Analysis of Risk Factors of ERP (Enterprise Resource Planning) Systems Information Technologies

William Bazán(✉), Teresa Samaniego, Abel Alarcón, and Ana Rodríguez

School of Computer and Information Science, Agrarian University of Ecuador, Guayaquil, Ecuador
{wbazan,tsamaniego,jalarcon,arodriguez}@uagraria.edu.ec

Abstract. The present paper is about collected information on software development and the implementation of the ERP system. It aims to get to know the perception on the given and installed system as well as the factors that were introduced after the system implementation.

The study explores the critical factors that were found in the implementation of ERP systems, which invites to delve into the review of these systems. The last part of the study presents conclusions and recommendations to take into consideration by enterprises about the elements which make the ERP system become successful. They allow us to classify it into each of the main aspects. The reason is that they have a positive impact on the project. In other words, they offer a series of parameters for software development of commercial organizations such as industry, which should think about the implementation of an ERP system.

Keywords: Critical factors · ERP system · Impact · Integrated systems

1 Introduction

Systems enterprise resource planning (ERP) software systems are considered software systems of a general kind, which contain modules that aid in the differentiation of areas such as production, sales, distribution, finance, human resources, and maintenance amongst others.

ERP systems are currently used to help us in order to manage any company more successfully. Within the study of ERP systems, we must not forget the existing critical risk factors of this software system, since it does not allow us to change decisions so we should try to ensure that the decisions we make are the most appropriate ones.

First of all, we face the arduous and expensive task of approving and implementing it, which requires much effort, commitment and a fully designed, planned study.

We can mention the following critical factors found:

- Transition to the new system
- Work overload
- Difficulty of estimating costs
- Rotation of key personnel
- Adjustment legislation

- Lack of technically trained personnel
- Lack of written procedures
- Adaptation of hardware and telecommunications

It is noteworthy that these can be improved with the aid of reengineering business processes, which will proceed to address the points of failure through continued application of feedback, which could be found before in other cases of application of ERP systems, allowing us to find the maximum benefits of these systems. It should also be taken into account that in the process of implementation of ERP systems, the provider should become the implementation consultant with possible solutions.

ERP systems are an evolution of systems Production Resource Planning, MRP (Manufacturing Resources Planning), which focus on the planning of activities of manufacturing companies. Before 1960, the main focus of the systems inventory control was based on the basics of inventory [1].

During the 1960s, the first computers and the first MRP first appeared. Planning Material Requirements (MRP-I) was one of the first applications for these businesses [2].

The MRP software supported the creation and maintenance of material master data and bills of materials across all products and parts, in one or more manufacturing plants [1].

During the 1970s some of the major software vendors such as SAP, J.D. Edwards and Oracle with its renowned Structured Query Language (SQL) appeared.

After a general overview of the article, presented in the introduction, we present a brief description of the literature review divided into three main points. Later, the article presents the methodology used. Finally, the main conclusions reached are presented.

2 Quality Software Metrics, Measurement and Indicators

We begin by defining the different concepts presented in the introduction, facilitating the understanding of the subject by the reader.

Tejerina, [3] states that Quality Software is the fulfillment of the requirements of functionality and performance explicitly established, explicitly documented development standards and implicit characteristics expected of all professionally developed software. Software quality is a complex mix of factors that may vary across different applications and according to customers who request them.

The success of the software depends on a set of qualities, which are designed to give a high degree of customer satisfaction: to the revisions of comprehensive techniques before testing, to quantitatively determine what is important for a product before starting the tests, to provide assurance and reliability to have a test plan and to list the objectives of the test concisely. To produce quality software is the major goal for engineers.

Pressman, (2010) in [4] mentions that Software Quality should be added in all product life-cycle management. The various tasks for inserting a quality control in software development in this analysis are:

1. Use of methodologies and development methods.
2. Reuse formal review procedure.
3. Constant testing of software development.
4. Adjustments to set standards for software development.
5. Verification of changes, measurable calculations and accumulation of information.
6. Managing reports about development in software quality.

