A Validation Model for ERP systems

Dr Nada M. Alhakkak
Computer Science Department, Baghdad College for Economic Science University, IRAQ
nadahakkak@hotmail.com, dr.nada@baghdadcollege.edu.iq

Abstract: ERP system became one of the main infrastructures for the organizations and manufacturing industry for the facilities it provides such as the integration between all the organization system which make the control of all the available resource easy, fast and controlled, implementation the ERP system is not an easy issue, many projects has been failed for many reasons; lack of knowledge, Requirements inconsistency, incompleteness and uncertainty, in this paper, a propose for a new method to solve the implementation based on the knowledge base which saves all the knowledge about the ERP system Modules and the new part developed to satisfy customers' requirements not available in the existing modules. In the same time, this work proposed a new validation method for the proposed method to ensure its validity.
Keywords: ERP, validation, model, requirements, knowledge.

1.Introduction

This paper goes through different aspects in the pre-implementation phases which include the most critical phase; the requirement definition for the implementation of the ERP systems (Gap analysis) which will avoid the unexpected difficulties and to make sure that the implementation which will include the customization for the needed modules will satisfy the organization's needs. □

The main point in this paper is to find the chain that connects all the per-implementation and the implementation phases which will reflect as a result to a successful ERP system using the knowledge base system that will provide the designers and the implementers with the needed modules and components in addition to the accuracy rate -using the accuracy grid- for each one based on the input goals and the requirements from the stakeholders.

There are row levels of ERP to fit the customers' requirements; the existing modules and features, and the customization that will provide the customer with more facilities and to satisfy the customer's business process needs [6]. There are many important activities before we decide which ERP system the organization will adapt; these activates depend on the budget, also if there is an existing legacy system and if there is any integration with the ERP system, the evaluating of the staff and the evaluating of the ERP capabilities.

However, for any ERP system to be validated, it should go through multiple validation steps for more evaluation; the developer can choose the suitable validation technique among multiple methods or techniques.

This paper is organized as follows; background, literature review, methodology, Discussion and Future Direction, conclusions and finally references

DOI: 10.5281/zenodo.3715385

2.Background

Validation for any ERP system, is the process of evaluating software at the end of the software development process to ensure compliance with software requirements, its main question is: "Am I building the right product?". There are many validation techniques, these are as follows:

2.1 Simple manual techniques.

Five relatively simple and easily implemented manual technique reading, cross-referencing, interviews, checklists, and models-can provide much valuable information for meeting verification and validation criteria.

Reading. Having someone other than the originator read the specification to identify potential problems is often referred to as "reviewing." Here, however, we call it "reading" and reserve the term "reviewing" for a more formal activity. Since reading subjects the specification to another point of view, it is very good for picking up any blind spots or misconceptions that the specification developer might have. This is particularly true if the reader is going to be one of the product's testers, users, maintainers, interfaces, or program developers; a tester can, for example, verify that the specification is testable and unambiguous. Another strength of reading is that it requires little preparation. It is also extremely flexible concerning when, to where, and to what level of detail it is done. Reading's strong points can turn into weak points if the "little preparation" it does require is not carried out. Readers can waste a lot of time looking for the wrong things or looking for nothing in particular-that could have been spent bringing a valuable perspective to focus on a set of significant issues. This is a particular danger on large projects. Still, reading is fundamentally limited in the extent to which it can be used to verify detailed specifications, completeness and consistency, or the feasibility of a complex system's performance requirements.

Manual cross-referencing. Cross-referencing goes beyond reading; it involves constructing cross-reference tables and various diagrams-for example, state transition, data flow, control flow, and data structure diagrams- to clarify interactions among specified entities. These entities include functions, databases, and interfacing hardware, software, or personnel. Manual cross-referencing is effective for the consistency (internal, external, and traceability) and closure properties of a specification, particularly for small to medium specifications. For large specifications, it can be quite cumbersome, leading to the suggested use of automated aids. If these are not available, manual methods are still recommended; the payoff will outweigh the cost and time expended. Manual cross-referencing will not do much to verify the feasibility of performance requirements or to validate the subjective aspects of human engineering or maintainability provisions.

