Issues on Evaluating Free/Open Source ERP Systems

Rogerio Atem de Carvalho
Federal Center for Technological Education of Campos (CEFET Campos)
R. Dr. Siqueira 273, CEP 28030-130, Campos/RJ, Brazil,
ratem@cefetcampos.br,
http://www.cefetcampos.br/ensino/dppg/nucleos_pesquisa/npssd/ratem.html

Abstract. Free/Open Source ERP Systems represent an area which is increasingly gaining acceptance due to many reasons. However, there is still a lack of specific evaluation methods for adoption and very little academic research is reported in this area. Therefore, this paper aims to discuss important issues on Free/Open Source ERP Systems evaluation and, as of consequence, to propose an evaluation method that takes into account not only aspects related to the software itself, but also to the possibility of joining the software development effort.

1 Introduction

Free/Open Source ERP Systems (FOS-ERP) are increasingly gaining acceptance for many reasons. One reason is direct cost, since they impose no licensing costs in general. Other reason is the perception of the fact that if customization is inevitable, why not adopt a solution that exposes its code to the client company that can freely adapt the system to its needs. Adapting is a crucial point to ERPs, given that, according to [1], despite being called *software*, enterprise systems in general "have nothing to do with 'shrink-wrapped', 'off-the-shelf' items that can be used instantaneously". This remark reinforces the freedom of manipulating the code by itself in FOS-ERP: if the vendor changes its contract terms, the client company is not locked in to a particular solution supplier [2]. Additionally, can two competing companies derive a strategic differential using the same ERP? Although this problem can also happen with FOS-ERP, it seems to be bigger for P-ERP, since, due to limited (or even none) access to source code, adaptations are also very limited, restricting real differentiation.

Please use the following format when citing this chapter:

Atem de Carvalho, R., 2006, in International Federation for Information Processing, Volume 205, Research and Practical Issues of Enterprise Information Systems, eds. Tjoa, A.M., Xu, L., Chaudhry, S., (Boston:Springer), pp.667-675.

If, for both kinds of ERP, the fact that integration among processes can by itself be a source of competitive advantage - even if only to strengthen the generic competitive position defined by the company by overcoming some specific localized trade-offs [3], this can be extrapolated to the possibility of changing source code to drive an even better advantage.

In summary, two basic advantages attract companies to FOS-ERP: lower costs and access to application code. But this is not enough to satisfy a company's needs. There are strategic questions that must be evaluated to decide which FOS-ERP to adopt or even if a P-ERP is more suitable to the specific case of the company. This paper is going to focus on the case in which a company wants to compare different FOS-ERP for adoption, not addressing any specific issue related to comparison between P-ERP and FOS-ERP.

1.1 Motivation

Evaluating FOS-ERP is harder than P-ERP. Besides the usual functionality, technology, and financial aspects, there are also aspects related specifically to Free/Open Source Systems (FOSS) that are vital for FOS-ERP.

Evaluating FOSS projects is a well-commented matter in industry. A search conducted at Google.com for "evaluating open source" brings about 21,000 hits as for January 2006; many of them are related to evaluation methods. Narrowing the search down to "evaluating open source erp", the result goes down to only 6 distinct hits. Of those six results, one has nothing to do with the subject, one was a dead link and the other four pointed to a site that offers a tool for evaluating ERPs, open source or not. Doing the same search in IEEE and ACM digital libraries returns only approximated results, none of them dealing with this specific issue.

Although search engines aren't guarantee to find the desired information, only one keyword combination was used, and the method utilized was not scientific, the discrepancy in results and the difficulty to directly find out information about evaluating FOS-ERP show how this subject is poorly analyzed, by both the industry and the academics. In the academic arena, this seems to be another situation where technology has outstripped the conceptual hawsers: according to Kim and Boldyreff [4], by September 2005, only one paper about Open Source ERP [5] "has been published in the whole of ACM and IEEE Computer Society journals and proceedings, whereas more numerous articles have been published in non-academic industrial trade magazines."

