Improving Software Design Decisions towards Enhanced Return of Investment

Pedro Valente [1], David Aveiro [2] and Nuno Nunes [2]
[1]*University of Madeira (UMA), Colégio dos Jesuítas, Funchal, Portugal*
[2]*Madeira Interactive Techonologies Institute (M-ITI), Campus Universitário da Penteada, Funchal, Portugal*

Keywords: Software Engineering, Enterprise Engineering, Business Process Improvement, Software Process Improvement, Software Metrics, Financial Metrics.

Abstract: One outstanding issue in modern information systems development is the Return of Investment (ROI) of supporting Business Processes (BPs) through in-house development and/or integration of business modules from component-based development. Software solutions to solve this problem will usually be based in internal software development processes where the inadequate decision, for e.g. the wrong software framework, may lead to losses that will range from minor adjustment budgets to financially catastrophic situations. Here we propose to use information from the analysis of BPs metrics to enhance decisions related to software design, based on software development effort estimation for the new enhancement, and the related ROI as a path to consistently raise project success. This paper frames a Software Process Improvement (SPI), Enterprise Engineering (EE) and Software Engineering (SE) based-solution to enhance ROI following better design decisions, and provides in-depth relevant considerations regarding our future work.

1 INTRODUCTION

Interactive Information Systems (IISs) allow organizations to collect data in order to produce, with more or less effort, all the information users need to complete their tasks according to everyday needs and strategic projects. However, when Business Processes (BPs) are enhanced, if the adequate structural information support is not provided for the new and changed tasks, users might be led to "jump" from system to system (including telephone) in order to keep track of situations they are responsible for or related to. Hence, lack of BP control may lead to pressure cascades that result in discontentment and diminished performance, instead of the envisioned augmented efficiency and/or effectiveness.

It is commonly accepted that dealing with BP Improvement (BPI) is a meaningful stage for Software Engineering (SE) as numerous Software Process Improvement (SPI) initiatives are continuously elaborated to deal with this problem. Nevertheless, only few inherently consider cost-benefit analysis (Unterkalmsteiner et al., 2011), and human factors are still not sufficiently considered weakening the strength of the crucial alignment between managers, SE practitioners and users in order to consistently reach project success (Ferreira and Wazlawick, 2011).

In order to achieve BPI, there is usually the need to promote new software projects that aim for: (i) corrective maintenance, in order to develop new interface features or enhanced data integration; (ii) improvement maintenance to e.g. provide new front-ends or new systems (Abreu, 2013). These projects will be carried out using internal or outsourced know-how and internal or acquired tools, as the commercial pressure for the option for cloud frameworks (Fox et al., 2013, Larry Coyne, 2014) and BP management tools (Skalle and Hann, 2013) may lead to a software development structural transition. We argue however, that this pressure and its related impact, may eventually disrupt the natural way of building new features according to the user's real needs using already existing resources, which we believe that can bring increased efficiency out of minor investment in an acceptable and adequate Return of Investment (ROI) according to the organization's dimension, if not avoiding financially catastrophic situations (Morgenshtern, 2007), when compared to solutions that comprise framework changes.

According to recent studies in effort estimation, decisions to enhance or change software framework

DOI: 10.5220/0005383803880394
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 388-394
ISBN: 978-989-758-097-0

are believe to add up to 75% of development effort, whereas preventive maintenance can decrease development effort in 25%, considering the same features (Alves et al., 2013). Moreover, there is the common knowledge that a high percentage of software projects still fail despite the continuous SPI efforts and the availability of new information technology (IT) tools (Unterkalmsteiner, 2011, The Standish Group, 2013). Another reason found in the literature for project failure, identifies imposition, regardless of practitioners and users will, as a reason to unsuccessful results, and in the opposite direction, points out addressing human factors as a solution to reach success (Ferreira and Wazlawick, 2011).

Besides SE, BPI is also an issue in Enterprise Engineering (EE), as the need for change to enhance efficiency has long been recognized as a challenge (Dietz, 2006). Although organizations are seen in a completely different way from SE, as the focus is on the organization's architecture, i.e. the coordination and the production acts combined in organizational transactions (as considered in the DEMO approach), the BPI problem is precisely the same, as a SE project is expected to solve the enhancement need. The EE perspective inherently focuses much more on human aspects, as the software system is seen as an added value to the enterprise, and not as an axiom of the problem, as organizations, earlier, already had their information systems (mostly in paper). Therefore, we believe that a cooperation between SE and EE is needed in order to achieve a higher level of BPI, since the more holistic perspective of EE about the organization can be and added value to SE.