Interviews. Discussing a specification with its originator will identify potential problems. With minimum effort, you can find out a great deal about its strengths and weaknesses; this will allow you to deploy your validation resources most effectively by concentrating on the weak points. Interviews are particularly good for identifying potential blind spots, misunderstandings, and high-risk issues. But, like spot-checking, they only identify and scope the specification's major problem areas; the detailed validation work remains to be done.

Checklists. Specialized lists, based on experience, of significant issues for assuring successful software development can be used effectively with any of the manual methods

DOI: 10.5281/zenodo.3715385

described above. Checklists are excellent for catching omissions, such as the missing items, functions, and products discussed under "Completeness." They are also valuable aids in addressing some of the life-cycle feasibility considerations: human engineering, maintainability, reliability and availability, portability, security and privacy, and life-cycle efficiency. But they are not much help in verifying the feasibility of performance requirements. or in dealing with detailed consistency and closure questions. One danger with checklists is that items might be considered absolute requirements rather than suggestions. For example, several features in a portability checklist will not be necessary if the software will never be used on another machine; adding them blindly will just incur unnecessary expense.

Manual models. Mathematical formulas can be used to represent and analyze certain aspects of the system being specified. These manual models are very good for analyzing some life-cycle feasibility issues, particularly accuracy, real-time performance, and life-cycle cost. They are also useful for risk analysis. They are not, however, much help in validating the details of a specification's consistency and completeness or in assessing subjective factors. Their manual nature makes them inefficient for detailed feasibility analysis of large, complex systems, but they are good for top-level analysis of large systems.

Simple scenarios. Scenarios describe how the system will work once it is in operation. Man-computer dialogues are the most common form of simple scenarios, which are very good for clarifying misunderstandings or mismatches in the specification's human engineering aspects but not for checking completeness and consistency details or for validating performance speed and accuracy.

2.2 Simple automated techniques.

Automation extends the power of two manual techniques-cross-referencing and simple modelling.

An automated cross-referencing. Automated cross-referencing involves the use of a machine-analyzable specification language-for example, SREM-RSL, Software Requirements Engineering Methodology-Requirements Statement Language, Problem Statement Language/Problem Statement Analyzer; or PDL, Program Design Language." Once a specification is expressed in such a language, it can be automatically analyzed for consistency, closure properties, or presentation of cross-reference information for manual analysis. Automated, cross-referencing is excellent for validating the detailed consistency and closure properties of both small and large specifications. Using a formatted specification language also eliminates many testability and ambiguity problems because the language forms help prevent ambiguous terms and vague generalities. The automated systems also have less of a problem in checking for additional clerical errors introduced in iterating and retyping a specification. Current automated specification aids have only limited capabilities in addressing the accuracy and dynamic performance issues, and some, particularly SREM-RSL, are not available on many computers. Although their performance on small and medium specifications has improved considerably, they are still somewhat inefficient in performing consistency and completeness checks on very large specifications. Even so, the costs of using them on large specifications are more than repaid by the savings involved in early error detection.

Simple automated models. Mathematical formulas implemented in a small computer program provide more powerful representations than manual models for analyzing such

DOI: 10.5281/zenodo.3715385

life-cycle feasibility issues as accuracy, real-time performance, and life-cycle costs. Simple automated models are especially good for risk and sensitivity analysis, but, like manual models, are not much help in verifying detailed consistency and completeness, in assessing subjective factors, or in performing detailed feasibility analysis of large, complex systems.

2.3 Detailed manual techniques.

Detailed scenarios and mathematical proofs are especially effective for clarifying human engineering needs and for verifying finite-mathematics programs, respectively.

Detailed scenarios. Detailed scenarios, which provide more elaborate-and thus more expensive -operational descriptions, are even more effective than simple scenarios in clarifying a system's human engineering aspects.

