Risk profile assessment for software development teams: a first step towards best practices adoption

Liziane Santos Soares, José Luis Braga, André L. de Castro Leal,
Carlos H. O. Silva, Juliana P. Campos

Departamento de Informática Universidade Federal de Viçosa
Av. P.H. Rolfs s/n Campus da UFV - 36570-000 - Viçosa - MG
lssoares@inf.ufrgs.br, zeluis@dpi.ufv.br, andrecastr@gmail.com,
chos@dpi.ufv.br, jupcampos@gmail.com

Abstract. Software quality is straightly related to the quality of the process and best practices employed in software production. However, in most cases process tailoring or best practices choice is not done using scientific criteria. Choices should rely on the quantification of key parameters related to the software development environment, leading to less exposure to risks. This work proposes a procedure including a questionnaire application followed by a systematic interpretation that allows the measurement and comparison of some of those parameters. Afterwards, those parameters can be plotted in a polar chart, thus originating a risk profile that could be used as guidance to less risky choices. The procedure was tested over a sample of small software development organizations within an ongoing project.

Keywords: Software quality, risk profile, software process

1 Introduction

The main goal of Software Engineering is developing quality software. This is a strong demand these days; because software is ubiquitous [1] and is embedded in systems and devices we use everyday [2].

Software development process is essential to product quality [3]. The choice of software development best practices, which include process tailoring, is still partly an art, and does not rely entirely on parameters that establish the relationship between organization context and best practices. There are many choices available, and a bad choice could lead to underachievements. In general, choices are made based on criteria taken from common sense and from the experience of members of the development team, and they do not always rely on a technical approach based on more scientific basis and fundamentals. There is the need of a more organized and well founded procedure to determine software development best practices that will adhere more closely to software development team necessities.

This paper presents a procedure for measuring some key parameters related to software development context. Afterwards, those parameters can be used for a risk-

based choice of software development best practices in a development team. The determination of those parameters is achieved by their definition, extraction and analysis in a systematic way, allowing a more accurate quantification and subsequently the construction of a profile that forms a basis for more adequate choices.

The remainder of this paper is structured as follows. In section 2, key concepts are presented. Section 3 presents methodological aspects related to the systematic procedure proposed in this paper. Section 4 presents the application of the procedure to real world cases, leading to profile determination. Section 5 presents concluding remarks.

2 Background

2.1 Process Based Software Quality

Software quality can be defined as the conformity of the software product to functional and performance requirements documented in an explicit manner [2], in addition to other implicit and explicit characteristics defined by the standard [4]: Functionality, Reliability, Usability, Efficiency, Maintainability and Portability.

However, it is important to devote attention not only to the product quality, but also to the software development process, tools and best practices adoption and people management. It is necessary that process quality and product quality go together. Software development process is essential to product quality [3].

A software development process consists of a set of steps, partially ordered, related with sets of artifacts, resources, organizational structures and constraints, aiming at software production [5]. Process application favors software quality management and software project management [6].

Information about process applied to previous projects can be collected and used to monitor and to improve the process to be applied in future projects. The improvement of the process allows the improvement of the software produced [7].

2.2 Hybrid Methods

For the purposes of our research, a hybrid method is one that combines characteristics of Plan-Driven Methods (PDM) with characteristics of Agile Methods (AM), aiming at more adherence of software development processes to a larger set of software development projects than the AM and PDM separately [8].

Plan-Driven Methods are focused on discipline. They use a systematic approach through which software is built from the requirements to the final code passing through a series of well defined and documented phases [8]. They are more adequate to large software that involve high risks for the final user, have stable requirements and are developed by larger teams. Examples of PDM are: Rational Unified Process (RUP) [9], IBM Global Services Method [10] and DMR Macroscope [11].

Agile Methods are lightweight processes that employ short iterative cycles. Users are involved to establish, prioritize, and verify requirements, and client satisfaction is

prioritized. These methods rely on tacit knowledge instead of a wide documentation. They aim at suitability to changes in requirements as a way to reduce the cost of changes during the project [8]. They value the support to new requirements and emphasize the importance of communication [12]. Examples of AM are: eXtreme Programming (XP) [13], Crystal [14] and Scrum [15].

Hybrid methods can be a good choice for tailoring a process to a specific context of software development [8]. And they can also be enriched by adopting software development best practices [9]. The blending of AM and PDM requires a method that considers the problem features, the development environment and team profile. In [8], Barry Boehm proposes a risk based approach to blend AM and PDM methods based on the analysis of certain development context features.