There are dozens of metrics or measures exclusively oriented towards product development, as it is the case with this quality software, the life cycle from its design, coding, testing and maintenance is related to the need to control different attributes, structure, accuracy, coupling and complexity of it being difficult to have a single quality value [4]. Different types of metrics allow for the quantification and qualification needed to improve the final software, with different criteria according to the appearance, technologies, and functionalities depending on user requirements, this allows for the quality to be valued from the moment we plan to develop any software.

A measure provides a quantitative indication of the amount, full, extension, size, capacity or size of some attribute of a product or process [4]. A measure is an indication of amount and size of a product, applying it in its development.

A metric is an assessment of the degree to which a system, component or process has a specific tribute (extension, quantity, dimensions, capacity or size). A software engineer collects measures and develops metrics in order to obtain indicators [4].

The metric is more directed to the evaluation factors of the system, component or procedure for a given attribute, where a group of measures exist, and they become metrics that provide us with indicators.

An indicator is a metric, or a metric combination that provides us with unknown information. This knowledge will allow the project manager or software engineers to prepare the process, project or product for it to work better [4, 5].

All these products are based on several parameters which lead software development. These parameters are required by organizations so that their applications can be efficient. They are all related: software quality, metrics, indicators.

2.1 Quality Management Software

The objective of managing software quality is to understand what the customer expectations for quality are, and to implement and plan in order to meet these expectations. After all, as we know the customer defines quality. Therefore, it is necessary to evaluate each individual quality of the software, in order to be able to determine one or more metrics that can be obtained to reflect these properties. One example of this could be the creation of a quality characteristic that guarantees that lesser amounts of errors occur. It can be measured by counting errors which are defects of a solution [6].

2.2 Risk Management Errors

"For a long time, the software projects have been considered high-risk projects prone to failure" as stated [7], and "recognizes that risk management is one of the best practices in the software industry to reduce the surprise factor" according to [8].

2.3 Errors Management

A key aspect, important in developing reliable software, is the phase analysis of error distribution as proposed in fault types defined in IEEE (1998) [9].

This analysis reveals the existence of procedures for data acquisition and technical checks, regular checks of a software product made by a team or qualified personnel, which determines its ability to use, try and recognized specifications and standards, that can be classified in requirements for reviews, analysis, design and documentation to be established at the time of the proposal with the client [8].

2.4 Measures for Error Handling

There are many techniques for the revision of a software product from its development stage, in addition to this, they give us a future vision of the results of a product.

Verification and validation can be defined as the process of ensuring that each phase of the life cycle development correctly implements the specifications of the previous phase, and that the software obtained meets its requirements. The tests imply the controlled execution of the code program looking for errors. Formal reviews are planned and periodic reviews of the products obtained carried out by developers, customers, users and managers to assess progress. Inspections and walk-throughs are systematic reviews of software products obtained by the pairs made with the purpose of finding errors [10].

2.5 Systemic Quality Model

Systemic Model of Quality in software development allows us to measure the Systemic Quality of a developer of software, which starts off with the quality of the product at the time it occurs and the quality of the process of its production. The model provides verification of a level of quality that will change between None, Basic, Intermediate and Advanced [11].

Systemic Quality Models indicate which processes need to be improved in the company and properties that have not been fulfilled in the software product developed [11].

3 Model for Defining Stages in Metric

The proposed model for defining stages in metric is shown in Fig. 1. This model is based on the work presented in [12]. As it can be seen the process consists of five different steps that are described below:

1. Identification - Stage in which the scope is defined when creating metrics, as well as asking hypothesis to carry out measurement software during its development. Obtaining information about the main requirements to be met by the metric.
2. Creation - Stage in which the metric is defined according to the requirements preset in the previous stage. You need to perform a theoretical validation based on statistical mathematical methods, among others, that ensure that the metric used meets the objective. An empirical validation through surveys, experiments and case studies validated metrics are also performed.
3. Acceptance - After obtaining a valid metric it is usually necessary to go through a stage of acceptance of the metric, testing in real environments whether the measure meets the desired objectives.
4. Application - The metric is accepted and used for the field of application for which it was created.
5. Accreditation - This stage runs in parallel with the implementation phase and aims at maintaining the metric, as a result of this stage a metric can be withdrawn, as it is no longer useful or reused to start a new process.