Mathematical proofs. Mathematical transformation rules can be applied to a set of statements, expressed in a precise mathematical specification language, to prove desired properties of the set of statements. These properties include internal consistency, preservation of 'invariant" relations, and equivalence of two alternate sets of statements (e.g., requirements and design specifications). For certain classes of problems small problems involving the use of finite mathematics proofs offer a near-certain guarantee of the properties proved. But mathematical proofs cannot deal with no formalized inputs (e.g., noisy sensor data, natural language); cannot deal conveniently with "real-variable" calculations or with many issues in synchronizing concurrent processes, and cannot deal efficiently with large specifications.

2.4 Detailed automated techniques.

Two final techniques-detailed automated models and prototypes-provide the most complete information.

Detailed automated models. Detailed automated models typically involve large event simulations of the system. While more expensive than simple automated models, they are much more effective in analyzing such issues as accuracy, dynamic consistency, real-time performance, and life-cycle cost. The process of generating such models also serves as a good way to catch specification inconsistencies and ambiguities.

Prototypes. In many situations, the feasibility of a specification cannot be conclusively demonstrated without developing some working software. Examples include some accuracy and real-time performance issues, as well as user interface suitability. In such cases, it is often valuable to build a throwaway prototype of key portions of the software: a program that works well enough to resolve the feasibility issues but lacks the extensive features, error-proofing, and documentation of the final product. The process of building the prototype will also expose and eliminate some ambiguities, inconsistencies, blind spots, and misunderstandings incorporated in the specification.[13]

3.Literature Review

Requirements Integration Model (RMI) can be used to increase the quality of the requirements; it consists of the Workflow model which can be used to identify the stakeholders, business process and functional specification, on the other hand, the work procedures used to ensure of the quality of development using the work and documentation standers and case tools [1].

Another technique which confesses the analyst to work with the user to understand the organization business process using the narrative analysis interviews as a pattern to

DOI: 10.5281/zenodo.3715385

convey the requirements which consider as a powerful tool as the authors claimed [2]. We can manage the ERP requirements by classifying it into logical classes (products, process, etc…) and subclasses (performance, GUI, etc…) until we reach the requirements (speed, availability, throughput, etc…) to improve the developers conception of the requirements and in the same time keep track for each requirement using predefined tags that indicate the status of the requirements (completely implement, not implemented yet, partially implemented) [3]. Analysis the stakeholders requirements concerning particular context, environment, and maybe individual users using the contextual design to ERP requirements specification can make the business process more efficient; the model depends on the consolation and redesign the modules derived from data collected to generate the analysis context of the work and requirements specification of the ERP system [4][5].

ERP requirements-driven approach [6] is an iterative process to match the customer's requirements with the ERP system functionality to organization requirements, this paper claimed the success of the ERP implementations depends on the customization process. Formal Concept Analysis (FCA) [7] is a mathematical technique for systematically combining and organizing individual concepts of a given context into a hierarchically ordered conceptual structure – concept lattice. We are proposing to apply FCA in analyzing the association between a set of test scenarios with a set of transitions specified in a UML state machine model. By applying the concept analysis mechanism, we can determine a minimal set of test scenarios that can sufficiently cover all the transitions for requirements validation.

The presentation of a comprehensive method for defining and validating the requirements of a system, based on formal analysis, automatic scenario generation, and support for rapid prototyping. Model-checking is used in the formal analysis to identify internal inconsistencies in the specification and to generate interesting 'scenarios' from the specifications that can help in identifying potentially incomplete or anomalous requirements. Prototype tools have been built for the automatable steps of the method, and it has been experimentally used to model and validate the requirements of two systems. The experiments resulted in the identification of many inconsistencies and anomalies in the requirements of each of these systems [8].

Scenarios and goals are effective and popular techniques for requirements definition. Validation is essential to ensure that they represent what stakeholders want. Rather than validating scenarios and goals separately, possibly driving the elaboration of one through the validation of the other, this paper focuses on exploiting the relationship between goals and scenarios. The aim is to provide effective graphical animations as a means of validating both. Goals are objectives that a system is to meet. They are elaborated into a structure that decomposes declarative goals into goals that can be formulated in terms of events that can be controlled or monitored by the system. Scenarios are operational examples of system usage. The relation between scenarios and goals is established through fluent that describe how events of the operational description change the state of the basic propositions from which goals are expressed. Graphical animations are specified in terms of fluent and driven by a behaviour model synthesized from the operational scenarios [9].

