Journal of Theoretical and Applied Information Technology





www.jatit.org

# RULE BASED EXPERT SYSTEM FOR SELECTING SOFTWARE DEVELOPMENT METHODOLOGY

**M. AYMAN AL AHMAR**

Asstt. Prof. and Deputy Dean, College of Engineering and Information Technology, Fujairah Campus,

Ajman University of Science and Technology, UAE

**ABSTRACT**

Software development methodology is a formalized approach that is used to plan and manage the process of developing a software system. Since there are many software development methodologies, one of the challenges faced by software developers is to decide which methodology to apply in a software project. This paper presents the modeling and development of a prototype expert system that helps software project managers and software engineers in selecting the appropriate software development methodology. The developed system is successfully designed as rule based expert system supported with object oriented modeling. The user interaction with the system is based on a user-friendly graphical interface.

**Keywords:** *Software Development Methodology, Expert System, Rule Based Expert System, Object Oriented Modeling*

## 1. INTRODUCTION

In this research a prototype rule based expert system for selecting software development methodology is modeled and developed. The system is named 'SDM-ES'.

Literature survey reveals that many expert systems were reported in various branches of the field of software engineering [1]-[7]. This paper extends prior work by considering the application of expert systems technology in the domain of software development methodologies.

**Expert systems**: Expert system (ES) can be defined as: A program that attempts to mimic human expertise by applying inference methods to a specific body of knowledge [8]. This body of knowledge is called the domain of ES. The three major components of ES are: Knowledge base (KB), inference engine (IE), and user interface (UI). For better interaction with users an ES should preferably contain an explanation subsystem component or justifier [9] [10].

The knowledge base contains the relevant knowledge necessary for understanding and formulating the ES domain. In rule-based expert systems that are supported with a database, the knowledge base is modeled to include two components: (1) rule base of heuristic rules that are used to solve specific problems in a particular domain, and (2) database of domain's data and facts. The inference engine is the component that provides a methodology for reasoning and formulating conclusions. The inference engine provides directions about how to use the system's knowledge to solve problems. The user interface consists of all screens of interaction between the user and the ES. Explanation subsystem helps in justification of ES conclusions by tracing conclusions to their sources and showing how was a certain conclusion reached.

Since the inference engine is common to different systems, expert systems are practically developed using specialized ES software packages known as ES Shells. ES shells support all major ES components including an 'empty' knowledge base that can be filled with domain's knowledge and constructed according to the model adopted by the ES developer.

**Software development methodologies**: A software development methodology (SDM) -also called systems development methodology- is a formalized approach for the development of software. Although there are many different SDMs, there are fundamental systems development life cycle (SDLC) activities which are common to all methodologies. These activities or 'phases' are briefly described below [11]:

1. Planning: It is the fundamental process of understanding why a software system should

be developed and determining how the project team will go about building it.

2. Analysis: The analysis phase answers the questions of what the system will do (requirements gathering), who will use the system, and where and when it will be used.

3. Design: The design phase determines how the system will operate (in terms of software, hardware, and network infrastructure), the user interface, and the specific programs, databases, and files that will be required.

4. Implementation: During this phase the system is actually built. It includes system construction, testing, installation, and post-implementation support and improvement.

Literature survey revealed many SDMs used in software industry. The major SDMs include: Waterfall, parallel, v-model, iterative, system prototyping, throwaway prototyping, reuse-oriented, extreme programming (XP), and Scrum development methodologies [11]-[13].

The rest of this paper is organized as following: Section 2 presents the summary of a sample SDM. Section 3 discusses the selection criteria of an appropriate SDM. The major components of the proposed expert system and a sample system consultation are addressed in Section 4. Finally, Section 5 concludes with the conclusion and future work.

## 2. SUMMARY OF A SAMPLE SDM

In order to understand and have more insight of the domain and the knowledge base of the developed expert system (SDM-ES) a summary of a sample SDM is presented here. This summary is related to iterative development. For detailed coverage of various SDMs and their classifications the reader is referred to software engineering and systems analysis reference books [11]-[13].

