Initial Industrial Experience of GQM-based Product-Focused Project Monitoring with Trend Patterns

Hidenori Nakai, Kiyoshi Honda, Hironori Washizaki, Yoshiaki Fukazawa
Waseda University
Dept. of Computer Science and Engineering
Tokyo, Japan
hide-and-seek@toki.waseda.jp, khonda@ruri.waseda.jp, {washizaki, fukazawa}@waseda.jp

Ken Asoh, Kaz Takahashi, Kentarou Ogawa, Maki Mori, Takashi Hino, Yosuke Hayakawa, Yasuyuki Tanaka, Shinichi Yamada, Daisuke Miyazaki
Yahoo Japan Corporation
Tokyo, Japan
{keasoh, katakaha, keogawa, makimori, tahino, yhayakaw, yasutana, shyamada, daimiyaz}@yahoo-corp.jp

***Abstract*— It is important for project stakeholders to identify a state of projects and quality of products. Although metrics are useful for identifying them, it is difficult for project stakeholders to select appropriate metrics and determine the purpose of measuring metrics. We propose an approach which defines the measured metrics by GQM method support identifying tendency of projects and products based on Trend Pattern. Additionally, we implement a tool as Jenkins Plugin which visualizes an evaluation results based on GQM method. We perform an industrial case study, which object is two software development projects. In our industrial case study, we can identify the tendency of project and product. We also identify the problem that product contains. Therefore if project stakeholders use our approach and tool, then they can identify the problem of project and product.**

Keywords—GQM, Visualization, Trend Pattern, Software Engineering

I. Introduction

It is important for project stakeholders to recognize a state of projects (e.g., test is insufficient) and quality of products (e.g., there are many defects). If they identify a state and quality, they can identify some defects more quickly and decrease costs for improve them. For example, as problems of source code quality affect the overall systems[1], their quickly detection helps project stakeholders to decrease costs. Additionally, this motivate to improve the process and quality.

Often metrics are used for identifying the state, quality, and tendency of projects and products. In CMM[2]/CMMI[3] which provide the roadmap, it is required for quantitative project management and improvement of the process that some metrics should be calculated and analyzed. However, it is difficult for project stakeholders to select appropriate measuring metrics and determine the purpose of measuring because they do not understand the purpose of a metric and what metrics they should measure. One way to resolve this problem is the Goal-Question-Metric (GQM) method[4]. The GQM method is used for describing relationship between metrics and measurement purpose (this is often equal with project goal) by using questions. Questions are evaluated for determining whether measurement purpose (project goal) is achieved or not. These items are described as a model.

Therefore, we propose the approach which defines measured metrics using the GQM method. Additionally, we propose the Trend Patterns of metrics and questions tendency which are defined by the GQM method. There are nine patterns and these patterns can support identifying tendency of projects and products. Project stakeholders can identify the tendency of project state and product quality by using Trend Patterns.

We implement a tool as Jenkins Plugin which visualizes an evaluation results based on GQM model. This tool is called GQM Plugin. Jenkins is a popular tool for continuous integration which is a framework that performs build, test, and inspection regularly and automatically, and results are provided to project stakeholders as feedback.

Additionally, we perform case study in software development project. In our case study, we can recognize that there is a problem about project and product, then this problem remains unresolved. As project stakeholders identify the problem, it motivates them to improve these problems.

Our main contribution are:

- We propose an approach to recognize tendency of projects and products.
- We propose Trend Patterns to recognize tendency of project state and product quality.
- We implements a tool named GQM Plugin as Jenkins plugin which is visualize an evaluation results and tendency of project and product.
- We perform case study in two software development projects, then we can identify the problem of product.

II. Approach

A. Overview

Our approach is intended for use in a specific process. Fig. 1 shows an overview of this process.

1) Create GQM model: First of all, project stakeholders must define project goals, and then create an appropriate GQM model throgh some workshops. Simultaneously, the

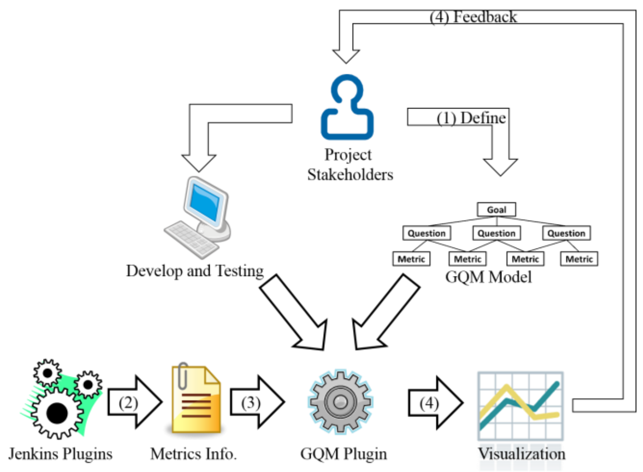

Fig. 1. Process overview

metrics' thresholds should be defined. And then, they develop and test a software product.