The presentation of a tool that provides effective graphical animations as a means of validating both goals and software designs. Goals are objectives that a system is

DOI: 10.5281/zenodo.3715385

expected to meet. They are decomposed until they can be represented as fluent. Animations are specified in terms of fluent and driven by behavior models [10]. Requirements modelling depends on having available knowledge of what should be modelled. Acquiring, or eliciting, this knowledge is recognized to be a hard problem, and different suggestions have been made to address it. In this paper, we examine one such approach-viewpoint resolution. It is based on the fact that software requirements can and should be elicited from different viewpoints, and that examination of the differences resulting from them can be used as a way of assisting in the early validation of requirements. Our research proposes a language for expressing views from different viewpoints and a set of analogy heuristics to perform a syntactically oriented analysis of views. This analysis of views is capable of differentiating between missing information and conflicting information, thus providing support for viewpoint resolution [11].

Use cases are analytical descriptions of how a system should react in interaction with actors. Use cases are appropriate as a specification of a system required behavior. A scenario is an example of execution of a system that may be defined with the intention that it should be supported or the intention that it should be avoided. As such scenarios are appropriate for requirements validation. This paper presents an approach where use cases and scenarios are used to complement each other for requirements engineering. Use cases define what a system should do, and scenarios are used to validate use cases [12].

4.Methodology

One of the major problems the ERP's implementation team still faces are the Requirements inconsistency, incompleteness and uncertainty which might be the result in delivering the system late, cost more than the expected budget, incomplete implementation; in addition to the difficulties in the implementation due to the size, scope and complexity of the modules, all of these reasons might push the organization to drop the ERP project.

The previous researches omit the point that the customization process will provide us with new knowledge that might be useful in the future, besides, the alignment requirements process with the ERP functionality is a process depends on the software engineers who might have different experiences to decide and make the right decision. As a result of these problems the proposed process will work as the base for the decision making to take the action which of the available components will satisfy the customers' needs and the new knowledge that will come from the customization and development for the new additional components will be used to feedback the knowledge base to ensure that the new features will satisfy the same requirements in the future.

To solve the previous problems we suggest a new process that contains a knowledge base system (KBS) about the ERP module and customization developed in addition to the next steps that contain validation processes for the decisions made by the ERP KBS and customization for new requirements and features to satisfy the customers' needs, Also a validation model will be applied on critical parts of the proposed process to ensure it's requirements validity, as in the following two figures.

