Continuous Product-Focused Project Monitoring with Trend Patterns and GQM

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, and supports 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 experiment with OSS and industrial case study with two software development projects. In our experiment, we can identify the problem and project tendency. In our industrial case study, we can also identify the problem that project contains. As our future work, we adopt our approach and GQM Plugin to software development project continuously to assess the effectiveness of them in long term.**[1]

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[2], their quickly detection helps project stakeholders to decrease costs. Additionally, this motivates to improve the process and quality.

Often metrics are used for identifying the state, quality, and tendency of projects and products (e.g., metrics are used for evaluation of the reusability of C language program source code[3]). In CMM[4]/CMMI[5] which provides 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 metrics and what metrics they should measure. One way to resolve this problem is the Goal-Question-Metric (GQM) method[6]. 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.

[1] The idea of our approach and GQM Plugin is included in [1]. In this paper we added overview of trend patterns and OSS experiment.

In this paper, 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 an experiment with OSS and industrial case study in software development project. In our experiment with OSS, we can identify the problem that project contains and project tendency. 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 contributions 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 implement a tool named GQM Plugin as Jenkins plugin which is visualize an evaluation results and tendency of project and product.
- We perform an experiment with OSS, then we can identify the problem of project and its tendency.

- We perform case study in two software development projects, then we can identify the problem of project.

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 through some workshops. Simultaneously, the metrics' thresholds are defined to identify the tendency of project state and product quality, and to assess whether poroject can achieve project goals or not. These thresholds need to be inputted manually. After difinition of GQM model and thresholds, project stakeholders develop and test a software product .

2) Measure Metrics: Metrics, which are defined in GQM model, are measured by other Jenkins plugins. This process are performed by Jenkins automatically and simultaneously with builds of the code at specific iteration.

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 thresholds. Finally, questions are evaluated based on results of metric evaluation. Metrics and questions 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. If many metrics which are connected with a question are evaluated "Error", a question is evaluated "Error".

4) Establish Reports and Feedback: After finishing evaluation, GQM Plugin establishes 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 are also able to improve own project state and product quality.

GQM Plugin collects some metrics information from other Jenkins Plugin's output. GQM Plugin can handle an output of Cobertura Plugin[2], Clover Plugin[3], Checkstyle Plugin[4] and Reliability Plugin[5]. Hence, GQM Plugin can collect data of test coverage, LOC, the number of coding standard violation, the number of violation of JavaDoc, issues data managed by GitHub, predicted issues, and the number of potential defects.

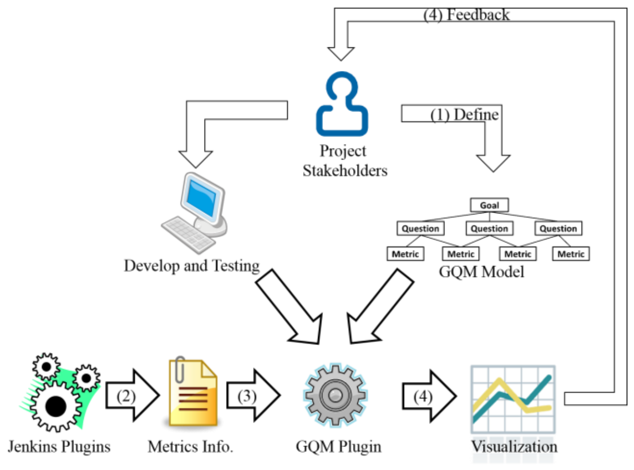

Fig. 1. Process overview

They are common Jenkins plugins. Additionally, they can be introduced easily into software development project. If the other Jenkins plugins to measure some metrics output metrics information in same format as Cobertura Plugin and so on, GQM Plugin can handle an output of the other plugin.

B. GQM Report and Trend Graphs

GQM Report denotes metric values 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, these metrics/questions adversely affects the achievement of project goals. On the other hand, if there are no "Error" or "Warning" metrics/questions, this indicates that metrics and questions have enough value to achieve the project goal. Hence, this report helps project stakeholders to realize what factor adversely affects the achievement of project goals.

MTG and QTG indicate how many metrics and questions are evaluated as "Error", "Warning", or "Normal". According to MTG, QTG, and Trend Patterns (described in follow section), project stakeholders can identify the tendency of project state and product quality. If MTG and QTG have bad pattern, project process should be changed to improve error and warning metrics/questions.

MVTG describes values of metrics in each builds. Metrics which are described in MVTG are LOC, test coverage, the number of coding standard violation, the number of violation of JavaDoc, and the number of open/close issue managed on GitHub. According to MVTG, project stakeholders can recognize information about project state and product quality. For example, if LOC is increasing and test coverage is decreasing (or it is not change), project stakeholders can identify that test is insufficient.

These three trend graphs describe each values in time-series. According to these trend graphs, project stakeholders can recognize the time-series variation of metrics values and evaluation results. Hence, project stakeholders can recoginize

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

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

[4] https://wiki.jenkins-ci.org/display/JENKINS/Checkstyle+Plugin

[5] We have developed other plugin. This plugin predicts the number of defects. The algorithm is described in [7]. It is private plugin.

what becomes large, what is improved, and what is worsening. If some metrics are worsening and they are evaluated error or warning, project stakeholders can identify them should be improved.