2) Mesure Metrics: Metrics, which are defined in GQM model, are measured by other Jenkins plugins, such as Cobertura Plugin[1], Clover Plugin[2], or Checkstyle Plugin[3].

3) Collect and Evaluation: Our tool, GQM Plugin, collects some metrics information from other Jenkins Plugin's output, then evaluates metrics and questions based on GQM model and threshold. They are evaluated as one of three categories: "Error", "Warning", and "Normal". "Error" metrics/questions indicate that they hardly meet the threshold, and they should be improved quickly. "Warning" metrics/questions indicate that they almost meet the threshold but they should be more improved. "Normal" metrics/questions indicate that they meet the threshold.

4) Establish Reports and Feedback: After finishing evaluation, GQM Plugin establish a report and three types of trend graph: GQM Report, Metrics Trend Graph (MTG), Question Trend Graph (QTG), and Metrics Value Trend Graph (MVTG). According to these report and trend graphs, project stakeholders identify tendency of project state and product quality, then they can improve own project and product.

B. *GQM Report and Trend Graphs*

GQM Report denotes metrics value and evaluation results based on GQM model. The evaluation results are described by three color, and according to this report, project stakeholders can recognize what metrics and questions are evaluated as "Error", "Warning", or "Normal". If there are "Error" or "Warning" metrics/questions, they are identified as must be improved. This report helps project stakeholders to realize what factor adversely affects the achievement of project goals.

MTG and QTG indicate how many metrics/questions are evaluated as "Error", "Warning", or "Normal". MVTG describes value of metrics in each builds. Metrics which is described in MVTG are Lines-of-Code (LOC), test coverage, the number of coding standard violation, the number of warnings about JavaDoc, and the number of open/close issue managed on GitHub. These three trend graph describes each value in time-series. Therefore according to these trend graph, project stakeholders can recognize the time-series variation of metrics values and evaluation results. Additionally, according to MTG and QTG, they can also recognize the tendency of project state and product quality using Trend Patterns.

C. *Trend Patterns*

We define nine patterns of MTG and QTG. We summarize the patterns in Table I. In Table I, "Up" means the number of error/warning metrics/questions is decreasing, "Stable" means it is not changed, and "Down" means it is increasing. Additionally, main patterns (Puu, Pss, Pdd) are shown in Fig. 2. The Puu is best pattern and the Pdd is worst pattern. When the end of project (e.g., at the release of a product), project is required to have Pdd.

Project stakeholders identify the tendency of project state and product quality by comparing trend pattern and own MTG and QTG. This process is performed every end of project cycle. If they have bad pattern (e.g., Pdd, Psd, and so on), project is not able to achieve the project goal probably.

TABLE I. TREND PATTERNS

Metrics State	**Questions State**		
	Up	***Stable***	***Down***
Up	Puu: Project state and product quality is improving. Project goal may be achieved.	Pus: Some metrics are improved, but they are insufficient to achieve project goal. Other metrics should be improved.	Pud: Project state and product quality is worsened, while many metrics are improved. Stakeholders should realize worsened questions and improve them as soon as possible.
Stable	Psu: Project state and product quality are improved, while some metrics are may be worsened. Stakeholders should check these metrics by GQM Report.	Pss: Project state and product quality is not changed. The number of error/warning should be decrease.	Psd: Project state and product quality are worsened, and some metrics are may be worsened. Stakeholders should check these metrics by GQM Report and improve them as soon as possible.
Down	Pdu: Project state and product quality is improved, while some metrics turn worse newly. Stakeholders should realize these new metrics.	Pds: Some metrics are worsened, but they do not adversely affect the achievement goal. Stakeholders should realize what metrics are worsening.	Pdd: Project state and product quality is worsening. Project process must be changed to be improve the state and quality.

[1] https://wiki.jenkins-ci.org/display/JENKINS/Cobertura+Plugin
[2] https://wiki.jenkins-ci.org/display/JENKINS/Clover+Plugin
[3] https://wiki.jenkins-ci.org/display/JENKINS/Checkstyle+Plugin

Thus, project stakeholders should change the project process to improve error/warning metrics and questions as soon as possible.

