements of the KBS to be created as well as its surroundings [7]. When it comes to CommonKADS, it is important to understand that there are a number of different models to choose from. Instead than focusing on modeling the KBS's operating environment and the activities done inside it, the first four models set out to depict (non-functional) elements of the KBS as it is currently being developed. After that, we will take a quick look at each of these frameworks before going into detail about the Expertise Framework:

- The administrative features and composition executed by each organisational structure are specified in the Operational Framework. Additionally, the weaknesses of present business operations are highlighted, as well as possibilities to enhance these processes through the introduction of KBSs.
- Using the Task Prototype, the KBS may be implemented in a hierarchical structure that describes all of the tasks that must be completed inside the company. This contains details of which agents are allocated to various assignments.
- The Agent Model explains the attributes of every agent participating in the task execution. In essence, an operator can be a person or system software, such as a KBS.
- The Communications Model specifies the numerous relationships between the various actors. It determines, among other things, what type of data is shared between the entities and which component initiates the contact.

The KADS method makes a significant addition by proposing a structure for the Expertise Framework, which separates three categories of knowledge necessary to accomplish a specific job. In general, the three kinds correlate to a fixed state, a functionality perspective, and a dynamical perspective of the KBS to be constructed (see Figure 4 for "application domain," "inferential layer," and "project layer," correspondingly).

The domain layer - At the application domain, all domain-specific data required to complete the job is represented. This comprises ontology for conceptualizing the domain and a declaratory theory for the needed domain knowledge. One goal of the domain layer's structure is to make it as adaptable as feasible for various purposes.

The inference layer - The KBS's logic procedure is explained at the inference level by making use of the PSM concept. The inference level specifies the general PSM's inference operations as well as the domain knowledge's responsibilities inside the PSM. Inference structures specify the relationships between inference operations and responsibilities. Additionally, the concept of roles offers a domain-agnostic perspective on domain expertise. The inference framework for the PSM Probabilistic reasoning Segmentation is depicted in Figure 1 (middle section). We can observe, for example, that "patient records" serves as "measurements" within the Heuristic Classifier inference framework.

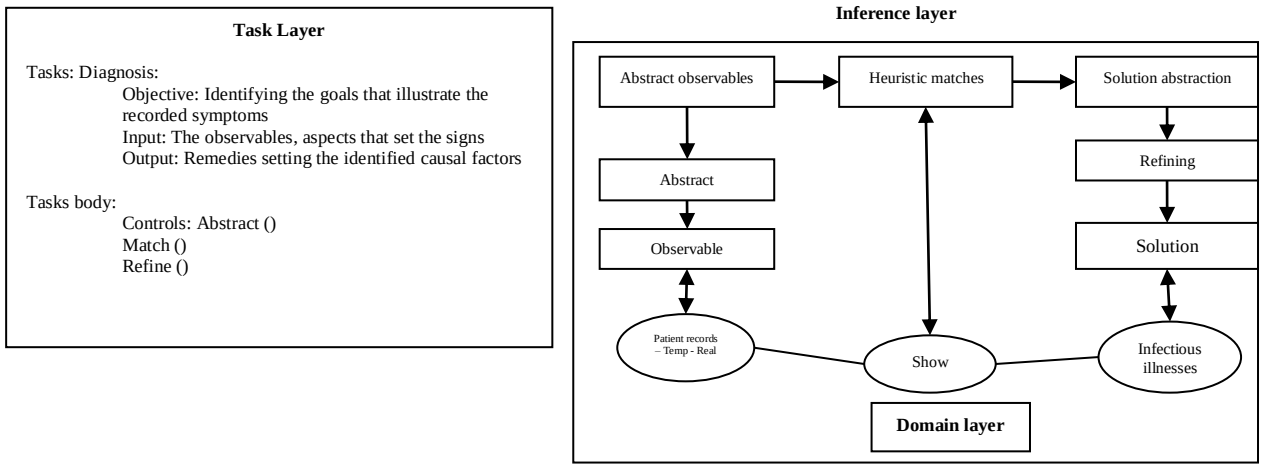

Fig 1. Expertise framework for clinical diagnosis

The task layer – The task layer decomposes activities into subprojects and reasoning operations, as well as a goal definition for each assignment and a requirement about how these objectives are met. The task level also allows you to set controls over the subtasks and reasoning operations established at the inference layer.