**Iterative SDM**: This methodology divides the intended system into a series of versions. The planning and analysis phases identify the overall system concept, and the requirements are categorized into versions that are developed sequentially. The first version of the system contains the most fundamental and important requirements. The analysis phase then leads into the design and implementation, but only with the set of requirements identified for version 1 (Figure 1). After implementing version 1, work starts on version 2. Additional analysis is performed on the basis of the previously identified requirements and new ideas and comments that arose from the users' experience with version 1. Version 2 then is designed and implemented, and work immediately begins on the next version. This process continues until completing the development of the overall system [11].

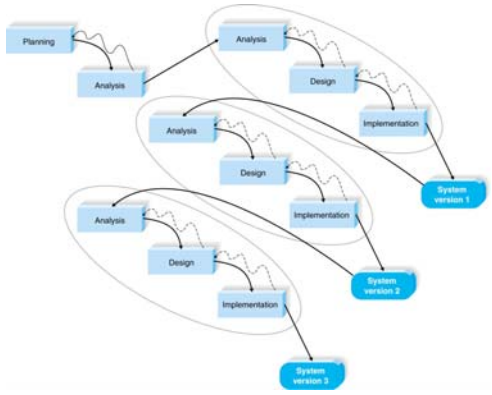


Figure 1. Iterative SDM [11]

Iterative development methodology has the advantage of quickly getting a useful initial system into the hands of the users. In addition, because users begin to work with the system sooner, it is more likely to identify important additional requirements. The major drawback to iterative development is that users begin to work with systems that are intentionally incomplete. It is crucial to identify the most important and useful features and include them in the first version, while managing users' expectations with subsequent versions [11].

## 3. SELECTING THE APPROPRIATE SDM

Since there are many methodologies, one of the challenges faced by software engineers is to decide which methodology to apply in a software project. Selecting a methodology depends on many factors and project features and no one methodology is ideal or always the best [11]-[13]. Therefore, it is useful and practical to apply expert systems technology in this domain and this is the core objective of this research.

Important methodology selection criteria include: Project time, clarity of user requirements, familiarity with technology, system complexity, system reliability, and schedule visibility. Following is a brief description of these six important SDM selection criteria [11].

**Project time**: Projects that have short time schedules are well suited for methodologies that are

designed to increase the speed of development. Prototyping, iterative development, and XP are excellent choices when project time is short because they best enable the project team to adjust the system functionality on the basis of a specific delivery date, and if the project schedule starts to slip, it can be readjusted by removing functionality from the version or prototype under development. The waterfall methodology is the worst choice when time is critical because it does not allow for easy schedule changes.

**Clarity of user requirements**: When the user requirements are unclear or subject to change, it is difficult to understand them by talking about them and explaining them with written reports. Users normally need to interact with the software to really understand what the new system can do. System prototyping, throwaway prototyping, and XP are usually more appropriate when user requirements are unclear or unstable because they provide prototypes for users to interact with early in the SDLC.

**Familiarity with technology**: When the system will use new technology with which the system analysts and programmers are not familiar, early application of the new technology in the SDLC will improve the chance of success. If the system is designed without some familiarity with the technology, risks increase because the tools may not be capable of doing what is required. Throwaway prototyping is particularly appropriate for a lack of familiarity with technology because it explicitly encourages the developers to develop design prototypes for areas with high risks. Iterative development is good as well because it creates opportunities to investigate the technology in some depth before the design is complete. Although one might think system prototyping would also be appropriate, it is much less so, because the early prototypes that are built do not investigate the new technology deeply. Usually, it is only after several prototypes and several months that the developers discover problems in the new technology.

**System complexity**: Complex systems require careful and detailed analysis and design. Throwaway prototyping is particularly well suited to such detailed analysis and design, but system prototyping is not. The traditional structured methodologies can handle complex systems, but without the ability to get the system or prototypes into users' hands early on, therefore, some key issues may be overlooked.