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 (i.e., project state and product quality are improving), "Stable" means it is not changed (i.e., project state and product quality are not changed), and "Down" means it is increasing (i.e., project state and product quality are worsening). Additionally, these patterns 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 Puu. If project does not have Puu, some problems are remain and project goal may not be yet accomplished.

Project stakeholders identify the tendency of project state and product quality by comparing trend patterns 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. Thus, project stakeholders should confirm which metrics and questions are evaluated as error/warning, and change the project process to improve error/warning metrics and questions as soon as possible.

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 are 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 are not changed. The number of error/warning should be decreased.	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 are 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 are worsening. Project process must be changed to be improve the state and quality.

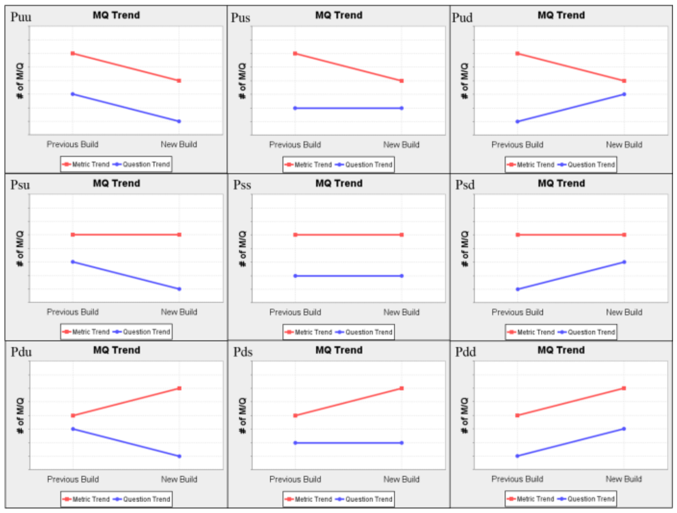

Fig. 2. Trend Patterns Overview

If they have good pattern (e.g., Puu), project stakeholders can identify that project is advancing toward achievement of goal. If there are no (or a few) error/warning metrics/questions, project process does not need to be changed.

III. INDUSTRIAL CASE STUDY

A. Case Study Design

We perform an evaluation experiment to assess effectiveness of approach and GQM Plugin. The objects of evaluation are OSS project, maven-android-plugin[6] project, and two actual software development projects, project A and project B. Maven-android-plugin is a maven plugin for android application development. Both of project A and project B are web software development projects in same organization. Project A develops a platform, and Project B develops a library. Table II shows overview information about the case study objects. In this case study, we focus on modules which use Java and PHP of Project A, and modules which use C++ of Project B.

We define the project goals 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 III. "Stay time" in Table III means time from detection to correction of defects.

TABLE II. CASE STUDY OBJECTS

Name	***Maven-android-plugin***	***Project A***	***Project B***
Domain	Maven plugin for android application development	Platform	Library
Language	Java	Java PHP	C++ Scala
Version	Maven-android-plugin2.6.0 – 3.9.0-rc.3	-	-

[6] https://github.com/jayway/maven-android-plugin

TABLE III. 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 enough tests on modules which have middle/low importance?	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
	…	…

This GQM model contains three goals, 13 questions, and 21 metrics. Additionally, we define some thresholds for each project. Table IV shows information about these thresholds.

In this evaluation, we define thresholds and evaluate these metrics information without project stakeholders. However, we create GQM model with stakeholders of Project A and Project B. Additionally, we carry out questionnaire survey to assess usefulness of our approach for project stakeholders.

B. Results

In an experiment with OSS, maven-android-plugin, we collect metrics information by Cobertura Plugin, Checkstyle Plugin, and Reliability Plugin. Hence, we collect data of test coverage, LOC, the number of coding standard violation, the number of violation of JavaDoc, issues data managed by GitHub, predicted issues, and potential defects. Fig.3 shows MTG and QTG, and Fig.4 shows a part of MVTG of maven-android-plugin. In project A we collect metrics information by Reliability Plugin. In project B we collect metrics information by Reliability Plugin and Cobertura Plugin. Hence, we can collect issues data managed by GitHub and predicted issues in Project A, test coverage, LOC, coding standard violation, violation of JavaDoc, issues data managed by GitHub, and predicted issues in Project B. Fig.5 shows MTG and QTG of Project A, Fig. 6 shows Project B.

TABLE IV. THRESHOLDS

Metrics Name	*Error Threshold*	*Warning Threshold*
Test coverage	60 %	80 %
# of open issue	80	50
# of close issue	50	80
# of potential issue	60	80
Stay time	3 days	0 day
# of violation of JavaDoc	200	100
# of coding standard violation	150	100

Table V shows metrics values of each module of maven-android-plugin, Project A, and Project B in last builds (version). In Fig. 3, Fig. 5 and Fig. 6, the y-axis denotes the number of error/warning/normal metrics or questions, while x-axis denotes the number of build. In Fig. 4, the y-axis denotes each value of test coverage, the number of open/close issues, while x-axis denotes the number of build.