To define an Expertise Framework, languages are provided: CML (Conceptual Modeling Language), a quasi-language with path diagram, and (ML) 2, a structured programming language based on the first degree predicate logic, meta-logic, and dynamic logic. Whilst CML is focused on establishing a foundation for communications between the inference engine and the analyst, (ML) 2 is focused on formulating the Expertise Framework. At the inference and task layers, the complete distinction of domain-specific expertise from the basic explanation of the PSM facilitates two types of reusability: on the one side, a domain strand definition can be repurposed for fixing multiple activities by distinct PSMs; on the other side, a specified PSM can be repurposed in a particular level by establishing a clear perspective on another application domain. The strong interaction issue concept, which was handled in the GT method, has been weakened by this reusable strategy.

The term "relative interaction concept" refers to a relationship between the architecture of domain expertise and the sort of problem that has to be addressed. The concept of layered ontology is offered to enable a dynamic adaption of the application domain to a new task surrounding: Task and PSM topologies can be thought of as perspectives on a larger ontology. Within CommonKADS, a repository of interchangeable and customizable elements has been created that may be utilized to construct an expert model. In general, the Expertise Framework and the Communication Network encapsulate the target system's performance requirements.

The Design Model was created based on these criteria, and it describes the architecture of the system and numerical simulations for carrying out the reasoning operations, among other things. The goal of KADS is to achieve a structure-preserving design, which means that the architecture of the Prototype Design should as closely as possible match the architecture of the Expertise Model. All of the implementation phases, which result in the sequential building of the many models, are encapsulated in a cyclic and risk-driven process model, comparable to Boehm's agile methodology. The expert model's basic component is comparable to the informational, operational, and controller views of a systems that are well-known in computer programming.

Nevertheless, there is a significant distinction between an inference level and a standard data-flow graph: A data-flow chart is entirely stated in domain-specific languages, but an inference level is characterized in general terms and offers a flexible link to the data provided at the application domain through responsibilities and domains' perspectives. Furthermore, the data model does not correlate to the application domain, because the domain layer could provide a model consisting of the domain in question that is only partly used against the inferential stack, while the data model characterizes precisely the data used to clarify the flow of data within the flowchart.

MIKE Method

The MIKE methodology (Prototype and Iterative KE) is a strategy for developing KBSs that covers all phases from derivation to definition, engineering, and execution. MIKE recommends incorporating experimentation and semiformal and formal specifications methodologies into an engineering framework. The primary difference between CommonKADS and MIKE is that CommonKADS integrates prototyping and promote for an iterative and reversal system design into a prototype paradigm.

- MIKE adopts CommonKADS' Expertise Model as its conceptual model design, allowing for a seamless experience from a semiformal model, the Structural Modelling, to a structured representation, the KARL Prototype, and finally to an integration representation, the Design Model. The Expertise Model's gradual shift between multiple representation layers is critical for permitting gradual and reversal system evolution in operation.
- Because the KARL Model is executable in MIKE, prototyping may be used to validate the Expertise Model. This greatly improves the expert's inclusion into the development cycle. Figure 1 depicts the various MIKE development efforts as well as the papers that emerge from these operations. Elicitation, Translation, Operationalization/Formalization, Designing, and Deployment are the subactivities that make up the whole design process in MIKE. Each activity focuses on a distinct element of the system's evolution.
- Elicitation is the first step in acquiring knowledge. Structured questionnaires, for example, are used to gather unstructured representations of knowledge about a given area as well as the problem-solving procedure itself. Knowledge standards are created to store the resultant knowledge stated in natural language. The knowledge structures that may be recognized in the learning procedures are portrayed in a semi-formal variation of the Expertise Model during the interpretation stage: the Structural Models.

All structuring data in this system is illustrated in a predetermined, limited language, such as dependency relationships between two hypotheses, but the fundamental building elements, such as the formulation of an assertion, are expressed by unconstrained languages. This representation gives a first organized description of the evolving cognitive structure and may be utilized as a basis for communications between the learning algorithm and the professional. As a consequence, the expert may be included in the knowledge-organization process. The Modeling Approach serves as the framework for the Operationalization/Formalization procedure, which culminates in the KARL Model, a formalized Expertise Concept.

