Management of Requirements in ERP development: A Comparison between Proprietary and Open Source ERP

Johansson, Björn; Carvalho, Rogerio Atem

Published in:
[Host publication title missing]

2009

Citation for published version (APA):
Johansson, B., & Carvalho, R. A. (2009). Management of Requirements in ERP development: A Comparison between Proprietary and Open Source ERP. In *[Host publication title missing]* (pp. 1605-1609)

Total number of authors:
2

Management of Requirements in ERP development: A Comparison between Proprietary and Open Source ERP

Björn Johansson
Center for Applied ICT
Copenhagen Business School
DK-2000 Frederiksberg
+45 3815 2421

bj.caict@cbs.dk

Rogério Atem de Carvalho
Federal Center for Technological
Education (CEFET Campos)
R. Dr. Siqueira, 273, Campos/RJ,
CEP 28030-130, Brazil
55-22-2726-2904

ratem@cefetcampos.br

ABSTRACT
Identification and specification of business requirements are extremely important when development of Enterprise Resource Planning systems (ERPs) take place. It can be stated that this is still a problematic area not well researched and, as a result from that, it does not exist much guidance on how to deal with requirements. In this paper we discuss if existing problems of requirements management are the same or if they differ according to the type of development: closed source (proprietary) or open source ERP. The reason is that it is possible that these two approaches can promote each other in how to improve the first phase in ERP development. From the discussion about similarities and differences between the two approaches it is suggested further research in this area that could end up in some more practical guidelines on how to do the requirements definition so that the finally developed ERPs better support adopters' needs.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *elicitation methods.*

General Terms
Enterprise Information Systems.

Keywords
Enterprise Resource Planning, Free/Open Source Software.

1. INTRODUCTION
Development of Enterprise Resource Planning (ERPs) systems is an endeavor that has a high level of complexity, and a great deal of this complexity is directly related to requirements management. The complexity resulting in several problems related to requirements management comes from the fact that developers often not are experts of the domain the developed systems are supposed to support. Shehab et al., [1], Esteves and Pastor [2] and Botta-Genoulaz et al. [3] show that there exists a great extent of

ERP research. When reviewing these reports over ERP research the impression that a major part of the research is on implementation of ERP systems is gained. It also shows that the main problem presented is the misfit between ERP functionality and business requirements. Soh et al. [4] describe this as a common problem when adopting software package. The problem of "misfit" means that there is a gap between functionality offered by the package and functionality required from the adopting organization. Askenäs and Westelius [5] describe this in the following way: "Many people feel that the current ERP system has taken (or been given) a role that hinders or does not support the business processes to the extent desire". Another way of describing this is as said by Bill Swanton, vice president at AMR research, that only 35 per cent of the organizations are satisfied with the ERP they use at the moment, and he says the reason for the dissatisfaction is that the software does not map well with the business goals [6].

Both practical experience and research described in the literature emphasizes on the problem about misfits between business requirements and ERP functionality. According to Soh et al. [4], the misfits could be related to the following three areas: architecture of the specific software, IT-architecture and business architecture. However, it can be argued that this misfit is a result over deficiency in the requirement management process, in which business analysts and developers are supposed to agree on what functionality that the ERP should support. From this the following question, which we discuss in the paper, is formulated: What similarities and differences are there between the two types of ERP development – closed source (proprietary) or open source – regarding the requirement management process? The reason for why this is of interest is that it can be that it is possible that these two approaches can promote each other in how to improve the first phase in future ERP development. However, to be able to say something about this there is a need to first discuss what the potential similarities and differences between these two approaches could be.

The following sections will briefly describe ERP business requirements, how management of requirements are made in Free/Open Source ERP (FOS-ERP) and proprietary ERP (P-ERP) respectively, what problems there are within these two development approaches, and finally a comparative discussion between these approaches, followed by some conclusions and suggestions for future research directions.