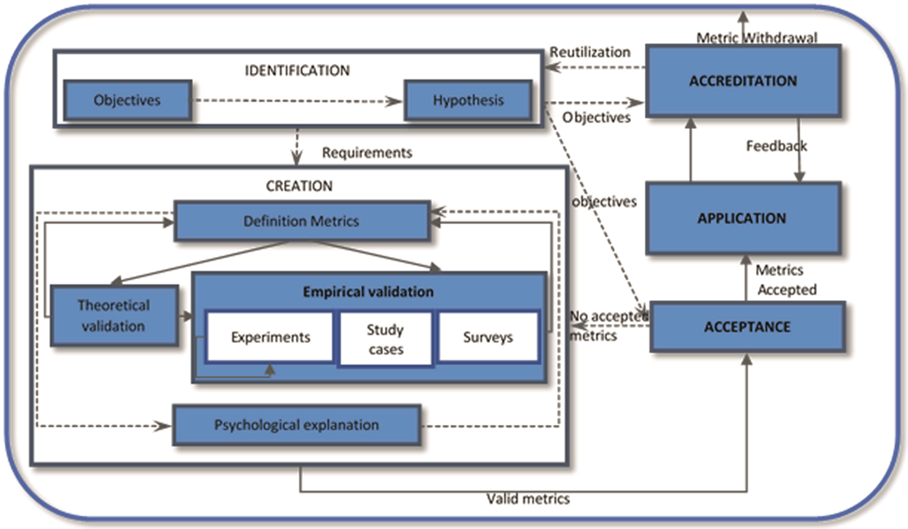

Fig. 1. Stages definition method metric.

One of the primary concerns among quality metrics is located in the measures of usability of the product produced. If the product is user friendly, it can be measured through the characteristics of how we see the end user use the system, considering the time employed in its use, the benefits provided, and how users value subjectively the system produced as mentioned [13].

Therefore, it is noteworthy that a product developed following the above steps will allow us to determine whether the software has been adequately developed, if the metric used was appropriate for the development of the product, or to identify why problems appeared. Below you can see a table of the stages of the metric model described.

3.1 Characteristics of a Functional Metric

According to Duran Rubio (2003, p. 48) “The size of the software could be measured in terms of bytes occupied on the disk, the number of programs, the number of lines of code, functionality it provides, or simply the number of screens or reports you have”. The characteristics of a functional metric are as follows:

- Technological independence. - Once we have established the required functionality we must choose technology needed to achieve this functionality.
- Simplicity. - The metric should not require great efforts to achieve a measure. A disadvantage of this feature would be that the software would not be as detailed for noticeable results, such as mathematical operations.
- Focus on the functionality provided. - This refers to the advantages to be acquired by implementing the new software, when making a technical review.
- Based on user requirements. - This gives an idea of what size the software will have before it is finished.
- Consistency. - The results obtained in different systems and different people must be consistent.

3.2 Metrics Function Points

This metric tries to measure the functionality that the software provides the user. According to Duran Rubio (2003, p. 49), “It is a metric to set the size and complexity of computer systems based on the amount of functionality required and delivered to the users” or, “The Function Points measure the logical or functional size of projects or software applications based on the functional requirements of the user”.

3.3 Standard Method Function Point Analysis

The method which is becoming standard in the industry is defined by the IFPUG, called Function Point Analysis (FPA) and its authors define it as follows:

“Standard Method for measuring software development from the point of view of the user” [14].

The selection of indicators includes efforts measurement (person-hours) and costs (in money), both real and planned ones, number of deliverables accepted by the user and deviations, reused effort from other projects or inactivity in human resources, number of modifications in the product and information on its evaluation (solicited, rejected and accepted), dedicated effort to error detection and correction, number of completed reviews, and information on error detection to evaluate product quality (i.e. errors detected before and after delivery). Many of the indicators also include broken down by phase measures (i.e. breakdown in requirements, design, coding and documentation) measurements.

3.4 Steps to Determine a Function Points Metric

In this section we describe the steps to determine a function points metric, where the method identifies the components of S.I. assigning a number of points based on the complexity function, and the sum of this gives us the unadjusted function points. The final adjustment is done at the end, taking into account the general characteristics of any computer system [15]. The different stages are shown in Fig. 2 and they are explained next in detail.

