Quality of Process Control in Software Projects

Yegor Bugayenko

TechnoPark Corp.
568 Ninth Street South 202
Naples, Florida 34102
egor@tpc2.com

Abstract. The method described in the article consolidates and automates the mechanism of quality of process (QoP) measurement in software development projects. Quality policy is defined as a collection of QoP requirements, with respect to software modules, that automatically calculate the value of each QoP metric. The weighted average of all metrics is a QoP indicator, which is used for the project quality control and team motivation. An experimental list of QoP requirements based on CMMI recommendations is presented.

Keywords: Quality of Process, Process Improvement, Quality Control

1 Introduction and Problem Statement

Quality of Process (QoP) in software projects is measured as compliance to a set of pre-defined process standards and requirements. There are a number of industry-wide process standards; while some of them are more precise, others less. Good examples of such standards are CMMI, PMBOK, RUP, MSF, Scrum, PRINCE2, ISO-9001, IEEE standards, UML, and coding rules and conventions.

The selection of the right set of standards, applicable to a given project, is a complicated task for a project manager. However, a much more difficult task is to keep the project (team, artifacts, results, processes) compliant to the selected set of standards during the entire project life-cycle. What is almost impossible for the majority of projects is a regular reporting of the level of compliance. Senior management and project stakeholders want to know how compliant the given project is to the standards required. In other words, what is the QoP on a numeric scale.

Without such a consolidated QoP measurement, neither the project manager nor project stakeholders can get a clear understanding of how compliant the project is to the standards desired.

The heterogeneous nature of project artifacts and lack of formality in existing standards are the two biggest difficulties in the automation of QoP. The article suggests a method of automated QoP calculation and gives an example of QoP requirements for one particular CMMI process area.

2 The Solution, a Formal Approach

The instant value of QoP is defined as $Q(t)$ in $[0;1]$ interval, calculated at any moment of time t as a weighed average:

$$Q(t) = \frac{\sum_{i}^{n} w_i \times R_i(t, A)}{\sum_{i}^{n} w_i}, \quad t_0 < t < t_1, \quad R_i \in R \tag{1}$$

Where n is the total number of QoP requirements defined in the project, and R_i is a function in $[0;1]$ interval, indicating the compliance of project artifacts A to an i-th requirement at a given moment of time t in the project schedule. It is assumed that the project schedule is baselined, starts at t_0, and the finish is planned on t_1. w_i is the weight rank of an i-th requirement, indicating the importance of this particular requirement in the entire formula.

Higher values of $Q(t)$ and R_i mean better compliance to requirement(s), while lower values mean the opposite. The process completely complies with the given set of requirements R iff $Q(t)$ equals to 1.

During the project life-cycle, $Q(t)$ will have different values, as shown in the graph in Figure 1.

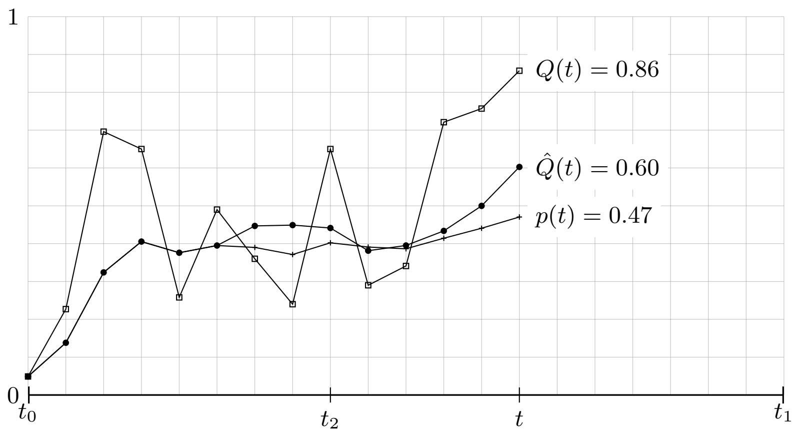

Fig. 1. Sample graph showing the dynamic of $Q(t)$ changing during the project life-cycle and the running average $\hat{Q}(t)$, which is a less pick-sensitive and more accurate indicator of the Quality of Process. $p(t)$ is the probability of the achievement of project objectives, according to the defined standard R.