The KARL Framework has same analytical model as the Structure Model, but the essential building components, which were previously described as plain language texts, are now written in KARL, a structured programming language [8]. These representations remove the uncertainty and complexity of human language representations, making it easier to grasp the full problem-solving procedure. Due to the fact that “KARL” is a functional language (with some restrictions) the model can be translated to an operative representation. The KARL Prototype, which is the output of the information acquisition process, encapsulates all required functionality for the ultimate KBS.

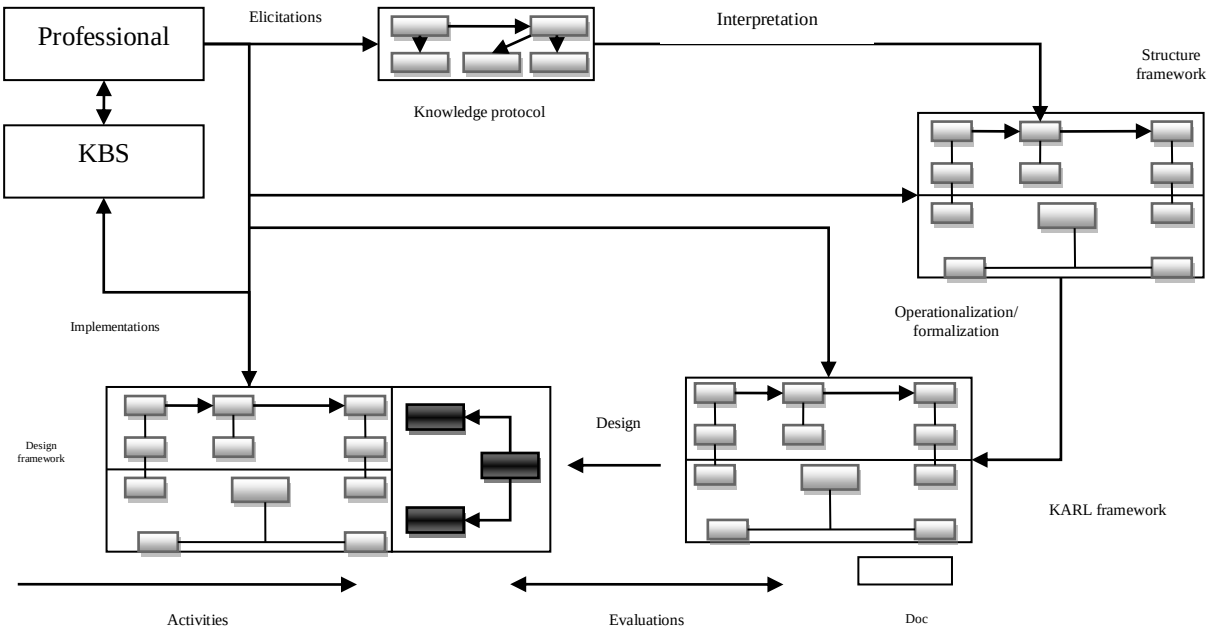

Fig 2. Documents and steps in MIKE development procedure

Additional non-functional needs are taken into account in the Development process. Effectiveness and manageability, for example, are non-functional prerequisites, as are the restrictions placed by reference hardware and software ecosystems. Efficiency is handled in part during the knowledge acquisition process, but only to the degree that it

influences the PSM. As a result, functional structure is already a component of the learning process. As a result, the MIKE conceptual design is the software development counterpart of conceptual design and component design. This phase's outcome is the Design Model that is written in the DesignKARL language. DesignKARL is a KARL extension that adds more primitives for organizing the KARL Model and specifying data mining algorithms kinds. You may also use DesignKARL to explain the conceptual design and how development decisions intersect. The Design Model captures all of the KBS's functional and non-functional needs. The conceptual model is translated into the target hardware / software environment during the project execution. The end outcome of all steps is a collection of interconnected Expertise Model refinement states. Explicit connections connect the knowledge in the Structure Model to the appropriate knowledge in the knowledge procedures. Conceptual frameworks and reasoning operations are connected to protocol nodes, which are expressed in plain language.

The Design Model improves the KARL Concept by converting inferences to programs and adding new data model. These elements of the Design Model are connected to the KARL Model's relevant inference, and therefore to the information procedures. Integrated with objective of maintaining the architecture of the Expert Framework during designing, the connections between the various framework variants and the last implementation ensure the tracking of non-functional requirements. The complete development procedure, i.e., the format of acquiring knowledge, designing and implementation is done in cycles determined by the spiral framework as a process framework.