2.3 Goal Question Metric (GQM)

The GQM approach is based on a procedure for the definition of metrics [16]. It is a systematic approach to create and integrate goals with software process models, products and quality perspectives, based on the needs of a project or organization [16]. It considers that for an organization to accomplish measurements in an efficient way, it is necessary to specify each goal to be reached by measurements results. The goals are refined into several questions that define each one of those measurement objectives. Finally, each question is refined into metrics which should be collected to answer the formulated question. GQM approach is a measurement framework that has three levels [17]:

- *Conceptual level (Goal):* an objective is defined for an object that can be a product, a process or a resource;
- *Operational level (Question):* a set of questions is associated with each objective;
- *Quantitative level (Metric):* a set of data is associated with each question in order to answer it in a quantitative way.

The GQM approach can be used as a standard for the definition of measurement frameworks [18] and has been applied by several organizations like Philips, Siemens, NASA [19].

3 Methodological Aspects

3.1 Critical Factors in Risk Profile Assessment

According to [8], five critical risk factors should be considered in determining the relative suitability of AM and PDM to a particular project situation:

- *Size* - related to the number of people in a team;
- *Criticality* - related to the impact in the problem environment due to errors in the software. It ranges from risk to human lives through financial or other less important risks;

- *Dynamism* - related to the percentage of requirements change per month;
- *Personnel* - related to the mix (percentage) of team members skills, defined as Cockburn levels [14];
- *Culture* - related to the percentage of the team that feels comfortable under chaos versus the percentage that prefers order, regarding requirements stability.

The five critical risk factors can be represented graphically as shown in Figure 1, where each factor is represented by an axis in a polar chart. From the analysis of development context features and problem context features, it is possible to obtain a shape that resembles a pentagon. This shape represents the region that should be considered to determine the proportion between AM and PDM risk-based best practices selection and adoption. The closer the pentagon-like curve is to the center of the polar graph means that AM related best practices can be adopted with less risks. The farther the pentagon-like curve is from the center of the polar graph means that risks can be minimized by adopting PDM related best practices.

According to Figure 1, if requirements change frequently an agile practice should be sought, because this kind of practice deals better with evolving or changing requirements than PDM. But for sure, this dimension should not be taken alone. It should be analyzed in the context of the other dimensions, since the interpretation of the results in one dimension can influence the interpretation of other dimensions.

3.2 A Systematic Procedure for Risk Profile Assessment

This paper enhances the original proposal presented in [8] using GQM to tailor a procedure. The procedure is used to determine values for the five factors. The values determine a graph region that helps managers make less risky decisions about best practices and process adoption.

Through the GQM application, questions and metrics were established and questionnaires were developed and tested. These questionnaires can be applied to other development teams in other contexts. Moreover, a systematic interpretation for the collected data is proposed as another part of the procedure.

Thus the questionnaire application followed by the systematic interpretation of the collected data compose the procedure and can guide the choice of best practices and process tailoring for a specific team.

3.2.1 Application of GQM

In this work each one of the five critical factors, shown in the polar graph of Figure 1, is a goal or objective in GQM application. To reach these objectives, a series of questions is formulated and from those questions, several metrics are established. Those metrics are used as a guide to measure each one of the five factors.

For example, Table 1 shows the application of the GQM approach to measure the critical factor Dynamism as proposed in [17]. The goal is to find out which are the necessary metrics for estimating the frequency of requirements changes caused by changes in the problem context.

Table 1. The GQM Approach to Extract Information Related to the Factor Dynamism

Purpose:	Estimate.
Issue:	Frequency of software requirements changes caused by the problem context changes.
Object:	Software requirements.
Viewpoint:	Process manager.

Question:	Which is the main reason for requirements changes in the system?
Metric(s):	- % of requirements changes as a consequence of the problem context changes. - % of requirements changes as a consequence of the visualization of delivered parts of system (prototyping). - % of requirements changes as a consequence of the incorrect specification of requirements.

Question:	Which is the frequency of requirements changes in cases that these changes occur as a consequence of problem context changes?
Metric(s):	- % of requirements changes / month (from the total established until now).

3.2.2 Questionnaire Design

The questionnaires are directed to the team manager and to each of the team members, and aim to obtain data about the team as a whole, about its members and about the development context. The data collected is related to general team features and individual features of each member. Two questionnaires are defined:

- *Managerial questionnaire*: aims to extract information related to the team as a whole. It should be answered by a professional who understands the team and the development routine, like a manager or other professional with an equivalent level of accountability;
- *Individual questionnaire*: aims to extract information related to each of the members of the team. It should be answered by each member of the team, independently of its role.