The running average for the quarter of total schedule duration in equation (2) more accurately indicates the Quality of Process $\hat{Q}(t)$.

$$\hat{Q}(t) = \frac{\displaystyle\int_{t_2}^{t} Q(t)dt}{t - t_2}, \quad t_2 = t - \frac{t_1 - t_0}{4} \tag{2}$$

It is clear that $p(t)$ in equation (3) is an indicator of process compliance to a given quality standard, expressed by a set of requirements R, at any give moment of time t:

$$p(t) = \frac{\displaystyle\int_{t_0}^{t} Q(t)dt}{t - t_0} \in [0; 1] \tag{3}$$

If the quality standard R in its maximum application guarantees a successful achievement of project objectives[1], then $p(t)$ is an instant probability at the given moment of time t of such an event, as visually indicated in Figure 1.

The project will achieve its objectives if all of followings statements are true:

1. The quality standard guarantees success if applied in its maximum
2. The set of requirements R is derived from the quality standard
3. $\forall t \in [t_0; t_1] : p(t) = 1$

The definition and analysis of quality standards is out of scope of the article. There are a number of mature quality standards, which independently or combined give a guarantee of the project success, if being applied to their maximum.

The calculation of $p(t)$ is a technical task that could be easily accomplished, if all $R_i(t, A)$ are formally defined and automated.

It is necessary to define and automate a set of requirements R that will analyze project artifacts A from different points of view. Section 4 shows how artifacts could be automated in the project, and requirements could be defined on such a set of artifacts, according to CMMI verbal recommendations.

3 Technical Implementation

Technically, the method explained above is implemented as an endless cycle between the automatic “Calculator of QoP,” “QoP Rate Indicator,” and “Motivation System,” as shown in Figure 2. The Calculator uses the database of artifacts and a set of requirements as software functions.

Using the automated requirements, the Calculator collects measures from artifacts. Section 4 explains how automated artifacts provide such homogeneous measures and how automated requirements are defined in order to process the measures. The result of this process is a set of R_i required in equation (1). The Calculator produces $Q(t)$ and passes it to the QoP Rate Indicator.

[1] Achievement of project objective does not always mean the successful delivery of the product, but it always mean the closure of the project according to the baseline (scope, cost, risks, schedule, quality).

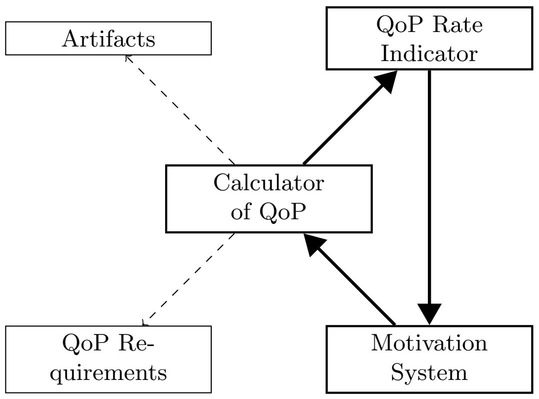

Fig. 2. Technical implementation of the method explained in Section 2, which includes three process components (Calculator of QoP, QoP Rate Indicator, and Motivation System) and two data components (Artifacts and QoP Requirements).

The QoP Rate Indicator distributes the calculated values to project stakeholders in a format convenient for reading and understanding (web page, document, e-mail, etc.) The indication could be either compact, like values of $Q(t)$, $\hat{Q}(t)$, and $p(t)$, or detailed with a full list of R_i along with their measures and explanations.

Values of QoP indicators should impact the motivation of project stakeholders, as defined in Motivation System. As shown above the value of $p(t)$ is a probability of project success.

The last and the most important process flow is between the Motivation System and the Calculator. Process engineers from Software Engineering Process Group (SEPG) or Project Management Office (PMO) should have a negative motivation for high values or QoP in each project.

4 Automated Artifacts and Their Metrics

CMMI, in its latest edition [1], breaks down the software development process into 22 process areas, and each of them may be automated with QoP requirements. The list of requirements below automates the "Configuration Management (CM)" process area. All of these requirements are validated by software modules in a fully automated mode.

"Repository Structure Is Baselined" (SP1.1) A structured list of configuration items exists and is approved. Every configuration item has a unique identifier (file name or URL), type, and an owner. The list is approved by the project manager and is not changed by anyone afterwards.

"ACL Is Baselined" (SP1.2) An Access Control List (ACL) of the repository exists and is approved, including roles and permissions. The ACL is approved by the project manager.