2. ERP BUSINESS REQUIREMENTS

The above describes that a major problem in ERP development is the misfit between required functionality and functionality offered, which also could be described as the distance between what the end-users want to have support for in the business processes they works within and what the ERP *de facto* gives support for. There are definitely a lot of different reasons for why this is the case and this paper will not deal with all of them due to space constraints. But, it can be stated that one important factor are difficulties of transferring business requirements from identification to specification and further on into implementation. To have some input on this it is first important to stipulate what requirements and especially ERP business requirements are. Jackson [7] describes requirements as descriptions of the application domain and the problems to be solved. He makes a distinction between requirements and specifications and states that specifications are descriptions of the interface between the developed system and the application domain. This is in line with the statement that a requirement specification should form a bridge between requirements engineering and software engineering [8]. From this it could be said that it is clear what requirements are, but, that is definitely not the case, and there are several reasons for that. Earlier research and practice have tried to classify and categorize requirements. Many of these classification schemes distinguish between functional and non-functional requirements [9]. For example, the IEEE standard for the software requirements specification [10] distinguishes fourteen types of requirements, divided into functional requirements and thirteen types of non-functional requirements. A similar distinction is made by Robertsons [11], who identifies seventeen different types of requirements divided into product constraints, functional requirements and non-functional requirements. From this it can be suggested that requirements could be seen as either: a function, capability, or property of a proposed system; and/or, the statement of such a function, capability, or property [9] and/or as described by Jackson [7] as descriptions of the application domain and the problems to be solved there. This last description emphasize on what and not how. This is to some extent in conflict with the description from Zave and Jackson [12] that state: *there was a time when the epigram "requirements say what the system will do and not how it will do it" summarized all of requirements engineering. That time is long past.* What Zave and Jackson state could be interpreted as that a change in what requirements should describe has occurred and that requirement specification now also to some extent focus on how the developed system will execute the wanted requirement. By stating this it can be suggested that the scope of what requirements are have broaden a lot, however, it also shows the importance of having requirements described in a way so that they can be implemented in the way they need to be implemented.

Another angle of why there are some basic problems when defining, what requirements are, comes from the fact that many stakeholders are involved [9] and that these stakeholders have different perspectives on what requirements are [13]. For instance, if a CIO says that the basic requirement on a new ERP is the need to support the organizations business processes, the developers then probably have a hard time to understand exactly how to do that. It is also so that the requirement specification as such is seen as an end product and not the evolutionary documentation of a process that it should be. This then results in the fact that,

especially for ERPs, that the specification does not reflect the implemented solution. The reason for this is that since the environment changes all the time and the ERP has to be adjusted to the environment it works in, the adopted ERP solution drifts away and relatively soon after its adoption it have changed.

It is shown from experience that the analysis and documentation of business and software requirements by means of models are essential for ERP development, which thereby makes it necessary to use proper techniques and tools [14]. Odeh and Kamm [14] state that for instance the Unified Modeling Language (UML) software specification techniques are not suitable for translation of business models into software models. One reason for this is that the modeling tools does not take organizational background characterized by shifting interests, interpretations, and power relations into account to the extent that is necessary. It can be stated that this is especially important when developing ERPs since these are systems that are supposed to support the entire organization and also support organizations interaction with stakeholders outside the organization.

The critique Odeh and Kamm stipulate can be compared to the three levels of software requirements that Wiegers suggests. Wiegers [13] states that software requirements include three distinct levels – business, user, and functional requirements. The interesting level, in this context, is the business requirements. According to Wiegers [13] business requirements are representations of high-level objectives that the organization or the customer requests from the system. In order to fully understand what these requirements are about it is needed to clarify what ERPs are. The definition we adopt of ERPs is the one given by Daneva and Wieringa [15]. They state that ERPs are packaged software solutions with the key function of coordinating the work conducted in an organization. The coordination means that ERPs should be seen as the vehicles that modern organizations use to achieve business connectivity in which everyone knows what everyone is doing in the organization. This definition of ERPs gives a relatively clear view over what ERP business requirements are, but, it also shows that identifying and clearly specifying these are a difficult task. Monnerat et al., [16] label the task of describing requirement at this level as systems requirements definition. How this is done in the two different approaches, P-ERP and FOS-ERP, is the focus of the next section.