Additionally, in the realm of P-ERP, there are methods, tools and sites for evaluation and comparison purposes. In spite of some portals like TechnologyEvaluation.com include some FOS-ERP on their evaluation database; there isn't a specialized structure for FOS-ERP evaluation in any case. One can find industry papers that offer some punctual evaluations but nothing like a specific method or tool.

Instead of a punctual evaluation of this highly dynamic kind of EIS, this paper aims to propose a basic framework for FOS-ERP evaluation, without the intension of

embracing all the matters related to this matter, but with the declared intention of bringing more attention to FOS-ERP aspects and evaluation.

2 Related Work

Among the various methods proposed for evaluating FOSS, three of them were selected to be briefly described here, because they are generic enough and have as a premise the fact that normally organizations have their own way of evaluating software, influenced by their technical expertise, organizational culture, specific needs, and experience of the decision makers. Although other methods found also present these characteristics, the three here summarized seems to be more suitable to be adapted for FOS-ERP evaluation.

De Carvalho, Costa, and Xu [6] present a method based on risk to evaluate FOSS, which focuses specifically on particularities of FOSS that are consequences of the fact of their code is open. This method does not concentrates itself on evaluating functional and non-functional requirements, instead takes into account two core factors related to the FOSS development: the strategic positioning of the adopter, and the chances of the FOSS "survive" in the future. The first factor will determine if the adopter will simply use the FOSS, collaborate somehow in its development or make profit from selling it [7]:

- Consumer: a passive role where the adopter will just use the software as it is, with no intention or capability of modifying or distributing the codes.
- Prosumer: an active role where the adopter will report bugs, submit feature requests, post messages to lists. A more capable Prosumer will also provide bug fixes, patches, and new features.
- Profitor: a passive role where the adopter will not participate in the development process but simply will use the software as a source of profits.
- Partner: an active role where the adopter will actively participate in the whole open source development process for the purpose of earning profits.

The second factor is measured through development conditions, survivability, and support level. This method gives a special attention to community engagement as an indicator of the chances of the project evolves in the long range.

Wheeler [8] describes a generic method for evaluating FOSS based on four steps, joined under the IRCA acronym: identify, review, compare, and analyze. *Identify* means to find out candidate systems for evaluation, possibly by the use of Generally Recognize as Mature/Safe (GRAM/GRAS) lists of programs. *Review* is based on obtaining existing evaluations of the candidate systems, which can be comprised by consulting portals like Freshmeat.net for instance. *Compare* is maybe the hardest part of the process, since it aims to compare functionality, Total Cost of Ownership (TCO), market share, support, maintenance/longevity, reliability, performance, scalability, usability, security, flexibility, interoperability, and legal issues. *Analyze*

finally implements an in-depth analysis of the remaining contenders, concentrating in functionality and security.

Golden [9] has introduced the Open Source Maturity Model (OSMM), which is a three-phase process for selecting and assessing open source software. OSMM assesses the maturity level of a FOSS through key product elements: the software itself, support, documentation, training, product integration, and professional services. In Phase 1 the adopter organization evaluates each product element in four steps: define requirements, locate resources, assess element maturity, and assign a score to the element. In Phase 2 weightings are applied to each product element, according to its perceived criticality for the adopter. Finally, in Phase 3 an overall product maturity score is calculated. OSMM supplies a set of minimum scores to determine if the FOSS is suitable to the adopter's needs.