"SCM Engine Is Live" (SP1.2) The Software Configuration Management (SCM) system is working and is in a consistent state. The consistency of SCM is validated by its built-in engine mechanisms. Most industry SCM systems have such a mechanism.

"Change Requests Database Is Live" (SP1.2) The front-end for change request registration and the back-end for change requests storage are working and are in a consistent state.

"Archived Copy Consistency" (SP1.2) A backup/archive copy of the repository content is available, fresh, and consistent.

"Release Ownership Validity" (SP1.3) The owner of every release should be either a project manager or some other project team member with relative rights granted.

"Mean Time to Produce a Release" (SP1.3) As suggested in [12] it means how often the project releases deliverables.

"Justification of Changes" (SP2.1) The rate of changes that are back-traced to change requests. A high rate is desired.

"Granularity of Changes" (SP2.1) The mean number of changes made per change request [13]. A low granularity is desired.

"Changes are Made in Branches" (SP2.2) Tge ACL of the repository is configured to prevent changes to the `/trunk` by the project team, while only few people have rights to merge branches/tags with main thread in the repository.

"Rate of Changes Documenting" (SP2.2) The portion of changes documented by their authors.

"Quality of Comments" (SP3.1) A text template of change comments is used and all changes comply with this template.

"Time to do a Configuration Audit" (SP3.2) Regularity of configuration management audits [12]

Applying, calculating and satisfying these requirements to their maximum will indicate that the project performs configuration management well enough to guarantee the success of the outcome.

5 Conclusion

QoP should be measured and controlled in modern software development projects. The long list of currently existing quality standards and development frameworks are not intended for a quantitative measurement of the QoP. The article showed the method of metrics deriving from major software development artifacts.

References

[1] Software Engineering Institute, CMMI for Development, Version 1.2, CMU/SEI-2006-TR-008, August 2006.

[2] TechnoPark Corp., Most Popular Software Development And Project Management Industry De-facto Standards (summary), http://www.technoparkcorp.com/innovations/standards

[3] Berenbach,, B., Borotto, G.: Metrics for model driven requirements development, ICSE'06: Proceedings of the 28th international conference on Software engineering, pp. 445–451, Shanghai, China, ACM (2006)

[4] Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap, ICSE'00: Proceedings of the Conference on The Future of Software Engineering, pp. 35–46, Limerick, Ireland, ACM, New York, NY, USA (2000)

[5] Cheng, B.H.C., Atlee, J.M.: Research Directions in Requirements Engineering, FOSE'07: 2007 Future of Software Engineering, pp. 285–303, IEEE Computer Society, Washington, DC, USA (2007)

[6] Kanjilal, A., Sengupta, S., Bhattacharya, S.: Analysis of complexity of requirements: a metrics based approach, ISEC'09: Proceeding of the 2nd annual conference on India software engineering conference, pp. 131–132, Pune, India, ACM (2009)

[7] Lauenroth., K., Pohl., K.: Towards automated consistency checks of product line requirements specifications, ASE'07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, pp. 373–376, Atlanta, Georgia, USA, ACM (2007)

[8] Kharb, L., Singh, R.: Complexity metrics for component-oriented software systems, SIGSOFT Software Engineering Notes, vol. 33, no. 2, pp. 1–3, ACM, New York, NY, USA (2008)

[9] Yang., Q., Li., J.J., Weiss, D.: A survey of coverage based testing tools, AST'06: Proceedings of the 2006 international workshop on Automation of software test, pp. 99–103, Shanghai, China, ACM (2006)

[10] Kim,, Y.W.: Efficient use of code coverage in large-scale software development, CASCON'03: Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, pp. 145–155, Toronto, Ontario, Canada, IBM Press (2003)

[11] Bonja., C., Kidanmariam., E.: Metrics for class cohesion and similarity between methods, ACM-SE 44: Proceedings of the 44th annual Southeast regional conference, pp. 91–95, Melbourne, Florida, ACM (2006)

[12] Bendix, L., Borracc, L.: Towards a suite of software configuration management metrics, SCM'05: Proceedings of the 12th international workshop on Software configuration management, pp. 75–82, Lisbon, Portugal, ACM (2005)

[13] Schackmann, H., Lichter, H.: Process assessment by evaluating configuration and change request management systems, WUP'09: Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010, pp. 37–40, Cape Town, South Africa, ACM (2009)