Both SE and EE recognize the necessity to elicit requirements in such a way that they are useful for BPI, and Use Cases seem to be the obvious solution, as it is a concept already recognized in both domains (Constantine and Lockwood, 2000, Dietz, 2003). In fact, similar approaches that design BPs with Use Cases already exist, such as Process Use Cases (Valente and Sampaio, 2007) from SE and the Systematic Approach (Cruz et al., 2014) from EE.

We wish to take the already existing communion between SE and EE further, as our approach is based on the idea that if system design is carried out based on the analysis of the proper metrics, more adequate solutions will be produced, avoiding waste and promoting efficiency and effectiveness, to overcome third-party solutions in terms of ROI, and give managers and practitioners a better decision support as they will be able to better envision cost-benefit implications, and decide whether to use more or less resources according to the organization's capabilities, and therefore achieve the meaningful task, for both domains, of, increasing project success rates.

This paper is structured in the following way. Section 2 considers the relevant techniques for the development of our solution from both EE and SE domains, Section 3 presents our solution approach, Section 4 presents literature's related work, Section 5 in-depths relevant considerations regarding our future work, and Section 6 presents our conclusions.

2 BASE TECHNIQUES

The following techniques are the theoretical basis of our proposal, which goal is to allow organizations to continuously be aware of the financial impact of their decisions, especially regarding BPI, the problem that we address in this paper.

Regarding the elaboration of our contribution for SPI, we focused on existing methods, from EE for organization management and design, and from SE for software development methods, as the basis to understand the organization i.e. the problem that leads to BPI needs, and the process that should follow i.e. the software solution for that BPI. Concerning the specificities of our SPI proposal, we then consider two software design use case-driven solutions that elicit the benefits of the new enhancement, and then consider two distinct approaches for software effort estimation that can be distinctly used to calculate the software cost. We then focus on the financial perspective, in terms of ROI to consider the previous cost-benefit variables.

2.1 SPI from the EE Perspective

EE looks at organizations as systems, and from this perspective, financial aspects are an axiom of the approach. We now consider two distinct approaches that define strategic and structural techniques to be used in order to design the organization.

DEMO (Dietz, 2006) defines the theoretical concepts on how an organization should be architected in order to achieve success. The teleological perspective explains the purpose of its existence, while the ontological perspective looks at the BP existence as a need to organize activities for production. Although DEMO focuses more on the organization's design than on BPI, relevant and complementary work, as the GOD approach (Aveiro, 2010), elicits the need to control BP metrics in order to identify exceptions and to act adequately in order to re-establish order. This can be done either by simply handling exceptions, i.e. situations in which minor adjustments are needed, or by engineering processes in order to regain control over dysfunctions caused by unexpected exceptions.

The Business Model Canvas (BMC) (Osterwalder and Pigneur, 2010), originally coming from the service design domain, focuses on the enhancement of the organization based on a strong relation with its external environment, and looks at organizations in terms of systems that should produce revenue as a result of the organization's social context. BMC claims that the path for success is based on the relation between partners and internal resources, and sees value propositions, i.e. how BP should be enhanced regarding customer's needs, as the process to enhance the organization. These value propositions should be elaborated based on existing or new key partners, activities and resources, in order to enhance customer relationships, product channels and customer segments. A cost structure and a related revenue stream should be elaborated in order to evaluate the cost-benefit of the new proposition.

2.2 SPI from the SE Perspective

We now consider a SPI approach from 2011 that focuses on the need to manage the change in a holistic perspective within the organization.

The Change Management Practices (Ferreira and Wazlawick, 2011), focuses on the need to control change within an organization as the way to achieve project success and avoid ROI losses. The approach emphasizes the need to establish a change vision and communicating that vision to reach the needed alignment between managers, practitioners and users, in order to effectively achieve the change demands. The authors propose practices to be carried out in three phases: (i) unfreezing, (ii) moving and (iii) refreezing; in which they elicit the need to form a coalition with the authority to implement change, by communicating that need, the associated strategy, and reach for short term wins, until the change is fully accomplished. Relevant SPI problems are highlighted as for example the resistance to change when there is not enough evidence of the benefits for the stakeholders, eventually resulting in lack of commitment in a cumbersome process. The study is complemented with a questionnaire, in which it is concluded that there is a real need to focus on people when there is a need for change.

2.3 Software Design

At the moment when enough knowledge is reached about the system to be developed, there is the need to design it in detail. We consider two distinct, use case-driven approaches for software design, which can provide the needed artefacts, not only for software implementation specification, but also for software effort estimation.