III. Industrial Case Study

A. Case Study Design

We performed an evaluation experiment to assess effectiveness of approach and GQM Plugin. The objects of evaluation is two actual software development projects, project A and project B. Both projects are web software development projects in same organization. We define the project goal as “To ensure the functionality”, “To ensure the maintainability”, and “To decrease the remedy time for bugs caused by the source code”, then we create GQM model, whose parts are shown Table II. “Stay time” in Table II means time from detection to correction of defects. This GQM model contained three goals, 13 questions, and 21 metrics. In this evaluation, our tool collects metrics information from Cobertura Plugin, Checkstyle Plugin, and Reliability Plugin[4], then we evaluate these metrics information without project stakeholders.

Additionally, we carried out questionnaire survey to assess usefulness of our approach for project stakeholders.

B. Results

In project A we collects metrics information by Reliability Plugin, and in project B we collects them by Reliability Plugin and Cobertura Plugin. Fig.3 shows MTG and QTG of project A, Fig.4 shows project B, and Table III shows metrics value of each module in both of projects. In Fig. 3 and Fig. 4, the y-axis denotes the number of error/warning/normal metrics or questions, while x-axis denotes the number of build.

All modules of both of projects has one error metrics and two error questions, and the number of error metrics and questions does not change in each build. According to GQM Report, we can identify that error metrics is “Stay time” and error question is “Are the defects corrected quickly”.

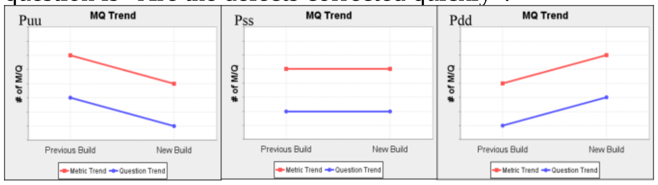

Fig. 2. Main trend pattern (Puu, Pss, Pdd)

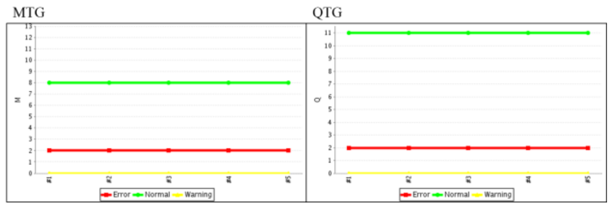

Fig. 3. MTG and QTG of project A

[4] We have developed other plugin. This plugin predicts the number of defects. The algorithm is described in [5]. It is private plugin.

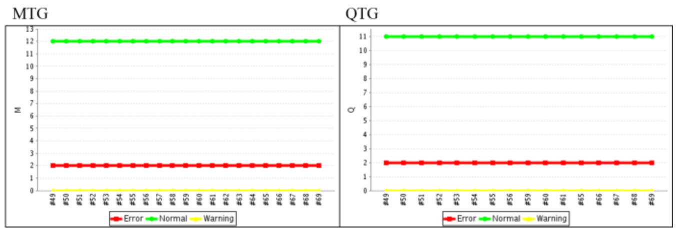

Fig. 4. MTG and QTG of project B

TABLE II. Parts of the GQM Model

Goal	Question	Metric
To ensure the functionality	…	…
	Are there enough tests on important modules?	Test Coverage
		Fun-Out
	Are there few defects?	The density of defects
		# of uncorrected defects
		# of corrected defects
		# of potential defects
	Are the defects corrected quickly?	Stay time
		Importance of defects
	…	…

TABLE III. Project Metrics Value

Metrics Name	Module Name				
	M-A1	***M-A2***	***M-A3***	***M-A4***	***M-B***
# of uncorrected defects	3	5	2	1	50
# of corrected defects	68	91	29	23	1153
# of potential defects	-5	-13	-4	-4	37
Stay time	9 days over	6 days over	7 days over	8 days over	9 days over
Test coverage	-	-	-	-	93.5%

IV. Discussion

In both of projects, the number of error metrics and questions does not change in each build. According to this result, we can recognize these projects has Pss of Trend Patterns, these projects’ state and product quality is not change. Additionally, error metrics and questions are not many. So these projects may be able to achieve the own project goal.

However, we can identify the problem that all modules of these projects have same error, that time from detection to correction of defects is so many. This probably causes the increase of the cost for correcting the defects. Therefore, we should improve this problem.

In this case study, we define 21 metrics in GQM model to recognize a project state and product quality. However, according to questioner survey result, our approach and GQM Plugin motivate to improve metrics values, but metrics is insufficient to recognize a product quality. Additionally, period using the GQM tool is too short to improve a process and product. Therefore, we cannot assess the effectiveness for identifying the tendency of project and product.

We can identify the problem about time to correct defects. However, we cannot identify the other problem of project state and product quality. Thus, we should inspect whether our approach and GQM Plugin can identify the other problem or not.