3. REQUIREMENT MANAGEMENT IN THE REQUIREMENT DEFINITION PHASE

3.1 Free and Open Source ERP

The obvious difference between P-ERP and FOS-ERP is that the second type exposes its source code. At first glance the access to source code would be a direct influence to the requirements elicitation process. However, ERP requirements are elicited as the candidate system were a "black box", meaning that there is no need to inspect source code, to answer if the system is capable of comprising to the adopter's business requirements.

Nevertheless, source code openness brings the possibility of the adopter in taking a more active position in relation to the system customization tasks. Carvalho [17] proposes a FOS-ERP evaluation method named PIRCS (Prepare the evaluation, Identify alternatives, Rate alternative' attributes, Compare alternatives, and Select the best alternative), which, among many other things,

takes into account the adopter's strategic positioning, as summarized below:

-Consumer: a passive role where the adopter will simply buy the customization service from a FOS-ERP vendor, without any direct collaboration into the development process.

-Prosumer: an active role where the adopter will assume entirely or partially the customization process, by reporting bugs, submitting feature requests, and posting messages to development lists. Depending on the inclination to share information, a more capable prosumer can also provide bug fixes, patches, and new features or (even) entire new modules.

Going further, Carvalho affirms that "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."

Therefore, it is important to know if the organization wants to shift from a user point of view to a developer role – with all the consequences of this shift, such as checking the technological knowledge necessary for the customization, allocating personnel for system development and maintenance, and coordinating with the project community.

Following the PIRCS method, in parallel with the requirements elicitation tasks, there should be a strategic positioning phase that will define if the adopter is firmly decided to be only a consumer or if the adopting organization is willing to become a prosumer - in the cases where development costs and lead-times are acceptable and FOS-ERP technology is known enough by the adopting organization so that it could contribute in the development task. Dual positioning is also possible, by customizing by itself some features and contracting for others. Additionally, the survivability level of the FOS-ERP project must be addressed, to avoid projects with high risk of not accompanying the necessary evolution pace expected for an enterprise system, as detailed in the paper by Carvalho, Costa and Xu [18].

3.2 Proprietary ERP

Johansson [19] describes from an investigation about challenges in ERP development, done at a major P-ERP vendor , that a major challenge is related to the requirements gathering process. What executives at the vendor state is that the time from feature identification to its implementation is too long. The reason suggested for this is that they see that there is a lack of or a need for an improved process for requirements management. Another problematic area suggested is the risk of implementing non-relevant requirements, which is also related to a deficiency in the requirement management process.

Therefore, it can be said that the requirements elicitation process in P-ERP is problematic, and the problems come from what could be described as the ERP development paradox. The paradox means that ERPs aims at being a standard software package at the same time as it aims at being a unique resource in its usage in different organizations. The problem also comes from the fact that ERPs, and especially P-ERPs, in most cases have a development chain that consists of at least three parties – the developer, the distributing partners, and the user organization – and it can be stated that all three of these contribute to the further development on the adopted ERP. However, it can also be said that the development chain restricts the feedback loop related to further development. One reason for this restriction is to keep the competitive advantage brought by the ERP usage in secret [20]. This restriction means that adopters believe that they gain competitive advantage by having a unique ERP implemented and therefore they do not want to give the developer the possibility of implementing the unique feature in the standard package. This also means that developers of P-ERP needs to develop a system that can be made unique by the individual using organizations at the same time as they make sure that the system is possible to upgrade without overwriting the unique features implemented. However, it is not only the requirements that are unique for individual adopters that are problematic when it comes to requirement elicitation, but practically all requirements, since many stakeholders has to agree on what the ERP should support. The P-ERP development chain as such creates definitely some problems when it comes to communication and how to describe requirements so that these are understandable by the stakeholders in the development chain. The question is then if the FOS-ERP and the P-ERP development models could promote each other and thereby improve the first phase in ERP development in general.