Goals (Valente, 2009) is a comprehensive use case driven, BP design, requirements elicitation, analysis, and system design methodology, that promotes traceability between all components of a system, in a method that can be applied in order to produce the minimal artefacts needed to ensure software quality for each project, as follows:

- Process Use Cases model – BP design;
- Use cases design – analysis of the user tasks and their sequence;
- Domain model – information structure design;
- User interface design - detailed design of the user screens;
- System Architecture – dependency relations between relevant system components.

Goals promotes the simplification of the system before it is built, and therefore, it may lead to the construction of more light and adequate systems.

Usage-centered design (UCD) (Constantine and Lockwood, 2000) is a methodology for user interface and interaction design with special concerns in usability. It is based on the definition of several concepts related to human-computer interaction (HCI) such as: (essential) use cases, actors and roles. UCD defines four models for requirements definition:

- Activity map – representation of the activities to the design problem;
- Activity profiles – purpose and performance of each activity;
- Participation map – participants and artefacts involved in the activity;
- Activity-task map – tasks (use cases) and actions extraction.

Following the definition of the tasks, the user interface should the designed based on the canonical abstract prototypes (CAPs), as enough system complexity knowledge already exists. UCD suits our approach, as we believe that specific user interface design decisions have relevant implications on the software development effort.

2.4 Software Effort Estimation

Accurate software effort estimation is a cornerstone of our approach as the calculation of the ROI of a SPI initiative will only be possible if the cost is measured. In a recent study from Alexei Botchkarev it is possible to observe that the mean magnitude of relative error, i.e. the deviation will rarely be close to zero (0), as, out of 47 projects from 15 studies, 75% of the deviations ranged from 20 to 60%, including the most relevant methods from the SE industry, namely, Use Case Points (UCP), COCOMO and Function Point Analysis (FPA) (Botchkarev, 2014).

We now consider two software effort estimation methods.

Interactive Use Case Points (iUCP) (Alves et al., 2013) is a development of the original UCP method (Karner, 1993), that introduced a human perspective to the original method, whereas, the user perspective provided by HCI methods adds complexity to the original actor weight, to better reflect the difficulty of interactive user interfaces development. The original use case weight was left unchanged. The authors also defend that requirements stability, the option for a new software development framework, and the project type (maintenance or new development) largely impact software development effort. In the iUCP proposal, these aspects can add up to 125% of the effort to the project, which contrasts with the original UCP method where the same aspects could only influence the final calculation for no more than 5%.

Function Point Analysis (FPA) (Albrecht, 1979) is an effort estimation method widely used in the software industry, which works at the detailed level of each function in a system, namely:

- External inputs - components responsible for introducing changes in system's internal data;
- External outputs - the ways system's internal data can be presented;
- External inquiries - methods for reading system's data without modifying it;
- External interface files - responsible for exchanging data with other systems;
- Internal logical files - files that are being used by the system itself.

The final effort is calculated considering the existing environmental and technical factors.

2.5 Return of Investment

Accurate return of the investment (ROI) as a result of detailed system design is the main objective of our approach. It is a simple calculation from the economical perspective, nevertheless, its usage is not a common practice in SE as stated in a comprehensive study from Michael Unterkalmsteiner as out of 148 analysed SPI initiatives, only 8 contemplated cost-benefit analysis to measure of the impact of the new enhancement (Unterkalmsteiner et al., 2011).

We believe that the contribution to SE of Solingen (Solingen, 2004), specific to SPI in terms of ROI calculation, to be a sufficient. In this approach estimated development effort and other human resources are considered, and, benefits are divided in direct benefits, in terms saved effort, and indirect benefits, in which case, predictions should be carried out by specific area experts.

The formula for the calculation of the ROI is:

$$\text{ROI} = (\text{Benefit} - \text{Cost}) / \text{Cost} \qquad (1)$$

3 OUR APPROACH

We frame our approach in the following perspectives.

3.1 Business Process Enhancement

In order to be aware of the cost of each BP occurrence, the enhancement of a BP should always be done using the following metrics:

- Average cost for each user intervention;
- Average cost per occurrence.

And in order to assure the quality of a BP, the following metric should also be calculated:

- Number of failures per number of occurrences.

In order to refine quality levels, the following objective should be defined:

- Acceptable number of failures per number of occurrences.

The BP should then be redesigned, using, e.g. the DEMO notation (Dietz, 2006), or a simple notation like Process Use Cases (Valente and Sampaio, 2007).

3.2 Software Development Process

Once the BP is designed, a software development process should follow in order to produce the following artefacts:

- Use cases model;
- Detailed user interface design;
- And, optionally, Comprehensive software architecture.