Each cycle is capable of producing prototypes of KBS that can be assessed by evaluating it in the actual targeted ecosystem. The findings of the assessment are utilized in the upcoming cycle to rectify, change and extend the prototype. The process of development of MIKE inherently incorporates various forms of prototyping. Language KARL executability permits the exploration prototyping. Therefore, the various phases of knowledge acquisition are done until the projected functionality has been attained. In the same manner, the experimental prototyping can be utilized in the design stage for assessing the Design framework since the DesignKARL is mapped within the version that is executable. The design framework is refined using the iterative sub-stages of the design stage until all the inclusive functional requirements have been reached.

Over the past few decades, a novel version of the specifications' language KARL has been structured that incorporates the ideology of tasks and therefore provides a manner of formally specifying tasks-to-method decomposition structures. The MIKE methodology as illustrated above is constrained to KBE modelling only the business ecosystem, the MIKE strategy is presently extended by novel frameworks that illustrate various perspectives on the firm. Major focus is placed on the smooth change from enterprise modelling to the problem-solving modelling procedures.

The PROTÉGÉ-II Method

The PROTÉGÉ-II Method focuses on providing support to the KBS develop by reapplying ontologies and PSMs [9]. Moreover, the method focuses more on the generation of the customized knowledge-acquisition instruments from the ontologies. The method also depends on the tasks-method-decomposition architecture. By depending on PSM, tasks can be decomposed into subtasks. The form of decomposition is refined down towards the level whereby the primitive approaches, known as methodologies, are present for handling subtasks. Presently, the method provides a smaller library of assessed PSMs that have been utilized to handle different varieties of tasks. With the methodology, the output and input of an approach is specified using the methodological ontology (see Fig 3); the methodological ontology defines the relationships and concepts, which are utilized by PSM for issuing its functionalities, such as the Board-Game approach utilizes among others the perspective of "moves", "locations" and "pieces" to issue its functionalities, which is to shift pieces between different locations on board. In this manner, approach ontology corresponds to the overall terminology at presented by the gathering of knowledge obligations of PSM.

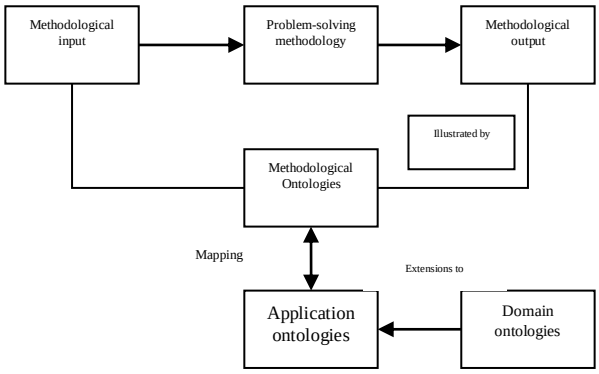

Fig 3. PROTÉGÉ-II ontologies

A second form of ontology utilized within the methodology is the domain ontology. These ontologies define a shared conceptualization of domains. Both the domain ontologies and PSMs are reusable elements for developing KBS. Nonetheless, as a result of the interaction issues, the interdependencies between PSMs and domain ontologies with their interlinked methodological ontologies have to be considered when building KBS from the reusable elements. Thus, the

methodology suggests a perspective of application ontology to potentially extend the domain ontologies with PSM-specific relationships and concepts. To associate the applications' ontology with the methodological ontology, this method provides various forms of mapping relationships.

- Renaming mappings are utilized for translating domain-specific terminologies into the approach-specific terminologies.
- Filtering mappings issue a method of choosing subsets of domain cases as the cases of the corresponding approach concepts.
- Group mappings issue a function to compute cases of methodological concepts from the application concepts and definition instead of the application cases.

On the contrary, mappings are same as the schema translation protocols as illustrated in the segment of interoperable database schemes, and, correspond to the lift operators in ML2 or in the KARL's perspective definition. The method focuses on providing a smaller collection of simplistic mapping to limit the re-usage efforts required to specify the mappings. Therefore, the methodology suggests the reuse of domain knowledge on instances where the essential mappings are maintained lower.