3 ERP Evaluation and Strategic Issues

Evaluating P-ERP in general involves comparing alternatives under the light of functionality, TCO, and technology criteria; for FOS-ERP the same criteria plus the ones related specifically to FOSS must be taken into account. However, it is not a question of simply aggregating all criteria into one evaluation method, because ERP systems are not ordinary software, like web servers or content management systems. According to Caulliraux and colleagues [3], "their high implementation costs can in itself establish the option for an ERP as strategic – it is a major commitment of money, and thus with long range implications even if only from a financial point of view." Caulliraux also states "ERPs are also important not only as a tangible asset – high-value hardware and software, but also as a catalyst through their implementation in the formation of intangible assets and the company's self-knowledge." At this point, it is clear that even if a FOS-ERP implementation assumes a smaller financial importance than a P-ERP one, in terms of a company's self-knowledge it can assume a much greater importance, since it holds not only a inventory of records and procedures, but also how those records and procedures are realized in a technological fashion – through source code.

In other words, a FOS-ERP can have a smaller financial impact but a much bigger knowledge and innovation impact. Although P-ERP are also adaptable through Application Program Interfaces (API) or even dedicated programming languages, the access to the whole source code in FOS-ERP, not only an API, will drive much better exploration of the ERP's capabilities, thus allowing a better implementation of differentiated solutions.

From this standpoint, the strategic positioning of an adopter in relation to an FOS-ERP seems to be of greatest importance, driving the necessity of a differentiated evaluation process, given the possibility of deriving innovation and competitive advantage from the source code. Although innovation and advantage could be obtained if the adopter (i) develops a whole solution by itself, or (ii) simply acquire a

P-ERP, it would go back to (i) the high costs of this kind of solution, or (ii) the shortcomings related to P-ERP stated before.

3.1 Strategic Positioning

For the specific case of an FOS-ERP adopter, the Profitor and Partner roles mentioned before are not applicable to the focus of this paper, and the other roles can be reinterpreted as follows; assuming that always will have adaptations to be done to the system:

- Consumer: a passive role where the adopter will simply buy the adapting service from a software house, without any direct collaboration to the development process.
- Prosumer: an active role where the adopter will assume the adapting process, reporting bugs, submitting feature requests, and posting messages to lists. Depending on the inclination to share information, a more capable Prosumer can also provide bug fixes, patches, and new features or even modules.

Clearly, the adopter strategic positioning has a great impact on the way it sees the FOS-ERP. Different kinds of adopters may assess an identical project feature quite differently. Insightfully, some weaknesses (such as lack of documentation) revealed in the evaluation may encourage the adopter to become a Prosumer and contribute to the project, impelling it, and consequently turning into a positive return in the form of new features created by other prosumers or partners of the project.

It is important to know if the organization wants to shift from a user point of view to a developer role. Also it is necessary to evaluate if the adopter's development team has the correct skills to develop new features. So, it is necessary to verify first the software's technical features or functionalities that the adopter needs. If the software has the necessary features, and they are fully compliant to the adopter needs, there is no problem and development conditions don't need to be checked. On the other hand, if there are missing features or the existing ones are partially compliant, the adopter must choose one of the following actions: reject the OSS, check if there are plans to develop the missing features (and if the plans fit to adopter's time constraints), or join the development effort.

If the adopter decides to become a developer, project management issues, like time and costs, must be addressed. Although having other community members involved in the project is good – lowering costs, this can mean managing a project where many variables lay outside the adopter organization, thus the level of managerial control is substantially lower. In general, FOSS communities carry highly informal relationships among its members. Also, it is necessary to follow a development cycle that is influenced by many members of the community, which sometimes requires a massive coordination effort [10], depending on the complexity of the task.

In some cases the adopter can assume a dual strategic positioning. As an example, there is the case of ERP5 user Coramy [11], a leading European apparel industry. Coramy acts both as a consumer, using the services of Nexedi, ERP5's creator for

customization purposes in general, and as a Prosumer, collaborating with the community with testing, discussion, and patching.

4 PIRCS: A Method for FOS-ERP Evaluation

The proposed method mixes ideas from the previously summarized methods: from [6] it uses the concept of strategic positioning, from [8] it uses some of its steps, and finally from [9], it uses the central idea of explicitly scoring each candidate. It can be understood as a *meta-method*, given that the method is composed by a series of steps or procedures that should be adapted for specific purposes, according to the adopter's software evaluation culture and specific needs.