A validation pre-test was conducted in our search for small errors, ill formulated questions and ambiguous or insufficient options for a specific question. This pre-test was carried out by applying the questionnaires to four software development teams that do not belong to the research sample. One of them is a large managerial software development team with many experienced members. The other three ones were small academic teams, devoted to the development of software to support academic and technological activities in a university environment. This validation allowed questionnaire error correction and misconception adjustments, leading to a finer adequacy tuning.

3.2.3 Interpretations Associated with the Questionnaire

In some cases, the data were obtained by more than one question. This is due to the fact that information is not easily obtained, and in some cases it can come from different sources and different questions. As an example related to the factor

dynamism, if the question “which is the percentage of requirements change occurred by month originated by changes in the problem context?” is asked to a manager, he certainly would have problems answering it. Instead of asking that, other simpler questions such as those that appear in the Appendix, can be proposed, allowing the indirect gathering of the same information using more than one question. The information so extracted should then be made compatible to be safely used.

In the Appendix, questions 3, 4 and 5 from the managerial questionnaire are shown. The answers to those questions are not directly in accordance with the scale of the chart of Figure 1. An association of the answers should be made to generate a value that represents the % of changes of requirements / month, as a point on the Dynamism axis inside the graphic. This association is accomplished according to the following calculation:

*%ofRequirementsChanges = AnswerForQuestion3a / 100 * AnswerForQuestion5 * Frequency*

The equation above produces a value for the % of requirements change due to alterations occurred in the problem context. That value is obtained from the answers to questions 3a and 5, and from the Frequency value. The Frequency value is obtained from the answer to question 4 that defines the frequency of requirements changes during development. The final result should be presented as a function of the % of changes/month. For example, if the answer to question 4 is *daily*, then the value for Frequency should be 30. The values for Frequency should be: [0 – *never*], [30 – *daily*], [4,29 – *weekly*], [1,00 – *monthly*], [0,50 – *once in two months*], [0,33 – *once in three months*], [0,17 – *once in six months*] and [0,08 – *yearly*].

Projects developed by the team can be categorized under the perspective of dynamism, and classified as dynamic or not dynamic. Development contexts which presents % of change/month larger or equal to 10% are considered dynamic; the other cases are considered not dynamic. In [20], Boehm rates a requirements changes/month between 10% to 30% as “quick” (or dynamic for our purposes).

It is necessary to establish the association between the other questions and the respective critical factors. Because of size constraints, they are not shown and discussed here, and can be found in [21].

4 Case Study

Seven case studies were carried out in small software development organizations that are part of a local production arrangement around information technologies in Viçosa-MG-Brazil. This is the sample we are working with as part of a partner project that is now under development, partially financed by SEBRAE – *Serviço Brasiliero de Apoio às Micro e Pequenas Empresas* (a Brazilian organization that offers technical and financial support to small business) and managed by our research team at *Departamento de Informática – Universidade Federal de Viçosa*. Partial results shortly described in this section were taken from the first step of the project, in which we focused on parameter gathering by applying the questionnaires to each of the organizations. We present some partial anonymous results of the research based on a first analysis of the data collected. No further statistical analysis were done yet.

The organizations are small ones, with small teams (less than 10 people) and the software developed by them present low risks to the public. Only one of the organizations is a little bit larger and can be considered medium sized, but it does not to deserve special attention right now. The organizations were labeled as DPI.xx, where xx ranges from 01 through 07, and the parameters obtained for them were organized in a table not presented here because of space limitations.

For each one of the organizations, a risk profile was built, taking the five factors into consideration. The profiles will be subsequently used on the project next step, to build a first recommendation of best practices that can be safely adopted by each organization (in this context, safely means low risk). So, we decided to show the profiles of two organizations, DPI.03 (continuous line) and DPI.04 (dashed line), represented by their polar charts in Figure 1. This graph shows a risk profile range for the seven organizations, ranging from DPI.03 that is more PDM prone through DPI.04, that is more AM prone. The profiles for the other organizations do lie somewhere between these two profiles, and agility or discipline best practices adoption should be established based on them.