An element of the methodology is that it can potentially generate knowledge-acquisition instruments from the application or domain ontologies [10]. Thus, the methodological system integrates the DASH system that takes ontologies as inputs and produce output knowledge acquisition instruments, which permit domain specialists to compute cases on domain concept. Therefore, the domain facts can be considered by the domain professionals themselves. Over the past few decades, the PROTÉGÉ-II Method has been adjusted to the CORBA-centred ontologies and PSMs that enable the re-application of these elements on the internet ecosystem.

IV. CONCLUSION

Research efforts in the field of Knowledge Engineering (KE) have amounted to significant results that are fundamental in fields such as knowledge management, software engineering and information integration. These results and achievements include (a) the definition of the model structure within the model of the model-centred KE that significantly separates the various forms of knowledge that are fundamental in notion of Knowledge-Based System (KBS). The Expert framework is the most vital form of these frameworks; (b) the separation of the perspectives of the task, issue-solving approach and the domain-knowledge issues a promising foundation for establishing reuse-based KBS development; (c) incorporation of a firm conceptual framework is a distinctive element of the formal specification language within KE.

The establishment of an Organizational Memory as a mechanism of knowledge preservation, dissemination, and reuse is a critical technological component of knowledge management. A blend of informal, semi-formal, and formal information will often be found in an Organizational Memory. Novel techniques for accessing and disseminating these knowledge pieces will be necessary when enterprise-wide organizational memory grow into extremely vast collections of quite varied knowledge items. Ontologies can be used to define ideas that are utilized to organize and structure knowledge items in organizational memory in this context. Moreover, such schemas can be used to assist users in locating relevant information, for example, by providing relevant constructs for posing questions. However, despite the fact that much effort is put into organizational learning, the development and utilization of overall organisational memories is still in its early stages.

References

[1]. N. Johnson, "Knowledge based systems series Vol. 1: Knowledge acquisition for knowledge based systems Vol. 2: Knowledge acquisition tools for expert systems", *Knowledge-Based Systems*, vol. 3, no. 3, p. 181, 1990. Doi: 10.1016/0950-7051(91)90030-6.

[2]. A. Meena, "Study and Analysis of MYCIN expert system", *International Journal Of Engineering And Computer Science*, 2016. Doi: 10.18535/ijecs/v4i10.41.

[3]. J. WANG, "Knowledge, Routines and Performance in Collective Problem Solving", *Acta Psychologica Sinica*, vol. 40, no. 8, pp. 862-872, 2008. Doi: 10.3724/sp.j.1041.2008.00862.

[4]. A. Habermann, "Engineering large knowledge-based systems", *Data & Knowledge Engineering*, vol. 5, no. 2, pp. 105-117, 1990. Doi: 10.1016/0169-023x(90)90007-z.

[5]. R. Hicks, "Knowledge base management systems-tools for creating verified intelligent systems", *Knowledge-Based Systems*, vol. 16, no. 3, pp. 165-171, 2003. Doi: 10.1016/s0950-7051(02)00082-5.

[6]. P. Moore and H. Pham, "Personalization and rule strategies in data-intensive intelligent context-aware systems", *The Knowledge Engineering Review*, vol. 30, no. 2, pp. 140-156, 2015. Doi: 10.1017/s0269888914000265.

[7]. J. Kingston, "Designing knowledge based systems: the CommonKADS design model", *Knowledge-Based Systems*, vol. 11, no. 5-6, pp. 311-319, 1998. Doi: 10.1016/s0950-7051(98)00071-9.

[8]. K. Schoonover, "Wastrels of Time: Slow Cinema's Laboring Body, the Political Spectator, and the Queer", *Framework: The Journal of Cinema and Media*, vol. 53, no. 1, pp. 65-78, 2012. Doi: 10.1353/frm.2012.0007.

[9]. J. Gennari, R. Altman and M. Musen, "Reuse with PROTÉGÉ-II", *ACM SIGSOFT Software Engineering Notes*, vol. 20, no., pp. 72-80, 1995. Doi: 10.1145/223427.316710.

[10]. J. Runkel and W. Birmingham, "Knowledge acquisition in the small: building knowledge-acquisition tools from pieces", *Knowledge Acquisition*, vol. 5, no. 2, pp. 221-243, 1993. Doi: 10.1006/knac.1993.1009.