The method is divided into five steps, identified by the acronym PIRCS: prepare (the evaluation), identify (alternatives), rate (alternatives' attributes), compare (alternatives), and select (the best alternative). The following topics detail each of these steps.

4.1 Prepare

This step is responsible for initiating the evaluation process, through the definition of a series of parameters that will have influence in the whole process. It is comprised by:

- Requirements definition: is related to functional and non-functional requirements for the ERP. This task defines a set of attributes that must be rated.
- Strategic positioning: will define if the adopter is firmly decided to be only a consumer or if it's willing to become a Prosumer in these cases where development costs and lead-time are acceptable and FOS-ERP technology is known enough to the development task. A dual positioning can be assumed, in cases where the adopter decides for developing by itself some features and contracting a third part for developing other features, because it skills are somehow limited or time and cost constraints drive this kind of positioning.
- Definition of extra attributes, used to identify the survivability level of the FOS-ERP: the adopter must analyze figures like number of posts on lists, of feature requests, of bugs and others, to identify the size and level of community involvement. In [6] a series of attributes for measuring community activity levels are listed and explained.
- Limits set up: will define lower and/or upper bounds for each attribute in the decision process. Outside these bounds, the candidate reaches a cut level, meaning that it won't be considered any more for evaluation. For attributes related to performance for instance, lower bounds are determined; for others, related to cost and time for example, upper bounds are used.
- Measurement method definition: since the method is based on scoring attributes, a measurement method must be defined. This method has to take into account

qualitative and quantitative value scales. For instance, robustness is normally qualitative, whereas cost and time are quantitative.

4.2 Identify

The identification step identifies the candidate FOS-ERP, or, in other words, the decision alternatives. To accomplish this, the process described by [8] can be used. At this point, GRAM/GRAS lists, web searches, and visiting portals specialized on ERP can be used.

4.3 Rate

Rating means measuring the utility of each attribute for each alternative. In other words, associating grades for each of the attributes defined at Prepare step. Rating will be divided in two parts:

- Rating the requirements for the system, meaning to evaluate each functional and non-functional requirement according to an established scale. For the functional requirements, it is interesting to note that the granularity of the analysis must be defined, for example, in some cases the adopter may want to analyze a module as a whole - like Payroll module, or specific functionalities of a given business process – as, for instance, automatic alerts of low inventory levels in a Inventory Control Module. If the granularity is high, the adopter must define a method to aggregate rates for specific functionalities towards a final grade for a macro-functionality or module.
- Rating the attributes that define the survivability level of the system. This kind of evaluation is also important since an active community around a FOS-ERP (or a FOSS in general) can mean fewer expenses on support and bigger probabilities of the project follow up in the future. It is also important to note that technological and methodological aspects of the FOS-ERP can define its survivability independent of the community. For instance, the use of obsolete technologies can determine a problem in the future for the system. On the opposite side, too innovative technologies can expose the project to a series of risks, including discontinuity, non-fulfillment of requirements, and fewer people working on the project.

Additional techniques can be used to accomplish rating. Weiss [12] focus on measuring the success of a FOSS using web search engines. This very intuitive technique can be used to infer how much a FOS-ERP is known – and as a consequence, used - thus giving an indirect measure of its survivability. In [6] is presented a method focused on risk level evaluation in accordance to community engagement and helpfulness, maturity, level of acceptance, technology, project hosting support, documentation, commercial support levels, release frequency, and lists activity. Both techniques can be used since the prepare level, to define an extra set of requirements for the FOS-ERP.

At the end of this step, a final rate for each attribute of each candidate (alternative) is created.

4.4 Compare

This step will take all the rates from all the candidates and establish a ranking or a pair to pair comparison, depending on the technique used to compare alternatives. To accomplish this, simple techniques like weighted averages - used in OSMM, or more sophisticated ones, like Multi Criteria Decision Making (MCDM) methods, can be used.