DOI: 10.5281/zenodo.3715385

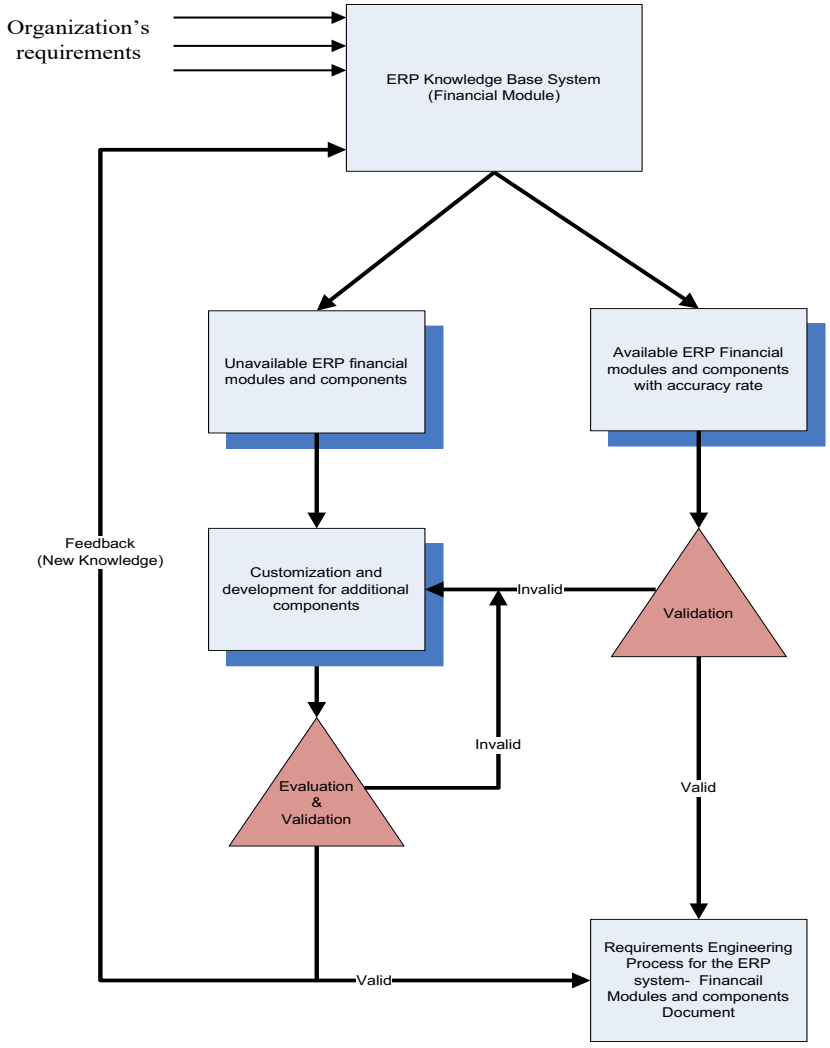

Fig1.Proposed validated ERP-KBS system

DOI: 10.5281/zenodo.3715385

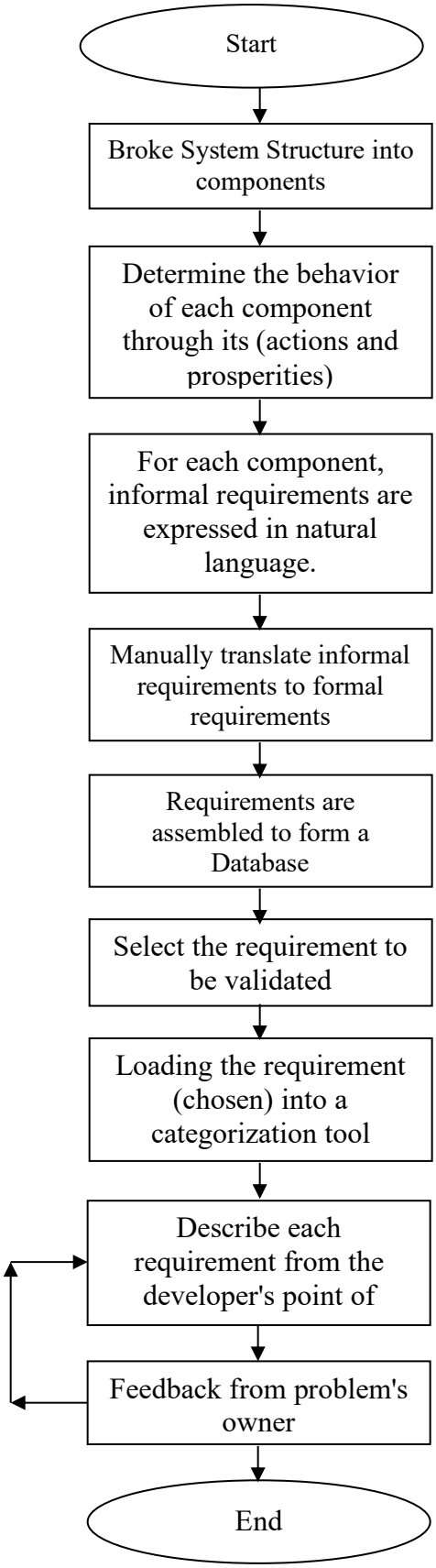

Fig2.Validation Check

DOI: 10.5281/zenodo.3715385

5.Discussion and Future Directions

Mainly, any organization has multiple types of requirements; like customer's requirements. For ERP Knowledgebase system (KBS) for the financial module should contain all the information about the financial module, in this phase, the KBS should check the organization requirements and find the best component that will satisfy the requirements. When the module and components are available for requirements, it will include a list of the organization's requirements and the available modules and components that satisfy these requirements. The valid requirements that match the modules and components will be adding ERP system documentation; the financial modules and components will be included in the implementations. When the module and components are unavailable for requirements then we will have a list of the requirements that the KBS can't find modules and components that can satisfy it and need to be customized or develop as an external tool. To satisfy the organization's requirements we have to make sure all the requirements are satisfied; from the previous phases the KBS can find and match the requirements with the available modules and components but on the other hand, some requirements that the existing modules and components need for a customization and development processes. Add to the ERP system documentation; the financial modules and components will be included in the implementations.

6.Conclusion

The final ERP system modules and components document will include all the modules and components that satisfy the customer.

However, adopting this idea as the base for all the ERP implementations brings a huge knowledge base that will work as the mainframe in the future implementations to save time and resource, as a result, SAVE MONEY. Also, we will be quite sure that there will be no more frails in the ERP implementations because all the customers' requirements will be satisfied and ALL WILL BE HAPPY.

7.References

[1] Mutchalintungkul, A., Oonhawat, J., Pholpipatanaphong, K., Sutivong, D., & Prompoon, N. (2006, May). Experience from applying RIM to educational ERP development. In Proceedings of the 28th international conference on Software engineering (pp. 620-623).