- Step 1. To determine the type of count. In this step we determine the target count, defined if it counts in the development, maintenance or if a software is already installed.
- Step 2. To identify the scope of measurement and limits of the application.
- Step 3. To count Data functions. Here we determine the data storage capacity. Both, internal logical file and the external interface are analyzed. A value of this complexity is assigned, considering the data. This value can be high, medium or low.
- Step 4. To count the Transactional functions. This step measures the ability to perform operations; each component is assigned a value of complexity (high, medium or low), considering the available data.

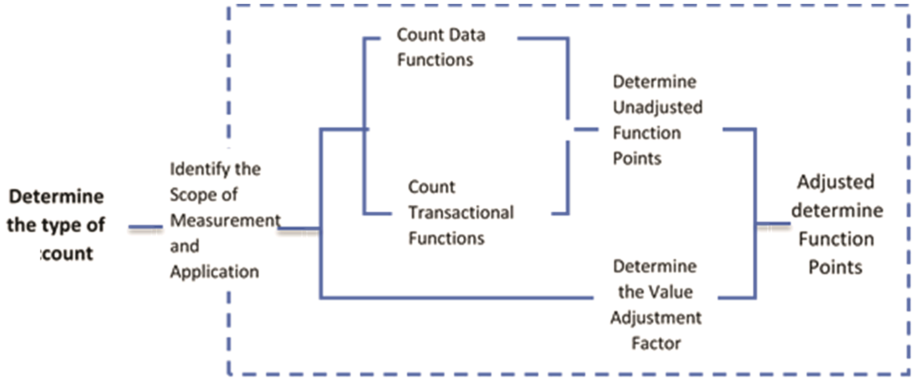
Determine the type of count
Identify the Scope of Measurement and Application
Count Data Functions
Count Transactional Functions
Determine Unadjusted Function Points
Determine the Value Adjustment Factor
Adjusted determine Function Points

Fig. 2. Steps to determine a function points metric [14].

Function points are calculated by using information parameters and the level of complexity of the software. The information parameters are five:

- Number of external input (EI). This input is the information originated about the user or the information that is transferred from other software. The input is frequently used for the updating of logic reports.
- Number of external output (EO). This output consists of data which are generated within the application and are transformed into information for the user.

- Number of external consults or reports (ECR). They are online information based on a user's requirement.
- Number of internal reports (IR). They are the data found in the database and which are updated with external input.
- Number of external reports (ER). They are data which come from storage out of the application.

Table 1 shows how these information parameters are related to the level of complexity.

- Step 5. To determine the unadjusted function points. In this phase the total is obtained by adding up the number of components according to the assigned complexity.
- Step 6. To determine the value adjustment factor. The adjustment value is obtained by adding 0.65 to the total sum of the degrees of influence of the 14 general system characteristics multiplied by 0.01.

Table 1. Calculation of the function point

CALCULATION OF THE FUNCTION POINT													
INFORMATION PARAMETERS	**TOTAL COUNTING**	LEVEL OF COMPLEXITY											Total
		LOW				MEDIUM				HIGH			
		Counting	X	Factor	+	Counting	X	Factor	+	Counting	X	Factor	
EI: EXTERNAL INPUT				3				4				6	
EO: EXTERNAL OUTPUT				4				5				7	
EC: EXTERNAL CONSULTS				3				4				6	
IR: INTERNAL REPORTS				7				10				15	
ER: EXTERNAL REPORTS				5				7				10	
				NON ADJUSTED FUNCTION POINT (NAFP):									

In order to calculate the FP, we use the following formula:

$$FP = NAFPT[0{,}65 + 0{,}01 \times SUMAF]$$

Where SUMAF (sum of the adjusted factor Table 2) are software technical parameters which are determined as it is shown in the following table. They range from 0–5, where this value is based on certain technical conditions for the development of the software.

- Step 7. To determine the adjusted function points. In this phase we considered the unadjusted function points by the adjustment factor.