4. COMPARISON BETWEEN P-ERP AND FOS-ERP

The next step in the paper is to take a more in-depth look into similarities and differences between the two options related to the requirement definition point of view. So far it can be said that the two options are very similar and does not show any huge difference in how the development is done and what problems are faced. It seems that P-ERP and FOS-ERP developers experience the same problems when it comes to requirements elicitation. To further discuss that we therefore suggest an additional categorization of P-ERP development namely: Internally developed P-ERPs and commercial prepackage ERPs. The FOS-ERPs were already categorized into consumer open source or prosumer open source ERP development. The question is then if these four ERP development situations differ in, for instance, closeness to the end-users.

In Table 1 we suggest an initial categorization of the four options related to contribution or involvement by the end-user organization in the specific ERP development. The X in the table should be understood as an indication of to what extent the end-user organization is involved in the actual development of the ERP system adopted. Since it is not possible to do a clear cut between contributions in different options we also use (X) in the table, which means that, for instance, internally developed P-ERPs has a higher grade of end-user organizations involvement than FOS-ERPs developed by prosumers. In fact, it is important to state that prosumers in the FOS-ERP case can only develop parts of the system, while using others "as-is".

When it comes to the case of P-ERP commercial prepackage ERP case we suggest that the major development is made without involvement of the end-user organization, but that some development also takes part with partial involvement from the

end-user organization, such as add-ons developed by partners for specific adopters that later on are implemented in the core product by the vendor. From this it can be argued that categorization of involvement in the table describes end-users organizations level of control over the development. This means that it can be suggested that end-user organizations level of control over development is highest in internally developed ERPs and then decreases to some extent with FOS-ERP prosumer development and further on with FOS-ERP consumer development and then finally the commercial prepackage ERPs shows the lowest level of end-user control over development.

Table 1. Contribution (involvement) by the end-user organization in the ERP development

	Totally	Partly	Not at all
P-ERP Internally developed	X		
P-ERP Commercial Prepackage ERPs		(X)	X
FOS-ERP consumer		X	(X)
FOS-ERP prosumer	(X)	X	

As described already a major challenge identified in ERP development is how to find the "correct" requirement as well as the time from identification to implementation. From the table it can be suggested that if including also internally developed ERPs in the discussion these are comparable with the FOS-ERP prosumer case development situation. These two situations then suggest that closeness to the end-user is of importance at least if one also state that these systems show better alignment between wanted functionality and delivered functionality. However, experience shows that even systems developed internally in an organization have problems related to the misfit description described by Soh et al., [4]. One reason for this is that also in this case the problem of being an expert in both the business processes and information systems development is something that are related to different stakeholders, which means that internally development also includes considerable amount of communication between different stakeholders. However, when excluding the situation of internally developed ERP - in most cases this is not seen as ERP – it can be argued that FOS-ERP development are made closer to the end-user, but the closeness as such does not solve the misfit problem.

To some extent it could be suggested that open source ERPs experience smaller discrepancies than proprietary ERPs do. One reason for this is that open source development takes place closer to the using organization. However, it can also be suggested that FOS-ERP development have a risk of becoming a one organization task, creating a development vacuum. If that happens it could be asked if the open source developer have enough knowledge about what happens in the environment that influences or should influence development of the specific ERP.