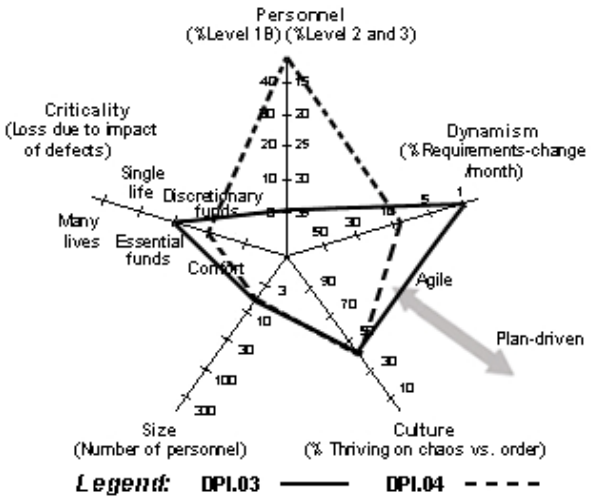

Fig. 1. Risk profile range for the seven organizations

In this phase of the project development, the goal is finding a common profile to help shaping a mix of AM and PDM that should be used by the whole sample. This should make learning curves less deep, leading to quality gains in less time and thus increasing return on investment. The graph presented in Figure 2 shows a "mean" profile that can be taken as common to all organizations in our sample. This profile analysis suggests that the adoption of PDM best practices and process tailoring will be less risky for the whole sample. The profile establishes that the projects developed have a low requirements change rate, low criticality, teams are small and have a mix of level 1B and level 2 and 3 around 30-20 meaning that teams are composed by 30% less skilled programmers (1B) and 20% more skilled ones (levels 2 or 3) according to Table 1, and culture suggests plan-driven solutions.

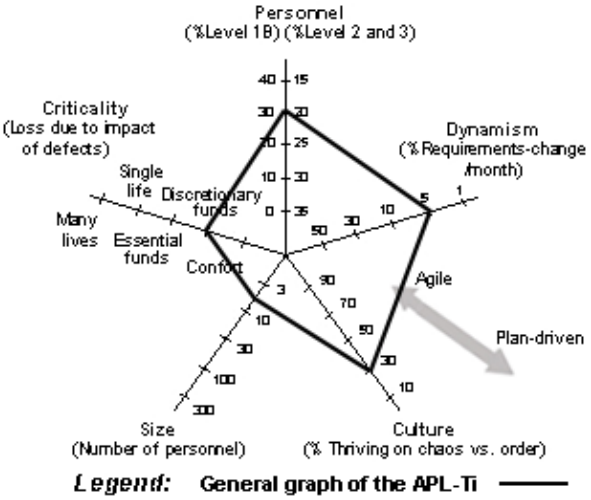

Fig. 2. Polar graph representing the profile of the whole sample.

5 Concluding Remarks

The goal of our research project is helping small software development organizations to enhance quality in their work. Unfortunately, solutions like CMMI [22] are more driven to large organizations that have resources available to submit themselves to evaluations and adopting expensive processes, tools and artifacts. Even models as MPS.BR [23], a model for process quality focused on the Brazilian organizations reality, are out of bound of the organizations aimed by this work. Small organizations should be advised to follow the steps of best practices adoption in a continuous, less expensive and less risky quality improvement program. We believe this approach will lead small organizations through a quality roadmap.

The contribution of this paper is presenting a systematic procedure to assess key parameters related to software development context. The procedure includes a questionnaire application and a systematic interpretation of the collected data. These measurements can guide the choice of the best practices during the software process tailoring.

The procedure was tested over a sample of small software development organizations within an ongoing project. The partial results presented here are a step towards our main goal. More data should be collected, solutions based on the data should be devised and implemented in the organizations, what means people training and adaptation, data collection on this phase also, evaluation of the learning curve for each organization, statistical data analysis, and so on. We seek adequate solutions, not necessarily best solutions.

Our next step will be going back to the organizations and devising possible solutions based on the profiles. If it is necessary, more data should be collected

regarding factors not considered so far, that will help guide best practices adoption more precisely.