Table 2. Adjust factor of technical complexity

ADJUST FACTOR OF TECHNICAL COMPLEXITY

FEATURE	VALUE
1 Data communication	
2 Distributed functions	
3 Benefits	
4 High use of configuration	
5 Transfer speed	
6 Online data input	
7 Efficiency design by the end user	
8 Online data updating	
9 Complexity of the internal logic process of the application	
10 Code reusability	
11 Ease of installation	
12 Ease of operation	
13 Multiple localizations	
14 Ease of change	
SUM OF ADJUSTED FACTOR (SUMAF)	

3.5 Software Project Risks

PMI (2013) mentions a temporary effort needed for the project to create a product complying with certain objectives [16]. Similarly, we understand risk as the event which, should it occur, produces an effect either positive or negative in any of the objectives for creating a product. For risk management software projects, we should take into account that for its definition and study we should identify, study, analyze and eliminate each and every one of the possible threats before starting the project, in order to encounter less elements of risk by the end of the project, which may impede achieving the proposed objectives.

Spector and Gifford, (2010) indicate that risks can be caused by various reasons or situations [17]. In order to improve the risk management in software projects, several guides have been published: PRINCE2, ISO standards 10006: 2003, and PMI. These guidelines provide methods and procedures focused on information systems, which help in risk management software projects.

Alba, (2008) [18] presents the stages for risk management in software projects, each of which focuses on the treatment of the risks mentioned below:

- Identify risks: It is about recognizing the potential risks which may cause problems or failures in the project.

- Analyze risks: Each of the risks is analyzed and classified by groups according to their priority.
- Assess risks: At this stage the chances of occurrence and potential grievances that may affect the project are evaluated.
- Risk treatment: we identify methods, procedures, implementations, modifications which may be needed to solve the potential risks in a project.
- Risk monitoring: This has to do with all the processes which are performed to address the risks which must be monitored, in order to ensure that controls are effective and provide valuable information to detect possible changes affecting the draft.

4 International Standard ISO

The evaluation and calculation of the functionality that a computer system has is regarded as the anguish that the whole industry dedicates to the development of the software. It is important to bear in mind that having a metric is not enough today, nonetheless, this metric should be standard so that it can be used in different companies, allowing developers to have access to, and to share indicators among the industries of software engineering products that are easy to manage and understand [19]. To compare the productivity (Function Points per person month) of a company with the industry data is critical in improvement plans [14].

5 Methodology

This research was eminently descriptive; we provided a detailed analysis of all results obtained in the exploratory study carried out, where in some organizations, ERP systems were fully settled satisfactorily. This study was supported in the review of the relevant resources available between 2013 and 2016. We examined seven implementation methodologies: Total Solution, FastTrack, ASAP, AIM, SureStep, OpenERP and Openbravo. Each of these methods was analyzed according to the proposed unified methodology of [20], which raises the following essential elements:

1. Project Management, which holds the project planning and scheduling, monitoring and feedback and risk management
2. Managing change as there seems to be a lack of focus on issues such as: activities, processes, methodologies associated with the understanding of this process, which in some cases have led to the failure of such projects.
3. Training, the complexity of these applications requires rigorous training, which if not carried out can lead to drastic consequences and is considered one of the main reasons for failure of ERP implementations.
4. Implementation level: Strategic, evaluation of current legacy systems; Tactical; Operating.

The critical factors found are studied in depth in the review of these systems, which are made including some questions about how efficient the systems are with senior management, which is the selection process for the system, composition when choosing the team, training and user participation, and the different problems facing their organizations during the implementation process and after.

This study ends by giving recommendations to companies about the issues that must be taken into account in order to implement ERP systems successfully.

6 Conclusions

Critical risk factors are presented as part of an ERP system. This research has been developed after a thorough exploratory study which does not transcend beyond the issue of the ERP. The type of solution that the author provides is reasonable in terms of representation and information shared.

Surveys, interviews and questionnaires let us know about the shortcomings of the proposed system, and after experimentation, in some companies we were able to observe some critical aspects of these systems. Thus, after a very detailed study, the author proposes feedback in order to provide solutions, where the provider must go beyond the questions raised.

Total Solutions methodologies and Fast Track, used by consulting houses, do not endorse any particular software, thus, they are more general in its recommendations and include considerations oriented more towards project management and change management. Nonetheless, it should be mentioned that Deloitte Fast Track offers on its website, its methodology focused on versions of both products SAP and Oracle.