[2] Alvarez, R., & Urla, J. (2002). Tell me a good story: using the narrative analysis to examine information requirements interviews during an ERP implementation. ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 33(1), 38-52.

[3] Sagheb-Tehrani, M., & Ghazarian, A. (2002). Software development process: strategies for handling business rules and requirements. ACM SIGSOFT Software Engineering Notes, 27(2), 58-62.

[4] Vilpola, I., Väänänen-Vainio-Mattila, K., & Salmimaa, T. (2006, April). Applying contextual design to ERP system implementation. In CHI'06 extended abstracts on Human factors in computing systems (pp. 147-152).

[5] Sutcliffe, A., Fickas, S., & Sohlberg, M. M. (2006). PC-RE: a method for personal and contextual requirements engineering with some experience. Requirements Engineering, 11(3), 157-173.

DOI: 10.5281/zenodo.3715385

[6] Rolland, C., & Prakash, N. (2001, August). Matching ERP system functionality to customer requirements. In Proceedings Fifth IEEE International Symposium on Requirements Engineering (pp. 66-75). IEEE.

[7] Ng, P. (2007, August). A Concept Lattice Approach for Requirements Validation with UML State Machine Model. In 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007) (pp. 393-400). IEEE.

[8] Sukumaran, S., Sreenivas, A., & Venkatesh, R. (2006, September). A rigorous approach to requirements validation. In Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06) (pp. 236-245). IEEE.

[9] Uchitel, S., Chatley, R., Kramer, J., & Magee, J. (2004, September). Fluent-based animation: Exploiting the relationship between goals and scenarios for requirements validation. In Proceedings. 12th IEEE International Requirements Engineering Conference, 2004. (pp. 208-217). IEEE.

[10]Chatley, R., Uchitel, S., Kramer, J., & Magee, J. (2005, May). Fluent-based Web animation: exploring goals for requirements validation. In Proceedings of the 27th international conference on Software engineering (pp. 674-675).

[11] Leite, J. C. S. D. P., & Freeman, P. A. (1991). Requirements validation through viewpoint resolution. IEEE transactions on Software Engineering, (12), 1253-1269.

[12] Some, S. S. (2005, August). Use cases based requirements validation with scenarios. In 13th IEEE International Conference on Requirements Engineering (RE'05) (pp. 465-466). IEEE.

[13] Boehm, B. W., Gray, T. E., & Seewaldt, T. (1984). Prototyping versus specifying: a multiproject experiment. IEEE transactions on Software Engineering, (3), 290-303.

DOI: 10.5281/zenodo.3715385