Once the system is designed, distinct development scenarios, should be considered in order to proceed with effort estimation.

3.3 Effort Estimation

Once the needed information is available, the effort estimation should follow to calculate the cost, and the project duration considering the number of available human resources.

3.4 Return of Investment Calculation

Based on the cost information, it should be possible to calculate the ROI as it will be possible to predict the benefits based on the following metrics:

- New average cost for each user intervention;
- New average cost per occurrence (including other operational costs)

Indirect benefits should also be possible to calculate based on the following metric (considering an improvement at this level):

- Number of failures per number of occurrences.

The simple ROI calculation would then be possible using formula (1), where both Benefit and Cost variables should assume the following values:

- Cost = Cost of the software development effort.
- Benefit = Direct + Indirect Benefits.

If the ROI is not acceptable, then, design changes should be considered in order to fine-tune the effort according to the organization's reality.

4 RELATED WORK

Although there are several approaches from both domains of SE and EE that focus on the value of software development within organizations, we have identified none that inspects ROI as the result of the specific design of the solution. The approaches that we select as related work, are however, significant work under progress in this direction, i.e. the need to understand the value that is directly derived from new software development efforts. We now present two approaches, one from SE and one from EE which reflect the concerns related to revenue from software development projects.

GQM+ Strategies for Business Value Analysis (Mandić, 2010) is a value based software engineering (SBVE) derived method that has it focus on the usage of goal question metrics (GQM) approach combined with business goals to understand business value, i.e. the importance of each project within the organization. The method then focus on the ROI of each project, and also considers risk factors for each case.

Value-oriented Solution Development Process (Pombinho, 2013) is a value-aware DEMO-based system development process, which defines a value model based on the investors needs in order to identify solutions (scenarios) for the problem, and select the most profitable implementation, and also define that the new enhancement should be evaluated in order to make sure that the objectives are attained. The following formula, which reflects specific operationalization costs, where OPEX stands for operational expenditure, is considered as an equivalent method for ROI calculation reflecting the dividends from a new investment:

$$\text{dividend} = \text{business OPEX} - \text{IT OPEX} - \text{investment} \quad (2)$$

5 DISCUSSION

Our approach motivation is based on the idea that relevant advances can be achieved in SPI if closer attention is taken to detail, especially on the user perspective, as we believe that following BP enhancement, the investment effort can be fine-tuned based on its relation to design, as it will be possible to better predict costs and benefits using a ROI analysis.

The main problem regarding our approach will be reaching the needed accuracy for effort estimation. According to the state of the art to this specific topic, we understand that there is still a long way until consistent accuracy, around 10% deviation, is reached, and that this will only be possible if a stable development environment exists. Nevertheless, by the identification of design and consequent software effort development patterns, it might be possible to reach, if not the accuracy, at least the awareness that design decisions directly affect the cost of a project, as e.g., the development of a new framework, or not, to solve specific SPI problem, will always be more costly than a simple integration enhancement between two existing systems (Alves et al., 2013). Hence, we believe that, if this complete analysis is made, previous to any software development, major mistakes will be less probable, since the organization itself, will have a more consistent SPI method.

We believe that another crucial aspect of our approach will be the development of patterns that will allow both managers and SE practitioners to first easily identify BP enhancement needs, and then identify a patterned design solution for that enhancement, and following that, in reduced time, understand how much they can benefit from a new BP enhancement in terms of ROI and also other added qualitative values for the organization, as we believe that if detailed attention is paid to user performance, discontentment will decrease resulting in intangible benefits for the organization.

Concerning the usage of our method for both SE end EE domains, one relevant problem might be related to finding the adequate use case definition, something that has been under discussion for several years, mainly in SE (Cox, 2002). We consider however that essential use cases (Constantine and Lockwood, 2000) support both domain concerns in terms of task completion and definition.

Our main objective will be reaching evidence that our approach is valid to produce reliable results within a stable software environment, for a precision around 10% for effort estimation and around 20% for ROI accuracy. In order to achieve this objective, we believe that our method must be applied in several organizations and that discussion with researchers can accelerate the elaboration process.

6 CONCLUSIONS

Our approach aims at enhancing decisions related to the software development, and specifically for BPI. We believe that a systematic process that leads to an implementation logic derived from the "big picture" of the organization down to the detailed design of software and including all relevant implications in terms of ROI, brings us to a new paradigm to give a new "life" to those who struggle to efficiently manage and further enhance their software patrimony and consequently attain their full organization potential. We also believe that this step towards complete software development and management awareness, once achieved, might be the missing step to increase software project success worldwide.