A fully documented system with a comprehensive literature allows companies to incorporate an ERP system and thus, providing technical information regarding the proper use of this resource, which must be based on the implementation of the system for its later adjustment to an existing system.

The final conclusion of this study eradicates in the importance given to the selection process of an ERP system, taking into consideration the fact that critical risk factors can affect my current system, which measures prevent chaos in case of replacement of the current system, with one that provides more comfort to our company, and how reliable this system is if we do not have the necessary data, as mentioned is an exploratory study where the main shortcomings of these systems were found, in cases where there was not information about problems that the use of ERP presents. Good planning and integration with the systems fosters the necessary change in working procedures, and proper selection of the supplier, to deliver the fully integrated system to our company, provides us with information about the process of implementation of the system with all its resources, together with the necessary support for our organization.

References

1. Metaxiotis, K.S., Psarras, J.E., Ergazakis, K.A.: Production scheduling in ERP systems: an AI based approach to face the gap. Bus. Process Manag. J. **9**(2), 221–247 (2003)
2. Orlicky, J.A.: Material Requirements Planning—The New Way of Life in Production and Inventory Management. McGraw-Hill, New York (1975)

3. Tejerina, W.C.: Product dimensions for the software. Jujuy National University of Jujuy (2014)
4. Pressman, R.: Software Engineering, 5th edn. McGraw Hill, New York (2010)
5. Pressman, R.: Good Practices. McGraw Hill, New York (1998)
6. Scalone, F.: Estudio Comparativo de los Modelos y Estándares de Calidad de Software. Buenos Aires, pp. 22–23 (2006)
7. Bannerman, P.: Risk and risk management in software projects: a reassessment. J. Syst. Softw. **81**, 2118–2133 (2008)
8. Dolado, J.J., Aguirregoitta, A., Presedo, C.: Study metrics for project control software. In: Proceedings of the Conference on Software Engineering and Database, SISTEDES, Barcelona pp. 65–72 (2010)
9. IEEE: Standard Dictionary of Measures to produce reliable software. IEEE (1998)
10. Thayer, R.H.: Software engineering project management, 2nd edn. Wiley - IEEE Computer Society Press, New York (2001)
11. Mendoza, L., Perez, M., Griman, A.: Prototype systemic quality model (MOSCA) software. Comput. Syst. **8**, 198–199 (2005)
12. Serrano, M., Piattini, M., Calero, C., Genero, M., Miranda, D.: A method for defining software metrics. ALARCOS Group, University of Castilla (2010)
13. Gorga, G., Madoz, M., Heavy, P.: Towards a proposal of metrics for evaluating educational software. Laboratory Research and Development in Informatics, pp. 16–18
14. Duran Rubio, S.E.: Points for function. Metric standard to set the size of the software. Comput. Policy Bull. **6**, 50–63 (2003)
15. Duran Rubio, S.E.: Puntos por función. Métricas estándar para establecer el tamaño del software. Boletín de Política Informática, No. 6, pp. 50–63 (2003)
16. PMI: A Guide to the Project Management of Body (2013)
17. Spector, A., Gifford, D.: A computer science perspective of bridge design (2010)
18. Alba, C.: Prediction and classification of the risk level in systems projects. Oviedo University (2008)
19. Fairley, R.: Managing and Leading Software Projects. Wiley - IEEE Computer Society Press, New York (2009)
20. Al-Mudimigh, A., Zairi, M., Al-Mashari, M.: ERP software implementation: an integrative framework. Eur. J. Inf. Syst. **10**(4), 216–226 (2001)
21. Tomala, C., Jazmin, S.: Quality metrics information systems, application quality certification of a company in the oil and gas sector. Guayaquil-ESPOL (2009)
22. Garzás, J., Irrazabal, E.: Basic metrics and analysis of open source tools to measure maintainability. REICIS Rev. Esp. Innovación **6**(3), 55–65 (2010)
23. Scalone, F.: Comparative study of models and software quality standards (2006)
24. Kendall, K., Kendall, J.: Analysis and Design of Systems. Ed. Pearson Education, Essex (2005)
25. Lafuente, G. J.: Conceptual framework for the definition and exploitation of quality metrics. Malaga University, Malaga (2014)