**System reliability**: For some applications reliability is critical (e.g., medical equipment), whereas for other applications it is merely important (e.g., games). Throwaway prototyping is the most appropriate when system reliability is a high priority, because it combines detailed analysis and design phases with the ability for the project team to test many different approaches through design prototypes before completing the design. System prototyping is generally a poor choice when reliability is critical, because it lacks the careful analysis and design phases that are essential for dependable systems.

**Schedule visibility**: Determining whether a project is on schedule is one of the challenges in software systems development. Methodologies in which design and implementation occur at the end of the project are 'poor' regarding this criterion (e.g. waterfall development) whereas methodologies that move many of the critical design decisions to an earlier point in the project can help project managers recognize and address risk factors and determine whether a project is on schedule.

For detailed criteria description, and comparison between SDMs the reader is referred to Ref. [11] which also presents a tabulated comparison as shown in Table 1 below.

Table 1. Criteria for selecting a methodology (comparing iterative and XP development methodologies) [11]

| Usefulness in developing systems | Iterative | Extreme Programming |
|---|---|---|
| with short time schedule | Excellent | Excellent |
| with unclear user requirements | Good | Excellent |
| with unfamiliar technology | Good | Poor |
| that are complex | Good | Poor |
| that are reliable | Good | Good |
| with schedule visibility | Excellent | Good |

## 4. THE DEVELOPED ES (SDM-ES)

**4.1. The Knowledge Base and Inference Engine**: System's knowledge is compiled from domain experts and the knowledge available in the literature [11]-[13]. The knowledge of the developed prototype system consists of facts (database) and rules (rule base). SDM-ES is developed using Kappa-PC Expert System Shell [14]. Kappa-PC is suitable for the system's domain because it enables the application developer to build rule-based expert systems with inference capabilities (inference engine), object oriented

modeling & programming, list processing, and graphical user interface. The database of the system is modeled as an object oriented database in which SDMs are represented as objects descendent from a root (parent) class called Software Development Methodology (Figure 2).

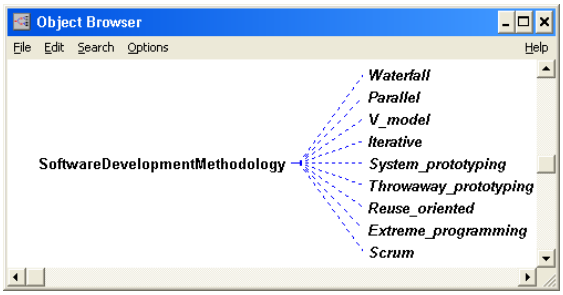


Figure 2. The object hierarchy

The object oriented modeling of the database is very suitable and promising for system's domain because each methodology is conveniently modeled as a distinct object that contains all data related to that particular methodology. This feature is also of crucial importance for updating and extending the system because: (1) updating a methodology requires changes to that specific object only, and (2) adding new methodologies to the database is simply done by adding new objects without affecting the integrity of the whole system.

Regarding the rule base 'production rules' knowledge representation model is adopted in the proposed ES for a number of reasons, including easy understandability of rules, their syntax is simple, rule chaining is easy to trace and evaluate, enhanced explanation facilities, and additional rules can easily be tested and added into the rule base.

The prototype ES in its present state contains nine SDMs and six selection criteria. The SDMs currently represented in the knowledge base are: Waterfall, parallel, v-model, iterative, system prototyping, throwaway prototyping, reuse-oriented, extreme programming, and Scrum development methodologies. The present selection criteria are: Project time, clarity of user requirements, familiarity with technology, system complexity, system reliability, and schedule visibility. Each SDM is assigned an appropriate rating value against each selection criterion. Currently the rating values are: Excellent, Good, and Poor as illustrated in Table 1 [11].

The core reasoning method (inference) of the system has two stages as will be clarified below.

Stage 1: In the first stage If-Then rules are applied in a forward-chaining manner to assign 'points' to each SDM under each selection criterion selected by the user. As an illustrating example consider the following If-Then rule category written in English-like syntax:

If: User requirements clarity is Low (i.e., unclear user requirements)

Then: Check SDMs for this property and assign them 0, 0.5, or 1.0 point for poor, good, or excellent values respectively.