The more relevant aspects of software development within an organization, which are reflected in our approach, in our opinion, are:

- BP enhancement design;
- Software design;
- Effort Estimation; and,
- ROI analysis.

As an extra outcome of a more consistent and coherent software development method, other outstanding problems can be diminished, such as:

- The difficulty to highlight change benefits;
- The difficulty to align managers, practitioners and users wills into change.

Finally, we believe, that gathering knowledge from the complementary domains of SE an EE, can be an added value for both domains, as both science domains are closely related by means of the need to enhance the productivity and welfare of our organizations.

REFERENCES

Alves, R., Valente, P., Nunes, N. 2013. Improving software effort estimation with human-centric models: a comparison of UCP and iUCP accuracy. In *Proc. of 5th ACM SIGCHI Symp. on EICS '13*. ACM. pp. 287-296.

Abreu, J., Ventura, P., Fernandes, S., Zacarias, M. 2013. Business Processes Improvement on Maintenance Management: A Case Study. In *Procedia Technology*. 9: pp. 320–330.

Albrecht, A. 1979. Measuring Application Development Productivity. In *Proc. of IBM Application Development Symp*. IBM Press. pp. 83–92.

Aveiro, D. 2010. GOD and Control (sub) organizations: a DEMO-based approach for continuous real-time management of organizational change caused by exceptions. *PhD Thesis*. UTL, Lisboa, Portugal.

Botchkarev, A. 2014. Estimating the Accuracy of the Return on Investment (ROI) Performance Evaluations.

Constantine, L., Lockwood, L. 2000. Structure and Style in Use Cases for User Interface Design. In *Object Modeling and User Interface Design*. Addison Wesley.

Cox, K. 2002. Heuristics for use case descriptions. *PhD Thesis*. Bournemouth University, Poole, England.

Dietz, J. 2003. Deriving Use Cases from Business Process Models. In *Conceptual Modeling - ER 2003, Lecture Notes in Computer Science.* Springer Berlin. Heidelberg. pp. 131–143.

Dietz, J. 2006. The Deep Structure of Business Processes. In *Communications of the ACM*. 49(5): pp. 58-64.

Cruz, E., Machado, R., Santos, M. 2014. From Business Process Models to Use Case Models: A Systematic Approach. In *Advances in Enterprise Engineering VIII, Lecture Notes in Business Information Processing*. Springer International Publishing. pp. 167–181.

Ferreira, M., Wazlawick, R. 2011. Software Process Improvement: An organizational change that need to be managed and motivated. In *World Academy of Science, Engineering and Technology*. 50: pp. 269-277.

Karner, G. 1993. Resource Estimation for Objectory Projects. *Objectory Systems*.

Mandić, V., Basili, V., Oivo, M., Harjumaa, L., Markkula, J. 2010. Utilizing GQM+ Strategies for an Organization-Wide Earned Value Analysis. In *Proceedings of the 36th EUROMICRO Conference*. 1-3: pp. 255–258.

Morgenshtern, O., Raz, T., Dvir, D. 2007. Factors Affecting Duration and Effort Estimation Errors in Software Development Projects. In *Information and Software Technology*. 49(8): pp. 827–837.

Osterwalder, A., and Pigneur, Y. 2010. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. 1st edition. John Wiley and Sons. ISBN: 978-0470876411.

Pombinho, J., Aveiro, D., Tribolet, J. 2013. Value-Oriented Solution Development Process: Uncovering the Rationale behind Organization Components. In *Advances in EE VII*. Springer Berlin. pp. 1-16.

Fox, S., Johnson, C., Follette, D. 2013. *Beginning SharePoint 2013 Development*. Wiley. ISBN: 978-1118654873.

Solingen, R. 2004. Measuring the ROI of Software Process Improvement. *IEEE Software*. 21(3): pp. 32–38.

Skalle and Hann. 2013. Applying Lean, Six Sigma, BPM, and SOA to Drive Business Results. In *TechRepublic*. IBM RedBooks.

The Standish Group, 2013. Chaos Report 2013.

Unterkalmsteiner, M. et al. 2012. Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review. In *IEEE Transactions on Software Engineering*. 38(2): pp. 398–424.

Valente, P., Sampaio, P. 2007. Process Use Cases: Use Cases Identification. In *Proc. of ICEIS'07*. pp. 301–307.

Valente, P. 2009. *Goals Software Construction Process: Goal-Oriented Software Development*. *VDM Verlag Dr. Müller*. ISBN: 978-3639212426.