Acknowledgements. The authors would like to thank to *CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* for the financial support provided to the first author in the form of a scholarship. The early days of the project were supported by *FAPEMIG - Fundação de Amparo à Pesquisa do Estado de Minas Gerais.*

References

1. Norman, D.: The Invisible Computer. MIT Press, Cambridge, MA (1998)
2. Pressman, R.: Software Engineering: A Practitioner's Approach. McGraw-Hill, Rio de Janeiro (2001)
3. Rocha, A., Maldonado, J., Weber, K.: Qualidade de Software: Teoria e Prática. Prentice Hall, São Paulo (2001)
4. ISO/IEC: ISO/IEC 9126-1.2, Information Technology - Software product quality - part 1: Quality model. (1998)
5. Lonchamp, J.: A structured conceptual and terminological framework for the software process engineering. In: Second International Conference on the Software Process, pp. 41–53 (1993)
6. Gruhn, V.: Process-centered software engineering environments a brief history and future challenges. Annals of Software Engineering, vol. 14, pp. 363–382 (2002)
7. Feiler, P., Humphrey, W.: A structured conceptual and terminological framework for the software process engineering. In: Software process development and enactment: Concepts and definitions, pp. 28–40 (1993)
8. Boehm, B., Turner, R.: Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley, Boston (2004)
9. Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley Longman Publishing Co., Boston (2003)
10. Nichols, R., Connaughton, C.: Software Process Improvement Journey: IBM Australia Application Management Services. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2005-TR-02, (2005)
11. OMG: Software Process Engineering Metamodel Specification. Version 1.0. Massachusetts: Object Management Group, Tech. Rep. formal/02-11-14 (2002)
12. Segal, J.: When software engineers met research scientists: a case study. Empirical Software Engineering, vol. 10, no. 4, pp. 517–536 (2005)
13. Beck, K.: Extreme Programming Explained: Embrace Change. Addison Wesley Longman, Inc. (2000)
14. Cockburn, A.: Agile Software Development. Addison-Wesley, Boston (2002)
15. A. D. Methods, Scrum, http://www.controlchaos.com/ (2007)
16. Basili, V.: Software modeling and measurement: The Goal Question Metric paradigm. UMIACS-TR-92-96, Tech. Rep., cS-TR-2956 (1992)
17. Basili, V., Caldiera, G., Rombach, D.: Goal Question Metric Approach. Encyclopedia of Software Engineering, pp. 528–532 (1994)
18. Solingen, R., Berghout, E.: The Goal/Question/Metric Method A Practical Guide for Quality Improvement of Software Development. McGraw-Hill Publishing Company, England (1999)

19. Solingen, R., Basili, V., Caldiera, G., Rombach, D.: Goal Question Metric (GQM) approach. Encyclopedia of Software Engineering (2002)
20. Agerfalk, P., Fitzgerald, B.: Flexible and distributed software processes: old petunias in new bowls? Commun. ACM, vol. 49, no. 10, pp. 26–34 (2006)
21. Soares, L.: Obtenção de requisitos para customização de processo de desenvolvimento de software. Master's thesis, Universidade Federal de Viçosa, CCE/DPI, dissertação de Mestrado (2007)
22. Chrissis, M., Konrad, M., Shrum, S.: CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley. 2nd ed. Illustrated. 676p (2006)
23. SOFTEX Mps.br - melhoria de processo do software brasileiro, Version 1.2 (2007)

Appendix: Questions from the managerial questionnaire

---

3 - During the software development cycle, changes can occur in the requirement already specified. In some cases, the reason for those changes is alterations occurred in the problem context. Other reason can be the viewing of delivered parts of the system (prototyping) what can lead the user to include other functionalities not yet specified. One more reason can be the incorrect extraction of requirements that implies in requirements changes in later development stages. According to your development context, the reason for requirements change is:

A) Alterations occurred in the problem context (only one option)
( ) 100% of the occurrences ( ) 30% to 50% of the occurrences
( ) 80% to 100% of the occurrences ( ) 0% to 30% of the occurrences
( ) 50% to 80% of the occurrences ( ) never

B) The use of delivered parts of the system (prototyping) what can lead the user to desire other functionalities not yet specified occurs (only one option)
( ) 100% of the occurrences ( ) 30% to 50% of the occurrences
( ) 80% to 100% of the occurrences ( ) 0% to 30% of the occurrences
( ) 50% to 80% of the occurrences ( ) never

C) The incorrect extraction of requirements that implies in requirements changes in later development stages (only one option)
( ) 100% of the occurrences ( ) 30% to 50% of the occurrences
( ) 80% to 100% of the occurrences ( ) 0% to 30% of the occurrences
( ) 50% to 80% of the occurrences ( ) never

4- What is the frequency of requirements changes in the system (only one option)?
( ) never ( ) monthly ( ) every six months
( ) daily () every two months ( ) annually
( ) weekly () every three months

5 - Approximately, each one of those changes affect what % of the total of requirements established until the moment of the change (only one option)?
( ) 0% to 20% ( ) 40% to 60% ( ) 80% to 100%
( ) 20% to 40% ( ) 60% to 80%