4.5 Select

Selection is split up from comparison when the method used to compare alternatives does establish a final ranking from where the best candidate is obtained. This situation occurs, for instance when pair to pair comparisons are used, or when the adopter decided not to mix in the same scale qualitative and quantitative attributes.

In these situations, the adopter must define a specific logic for selecting the best alternative. At this point, it is important to note that the combination of upper and lower bounds (interpreted as cut levels) and performance of an FOS-ERP can determine the adopter's strategic positioning or even the elimination of the alternative from the evaluation process. For instance, let's say that for the Production Planning and Control module the adopter defined a lower and an upper bound, respectively X and Y, where:

- Bellow X the candidate is eliminated from the process.
- Between X and Y and considering acceptable costs, the adopter will improve the module functionality. If the costs are prohibitive, the candidate is eliminated.
- Above Y the candidate will be accepted as is (for this attribute).

5 Conclusions

The aim of this paper is twofold: discuss some important issues on evaluating FOS-ERP, and also propose a basic method that can be improved for specific purposes and needs, or even generalized for evaluation of FOSS in general.

An important conclusion is that, given the strategic nature of ERPs, evaluating a FOS-ERP must include the strategic positioning of the adopter towards the development process of the candidate systems: in some cases joining the development effort means extra strategic differential for the adopter.

Additionally, the information available on the Internet about FOS-ERP, is not only inexpensive, but also is generally non-biased, given the fact that the system's source code is totally exposed for analysis, allowing a deeper evaluation of its functionalities, security, technology, and flexibility.

An interesting outcome of this discussion and possible future work is the creation of a MCDM model that would cover Rate, Compare and Select steps of the proposed method, and the application of this model in a real situation.

References

1. Dreiling, H. Klaus, M. Rosemann, and B. Wyssusek, Open Source Enterprise Systems: Towards a Viable Alternative, 38th Annual Hawaii International Conference on System Sciences, Hawaii (2005).
2. Kooch, Open-Source ERP Gains Users (February 01, 2004); http://www.cio.com/archive/020104/tl_open.html
3. H. M. Caulliraux, A. Proença, and C. A. S. Prado, ERP Systems from a Strategic Perspective, Sixth International Conference on Industrial Engineering and Operations Management, Niteroi, Brazil (2000).
4. H. Kim and C. Boldyreff, Open Source ERP for SMEs, Third International Conference on Manufacturing Research, Cranfield, U.K. (2005).
5. J. Smets-Solanes and R. A. De Carvalho, ERP5: A Next-Generation, Open-Source ERP Architecture, *IEEE IT Professional* **5**(4), 38–44 (2003).
6. R. A. De Carvalho, H. G. Costa, and N. Xu, A Risk Based Method for Open Source Evaluation, Federal Center for Technological Education of Campos, Research Management Technical Report 01/2003, 2003.
7. N. Xu, An Exploratory Study of Open Source Software Based on Public Archives, Master Thesis, John Molson School of Business, Concordia University, Montreal, Canada, 2003.
8. Wheleer, How to Evaluate Open Source Software / Free Software (OSS/FS) Programs (August 26, 2005); http://www.dwheeler.com/oss_fs_eval.html
9. B. Golden, *Succeeding with Open Source* (Addison-Wesley Professional, 2004).
10. M-D. Wu and Y-D. Lin, Open Source Software Development: An Overview, *Computer* **34** (6), 33-38 (2001).
11. J. Smets-Solanes, and R. A. De Carvalho, An Abstract Model for An Open Source ERP System: The ERP5 Proposal, Eighth International Conference on Industrial Engineering and Operations Management, Curitiba, Brazil (2002).
12. Weiss, Measuring Success of Open Source Projects Using Web Search Engines, First International Conference on Open Source Systems, Genova, Italy (2005).