Thus, according to this logic and with reference to Table 1, iterative development methodology is assigned 0.5 point and extreme programming is assigned 1.0 point under clarity of user requirements criterion.

Stage 2: The objective of this stage is determining and ranking the suggested SDMs. The list of suggested SDMs includes all SDMs that have 0.5 or 1.0 as their assigned points under all selection criteria chosen by the user. It should be clear that whenever a methodology is assigned 0 under any selection criterion (for being a 'poor' choice), then it can never be listed as a suggested solution. This implies that 'winning' SDMs that are suggested by the expert system have all their rating values as either Excellent or Good under all selection criteria selected by the user. After determining the list of candidate methodologies, each methodology is given a final score or grade by adding all its points. Consequently suggested methodologies are ranked and displayed to the user in descending order according to their final grades. Scoring detail is displayed to the user as a part of the explanation subsystem (justifier) of the system.

**4.2. User Interface and Sample System Consultation**: Interactions between the users and the system are supported through a friendly graphical user interface running under Windows environment. Figure 3 shows the main screen of the system where various options are displayed. The button titled View Available SDMs allows the user to browse and read all present data on the currently available SDMs. The user can start a consultation session by clicking on the button titled Consultation. The user can also enter the user manual and get more help by selecting the HELP button, or exit the system by clicking Exit. Other major screens and features of the user interface will be described by the following sample consultation: Consider a software project that is mainly characterized by the following key challenges: It is a complex software system, with unclear user requirements, and limited project time (i.e., short time schedule). Figure 4 shows the 'multiple

selection' menu from which the user can choose essential selection criteria. In this case the selected criteria are: Clarity of requirements, system complexity, and project time (as shown in Figure 4). Subsequent screens ask the user to specify the values of the selected properties (Figure 5). Based on the selected criteria and user inputs, the system presents the suggested SDMs in a ranked order as shown in Figure 6. From the screen of Figure 6 the user can click on the button titled Explanation in order to obtain the explanation or justification screen shown in Figure 7.

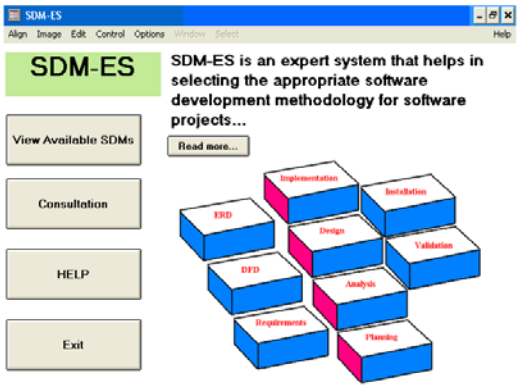


Figure 3. Main screen of the system

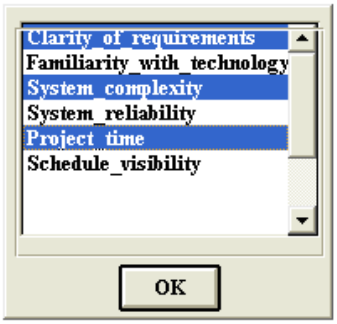


Figure 4. Choosing essential selection criteria (multiple-selection menu)