Boulanger [21] asks how a disparate loose-knit group of developers can produce software that has comparable quality with proprietary software for free. He describes the feedback loop as

one difference between development of proprietary software and open source software making it possible. The most common approach in a proprietary software development process is what could be described as a waterfall model. This means that the development more or less follow a clear structure and uses a set of five well-defined phases. Boulanger [21] presents the following five phases as a generic structure for proprietary software projects: The requirements phase, the system and software design phase, the implementation and unit-testing phase, the integration and system-testing phase, and the support and maintenance phase. He says that this structure of course is an iterative process, but that open source development phases are more intertwined in each other. Suggesting that open source development are more intertwined stipulates that the more intertwined development process makes it possible to decrease the time from identifying a specific requirement to when it is implemented, since the feedback is more direct. However, it must be remembered that ERP are different, since a given needed modification on the system can be related to a single adopter, so that the community may not get so interested on contributing to it. It seems that generic features, those related to well known business practices, are highly to be supported by the community, while other, directly related to the business culture of a single adopter, probably will be supported only by this adopter and/or its consultants. This assumption is reinforced by the fact that most of times internally developed competencies that can drive competitive differential are not revealed, at least immediately to the community.

5. CONCLUSIONS AND FURTHER RESEARCH

It can be concluded that in the FOS-ERP development case the distance between the user and the developer is smaller than it is in the P-ERP case. But, important to state is that this is only the case when the user is a prosumer. In the consumer case the distance between developer and user are probably the same and the same problem with misfit between needed functionality and delivered functionality probably shows up.

A major problem identified in the ERP requirements elicitation phase is the problem of describing requirements so that the developer understands the actual needs from the users; and at the same time as the users can evaluate and make clear that the developer develops what is wanted. It can be stated that this, which could be described as a communication problem – the developer and the user do not speak the same language - is a problem in both P-ERP as well as FOS-ERP development.

The main conclusion is that there are no big differences between the two types – closed source or open source - of ERP development related to requirements management in the requirement definition phase. The case is instead that the two types show the same problem during requirements elicitation when developing the basic product. Instead the difference could maybe be found if categorizing the development situation into the four cases: open source consumer, prosumer open source, closed source prepackage ERP, and closed source internally developed ERP. One interesting research question is then to compare open source consumer and closed source prepackage ERP with open source prosumer and closed source internally developed ERP. By selecting these four types of development situations it would

probably be possible to gain some interesting knowledge about ERP development related to requirements management that later on could act as guidelines in how to improve the requirement management process.

Another question is to evaluate how the development process influences on the misfit problem. It seems that this problem can be minimized by customization processes based on Agile Methods assumptions, such as shorter iterations (instead of long waterfall phases) and making the user part of the development team. The communication problem, related to the difference between the languages used by users and developers, can be minimized by the use of a Ubiquitous Language [22], a language structured around the domain model and used by all development team members to connect all the activities of the team with the software. In that way, using the same language to communicate to a closer user can reduce misfit.

6. REFERENCES

[1] Shehab, E.M., Sharp, M.W., Supramaniam, L., and Spedding, T.A., Enterprise resource planning: An integrative review. *Business Process Management Journal*, **10**,4 (2004), 359-386.

[2] Esteves, J. and Pastor, J., Enterprise Resource Planning Systems Research: An Annotated Bibliography. *Communications of AIS*, **7**,8 (2001), 1-51.

[3] Botta-Genoulaz, V., Millet, P.A., and Grabot, B., A survey on the recent research literature on ERP systems. *Computers in Industry*, **56**,6 (2005), 510-522.

[4] Soh, C., Kien, S.S., and Tay-Yap, J., Cultural fits and misfits: Is ERP a universal solution? *Communications of the ACM*, **43**,4 (2000), 47-51.

[5] Askenäs, L. and Westelius, A. *Five roles of an information system: a social constructionist approach to analyzing the use of ERP systems*. in *twenty first international conference on Information systems*. 2000. Brisbane, Queensland, Australia: Association for Information Systems.