V. RELATEDWORK

The Software Project Control Center (SPCC) introduced in [6] is useful for systematic quality assurance and management of software development projects [7]. Using SPCC, a project manager can understand the state of a project and check the quality more easily. From this, the Specula approach has been proposed [7], [8]. This approach collects measurement data based on the GQM model. The collected data is interpreted, evaluated, visualized, and feedback to project stakeholders. If the control center is used in first iteration of the software development projects, the projects can make a quite good start [9]. Similar to our approach, interpretation and visualization are based on the GQM method. Because our approach also includes the trend pattern, whether a goal is attainable is easier to determine.

The Empirical Approach to Software Engineering (EASE) project developed project measurement platform called Empirical Project Monitor (EPM) [10], [11]. EPM collects project management data automatically from some tools such as configuration management system, CVS, mailing list management system, mailman, issue trucking system, GNUTS. The collected information includes changes for source code and time of check-in/check-out from configuration management system, fault report and fault correction report from mailing list system or issue trucking system [12]. EPM analyze and visualize these data and feedback to software development project. In [13], Monden creates the model of detecting main software project delay causes using the GQM method. Using EPM involved in using other tools, CVS, GNUTS, and mailman. Because only using Jenkins, our approach apply the software development projects more easily.

VI. CONCLUSION AND FUTUREWORK

In this paper, we propose an approach which defines the measured metrics by GQM method, presents a metrics' time-series variation, and support identifying tendency of projects and products based on Trend Pattern. Also we implement a GQM Plugin as Jenkins plugin. This plugin collects metrics information, and evaluate metrics and question based on GQM model. After evaluating, GQM plugin establish GQM Report and three trend graphs. From these report and trend graphs, project stakeholders can recognize a project state and product quality. Also, they can identify some insufficient items which are needed for achieve project goals.

In a case study, we can recognize the problem, time from detection to correction of defects. So our approach is effective for identifying problem of project state and product quality. However, metrics which are defined in GQM model and period using GQM Plugin is insufficient to identify the tendency of project and quality.

In the future work, we adopt our approach and GQM Plugin to software development project continuously to assess the effectiveness of them in long term. Additionally, we extend our GQM model to collect more kinds of metrics information to propose more detailed trend of project state and product quality to project stakeholders. By this, we can identify a tendency of project state and product quality more exactly from many aspects.

ACKNOWLEDGMENT

We thank the developers and quality managers that cooperated with us while creating the GQM model and industrial case study.

REFERENCES

[1] H. Washizaki, et al., "A Framework for Measuring and Evaluating Program Source Code Quality," PROFES 2007, 2007.

[2] M.C. Paulk, et al., "Capability maturity model (version 1.1)," IEEE Software, vol.10, no.4, , pp.18-27, 1993.

[3] Team, CMMI Product "Capability Maturity Model® Integration (CMMI) version 1.1," CMMI for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1), 2001.

[4] V.R. Basili, et al., "Goal Question Metric Approach," Encyclopedia of Software Engineering, John Wiley & Sons, Inc., , pp. 528-532, 1994.

[5] K. Honda, et al., "A generalized software reliability model considering uncertainty and dynamics in development," PROFES 2013, 2013.

[6] J. Munch, et al. "Software Project Control Centers: Concepts and Approaches," Journal of Systems and Software 70(1), , pp.3-19, 2003.

[7] J. Heidrich, et al., "Goal-Oriented Setup and Usage of Custom-Tailored Software Cockpits," PROFES 2008, 2008.

[8] P. Liggesmeyer, et al., "Visualization of Software and Systems as Support Mechanism for Integrated Software Project Control," HCI 2009, pp. 846-855, 2009.

[9] M. Ciolkowski, et al., "Evaluating software Project Control Centers in Industrial Environments," ESEM 2007, 2007.

[10] M. Ohira, et al., "Empirical Project Monitor; A Tool for Mining Multiple Project Data," MSR 2004, 2004.

[11] Y. Mitani, et al., "A Proposal for Analysis and Prediction for Software Projects using Collaborative Filtering, In-Process Measurements and a Benchmarks Database," MENSURA 2006, 2006.

[12] M. Ohira, et al., "Empirical Project Monitor: A System that Automatically Collects and Analyzes Quantitative Development Data toward Process Improvement," Technical report of the Institute of Electronics, Information, and Communication Engineers, Software Science in Japanese, pp. 13-18, 2004.

[13] A. Monden, et al., "Customizing GQM Models for Software Project Monitoring," IEICE Trans., Vol. E95-D, No. 9, pp. 2169-2182, 2012.