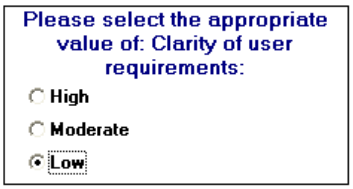


Figure 5. Sample user input screen

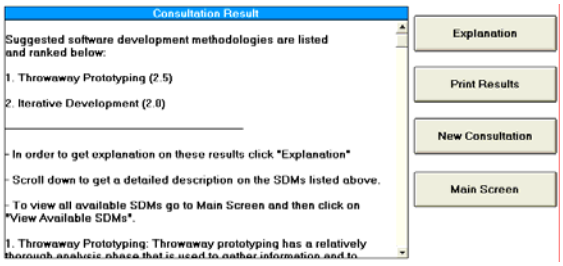


Figure 6. Sample consultation result screen

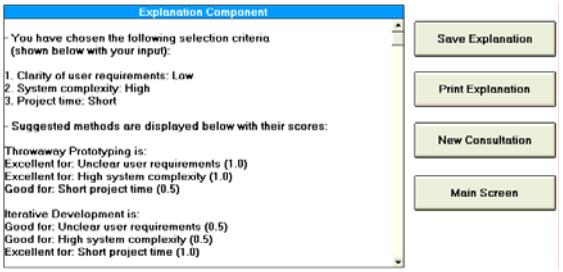


Figure 7. Sample explanation screen

## 5. CONCLUSION AND FUTURE WORK

This paper presented the modeling and development of a rule based expert system for selecting a suitable software development methodology according to software project features. By combining rule based knowledge representation with object oriented database modeling, a flexible and extensible prototype expert system could be developed. The system can be improved in several ways. Some areas of system improvement and future work are: Adding more software development methodologies, adding more selection criteria, prioritizing selection criteria, and interfacing the system to existing computer aided software engineering (CASE) tools.

## REFRENCES:

[1] LIU Yu-xi and LU Zhong-ning, "The Construction of a Web Based Expert System in Software Engineering Measurement", *Journal of Henan Normal University (Natural Science)*, Vol. 36, No. 4, 2008.

[2] He Qing, "A Software Engineering Measurement Expert System", *M.Sc. Thesis*, University of Calgary, 2003.

[3] M. Georgiopoulos, I. Dagher, G. L. Heileman, G. Bebis, I. Vlahavas, I. Stamelos, I. Refanidis, and A. Tsoukias, "ESSE: An Expert

System for Software Evaluation", *Knowledge-Based Systems*, Vol. 12, No. 4, August 1999, pp. 183-197.

[4] J. S. Mertoguno, R. Paul, N. G. Bourbakis, and C. V. Ramamoorthy, "A Neuro-Expert System for the Prediction of Software Metrics", *Engineering Applications of Artificial Intelligence*, Vol. 9, No. 2, April 1996, pp. 153-161.

[5] N. S. Bukovsky, "Building an Expert System for Software Quality Evaluation", *Microprocessing and Microprogramming*, Vol. 28, No. 1-5, March 1990, pp. 179-182.

[6] C. L. Ramsey and V. R. Basili, "An Evaluation of Expert Systems for Software Engineering Management", *IEEE Transaction on Software Engineering*, Vol. 15, No. 6, 1989, pp. 747-759.

[7] B. I. Blum and R. F. Wachter, "Expert System Applications in Software Engineering", *Telematics and Informatics*, Vol. 3, No. 4, January 1986, pp. 237-262.

[8] Keith Darlington, "The Essence of Expert Systems", Prentice Hall, 2000.

[9] E. Turban, J. E. Aronson, and T. P. Liang, "Decision Support Systems and Intelligent Systems", 7th Edition, Prentice Hall, 2005.

[10] S. Russel and P. Norvig, "Artificial Intelligence: A Modern Approach", 3rd Edition, Prentice hall, 2010.

[11] A. Dennis, B. H. Wixom, and R. Roth, "Systems Analysis and Design", 4th Edition, John Wiley & Sons, Inc., 2009.

[12] I. Sommerville, "Software Engineering", 9th Edition, Addison Wesley, 2010.

[13] R. S. Pressman, "Software Engineering: A Practitioner's Approach", 7th Edition, McGraw-Hill, 2010.

[14] KAPPA-PC 2.4 Reference Manual, IntelliCorp, Inc., USA, 1997.

## AUTHOR PROFILE:

**Dr. M. Ayman Al Ahmar** is Assistant Professor and the Deputy Dean of the College of Engineering and Information Technology, Fujairah Campus, Ajman University of Science and Technology, UAE. He received his B.Sc. (1994), M.Sc. (1997), and Ph.D. (2001) degrees from Middle East Technical University (METU), Ankara, Turkey. His current research interests include Artificial Intelligence, Software Engineering, and Engineering Information Systems. He is a member of IEEE and IEEE Computer Society.