**NAME ETABO KELVIN ESEKON**

**REG NUMBER:SC212/0743/2015**

**COURSE:BSC SOFTWARE ENGINEERING.**

**KBS ASSIGNMENT.**

**EXPERT SYSTEM DEVELOPMENT PROCESSES**

An expert system is a knowledge based system that has the knowledge from the domain expert or the human expert in form of program. An expert system is typically and refined over a period of several years.in practise it may not be possible to break down the expert system development cycle precisely. however, an examination of these five stages may serve to provide us with some insight into the ways in which expert ae developed. There are five phases of expert system development processes described below.

1. **Identification of the problem the system is going to solve;**

Before you can begin to develop an expert system, it is very important that you described, with as much precision as possible, the problem that the system is intended to solve. It is not enough simply to feel that the system would be helpful in certain situation., you must determine the exact nature of the problem and state the precise goals that indicate exactly how we expect expert system to contribute to the solution.

1. **Conceptualization**

Once you have formally identified the problem that an expert system is to solve, the next stage involves analysing the problem further to ensure that its specifics as well as it generalities are understood. In the conceptualization stage the knowledge engineer frequently creates a diagram of the problem domain. it is often helpful at the stage to divide the problem into a series of sub-problems and to diagram both the relationship among the pieces of each sub-problem and the relationship among the various sub-problems.

Detail Design as with other methodologies, conceptual design should proceed relatively independently of implementation details. These details are addressed during detail design. Knowledge engineers: identify propositions for logic, write descriptions and pseudo-code for procedures, draw network diagrams for semantic networks, write English language rules for production rules, draw diagrams or build models for direct representations, identify and name slots for frames and scripts and identify and name table entries for data tables.

Detail design addresses syntax and language specific constraints, but the temptation to start coding should be resisted. during detail design, the knowledge engineers converts the and/or tree into English language rules. These rules provided a detailed coding specification.

1. **Formalization**

In the preceding stages, no effort has been made to relate the domain problem to the artificial intelligence technology that may solve it. During the identification and the conceptualization stages, the focus is entirely on understanding the problem. Now, during the formalization stage, the problem is connected to its proposed solution, an expert system by analysing the relationships depicted in the conceptualization stage.

During formalization, it is important that the knowledge engineer be familiar with the following

1. The various techniques of the knowledge representation and heuristic search used in expert systems.
2. The expert system “tools” that can greatly expedite the development process.
3. Other expert systems that may solve similar problems and the thus may be adequate to the problem at hand.
4. **Implementation:**

During implementation stage, the formalized concepts are programmed onto the computer that has been chosen for system development, using the predetermined techniques and tools to implement a “first pass” prototype of the expert system.

Theoretically, if the methods of the previous stages have been followed with diligence and care, the implementation of the prototype should be as much an art as it’s a science, because following all rules does not guarantee that the system will work the first it is implemented. Many scientists actually consider the first prototype to be a ‘throw-away’ system, useful for evaluating process but hardly a usable expert system.

1. **Testing**

Testing provides opportunities to identify the weakness in the structure and implementation of the system and to make the appropriate corrections. Performance verification for expert systems is critical as the risk inherent in a system that delivers incorrect advice can be enormous. it has been recommended that multiples feature of an expert, including its recommendation, the reasoning modelled in it. The user interface and the explanation facilities be subject to testing and validation. Care should be taken to make sure that a sufficient wide range of test cases are generated in order to achieve domain coverage. Production rules are theoretically supposed to represent modular, independent chunks of knowledge. however, rules are never completely independent of each other and may interact in unexpected ways. The logic encapsulated in the inference rules was validated thoroughly so as to avoid any unanticipated consequences of rules interaction.

**Code The KBSDLC elaborates the** “implementation” phase by defining the “code,” “test reasoning,” “test knowledge” processes. Coding is the recording of events - knowledge in this case-by using some language. Coding translates the detail design English language rules, for example-into the language of the knowledge engineering tool. Coding includes entering the knowledge representation into the tool’s knowledge base. Although not identical to coding in a low-level language, translation into Lisp, Prolog, or an expert system shell language has much in com- mon with other types of program coding. During coding the knowledge engineer frequently discovers problems in the detail design. Therefore, detail design and coding often run concurrently or else cycle in a tight loop. In addition, problems found during coding may cause extensive process reactivations. This happened in Med claim. In phase 2, when the knowledge engineers were integrating the code for the eligibility and appropriateness sub-problems, difficulties with integration resulted in completely redesigning the eligibility knowledge representation. The original eligibility prototype was entirely rule based. After integration, the eligibility sub-problem was partitioned between rules and the database.

**Test Reasoning**, the first of three KBSDLC testing processes, searches for invalid reasoning. This process tests the “mechanical” details such as interfaces and internal flow. The knowledge engineers construct artificial test cases to test specific parts of the system. Coding is reactivated to correct errors of invalid reasoning. Invalid reasoning occurs because the programmer incorrectly translates the knowledge. It reveals itself when the system violates the programmer’s expectations. During the Medclaim project, this process was not adequately isolated. The expert would diagnose a problem only to find it was caused by a coding error. The expert’s time would have been better spent when she worked with the system, if the mechanics of the knowledge representation had been tested more thoroughly.

**Test Knowledge** Especially in knowledge-based systems, correct code does not mean correct knowledge. The second testing process attempts to detect invalid and ambiguous knowledge. In addition to artificial cases, the expert and knowledge engineer use real cases to test that the system makes decisions appropriately. They save the test cases for later use.

**CONCLUSION**

The structure provided by system development life cycles sometimes makes system development easier. When developing transaction processing systems, many organizations simplify the structure further by executing the SDLC in a rigidly sequential manner. They also implement de- tailed checklists, attempting to squeeze the last bit of uncertainty from the development process. These tactics have had only limited success. As systems become more complex, however, simple methodologies become even less appropriate. As with most decision support systems, attempts to build knowledge-based systems without reviewing and re- vising will almost surely fail. At the same time, the need for a complex methodology does not negate the need to accomplish many tasks needed to build simpler information systems. The KBSDLC incorporates these tasks. It elaborates on previous methodologies, both sequential and evolutionary, to accommodate the specific idiosyncrasies of building knowledge-based systems. Building knowledge- based systems is ill-suited to a sequential progression from problem definition, to design, implementation, testing.

**REFERENCE**

1; Enger N L classical and structured system life cycle phases and documentation. (1982) pp-24

2: Toeliner, J D project management of systems projects in cotterman (1982 pp91-98

3: Ackoff R L management misformation system manage (1967) pp B-147-B156

4: berry D C the problem of implicit knowledge expert system vol 4 no 3 (1987) pp14-150