[6] Sleeper, S.Z. *AMR analysts discuss role-based ERP interfaces - the user-friendly enterprise*. 2004 [cited 12-04-2007]; Available from: http://www.sapdesignguild.org/editions/edition8/print_amr.asp.

[7] Jackson, M., *Software requirements & Specifications: a lexicon of practice, principles and prejudices*. 1995, London: ACM Press.

[8] Jackson, M., The meaning of requirements. *Annals of Software Engineering*, **3**,1 (1997), 5-21.

[9] Power, N.M., *A grounded theory of requirements documentation in the practice of software development*, in *School* 2002, Dublin City University: Dublin. 223.

[10] IEEE, *IEEE STD 830-1998: IEEE recommended practice for software requirements specifications*. 1998, Los Alamitos, CA: IEEE Computer Society Press

[11] Robertson, S. and Robertson, J., *Mastering the requirements process*. 1999, Harlow, UK: Addison Wesley.

[12] Zave, P. and Jackson, M., Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, **6**,1 (1997), 1-30.

[13] Wiegers, K.E., *Software Requirements*. Vol. 2nd ed. 2003, Redmond, Washington: Microsoft Press.

[14] Odeh, M. and Kamm, R., Bridging the gap between business models and system models. *Information and Software Technology*, **45**,15 (2003), 1053-1060.

[15] Daneva, M. and Wieringa, R., Cost estimation for cross-organizational ERP projects: research perspectives. *Software Quality Journal*, (2008).

[16] Monnerat, R.M., de Carvalho, R.A., and de Campos, R., *Enterprise systems modeling: the ERP5 development process*, in *Proceedings of the 2008 ACM symposium on Applied computing*. 2008, ACM: Fortaleza, Ceara, Brazil.

[17] Carvalho, R.A., *Issues on evaluating Free/open source ERP sysyems*, in *Research and Practical Issues of Enterprise Informations Systems* 2007, Springer Verlag New York Inc: New York. 667-676.

[18] Carvalho, R.A., Costa, H.G., and Xu, N., *A Risk Based Method for Open Source Software Adoption Evaluation*, in *XIII CLAIO - Congreso Latino-Iberoamericano de Investigación Operativa*. 2006: Montevideo, Uruguay.

[19] Johansson, B., *Pain points challenges for future enterprise resource planning (ERP) systems*, in *3gERP workshop*. 2007: Copenhagen.

[20] Johansson, B., *Developing a "better" ERP system: The risk of loosing competitive advantage*, in *Research and practical issues of enterprise information systems II*, L. Xu, A.M. Tjoa, and S. Chaudbry, Editors. 2007, IFIP Springer.

[21] Boulanger, A., Open-source versus proprietary software: Is one more reliable and secure than the other? *IBM Systems Journal*, **44**,2 (2005), 239-248.

[22] Evans, E. Domain-Driven Design Tackling Complexity in the Heart of Software. Addison-Wesley, 2003.

Björn Johansson holds a Doctoral and a Licentiate degree in Information Systems Development and a Bachelor degree in Business Informatics. Currently he works at the Center for Applied ICT at Copenhagen Business School, within the 3gERP project (http://www.3gERP.org). He is a member of the IFIP Working Group on Diffusion, Adoption and Implementation of Information and Communication Technologies and the research networks: VITS and KiO

Rogerio Atem de Carvalho holds a Doctoral and a Master degree in Production Engineering and a Bachelor degree in Computer Science. He was awarded with the IFIP Distinguished Academic Leadership Award, for his studies on the field of FOS-ERP, in Vienna, Austria in 2006. He is a consultant for the ERP5 FOS-ERP (http://www.erp5.com), and project manager of Enterprise Content Management solutions for governmental and private sectors since 2000. He is Chair of the Brazilian Chapter of the IFIP Working Group on Enterprise Information Systems and Founder Member of the IEEE SMC Society Technical Committee on Enterprise Information Systems. Also, he is an Associated Editor for Enterprise Information Systems journal.