Maven-android-plugin project has four error metric, one warning metric, and four error questions in last build. However, the number of error metrics and questions is decreasing. According to MVTG of this project, we can identify that test coverage is not enough in each build. On the other hand, detected issues are almost improved. Additionally, according to GQM Report, we can identify that error metrics are "Stay time" and "Test coverage", warning metric is "The number of potential defects", and error questions are "Are there enough tests on important modules?", "Are there enough tests on modules which have middle/low importance?" and "Are the defects corrected quickly?".

All modules of both of Project A and Project B have one error metric 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 metric is "Stay time" and error question is "Are the defects corrected quickly?".

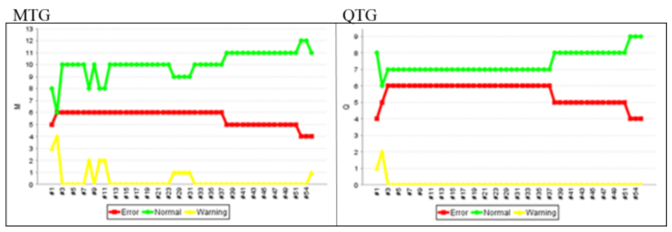

Fig. 3. MTG and QTG of maven-android-plugin

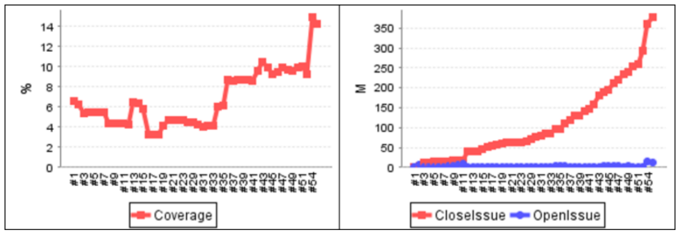

Fig. 4. MVTG of maven-android-plugin

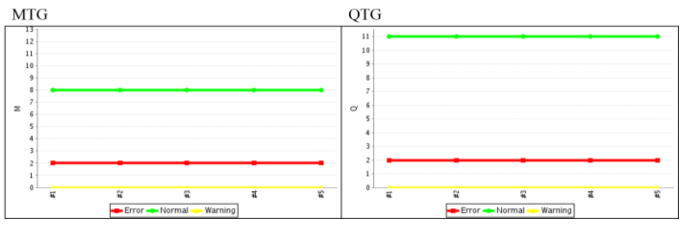

Fig. 5. MTG and QTG of Project A

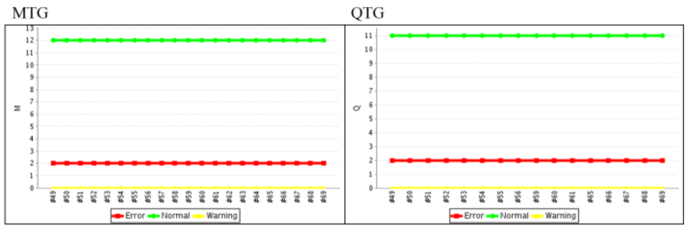

Fig. 6. MTG and QTG of Project B

TABLE V. PROJECT METRICS VALUE

Metrics Name	Module Name					
	OSS	***M-A1***	***M-A2***	***M-A3***	***M-A4***	***M-B***
# of uncorrected defects	12	3	5	2	1	50
# of corrected defects	377	68	91	29	23	1153
# of potential defects	135	-5	-13	-4	-4	37
Stay time	11 days over	9 days over	6 days over	7 days over	8 days over	9 days over
Test coverage	14.3%	-	-	-	-	93.5%
# of coding standard violation	14	-	-	-	-	-
# of violation of JavaDoc	0	-	-	-	-	-

IV. DISCUSSION

In maven-android-plugin projects, the number of error metrics and questions is decreasing. According to this result, we can recognize this project has Puu of Trend Patterns. Hence, we can identify that this project's state and product quality are improved, and this project is advancing toward achievement of goal.

However, this project has some error metrics and questions. According to experiment result, we can identify the problem that test is not enough and time from detection to correction of defects is so long. They probably cause delay of detecting issues, and the increase of the cost for correcting the defects. Thus, these problems should be improved. Additionally, this project also has a warning metric. According to experiment result, we can identify that it is "The number of potential defects". This indicates that there are some undetected defects. This problem makes the correction of the defects difficult. On the other hand, this may be caused by low test coverage. Hence, it is necessary for project stakeholders to improve test coverage, thereby some problem may be improved simultaneously.

We define the project goals as "To ensure the functionality", "To ensure the maintainability", and "To decrease the remedy time for bugs caused by the source code", and create GQM model presented in Table III. However, these goals and GQM model are defined without members of this project. Hence, these goals and GQM model may not be appropriate. To identify the problem and tendency of project state and product quality more exactly, we should define the goal and create GQM model with members of this project.

In both of Project A and Project B, the number of error metrics and questions does not change in each build. According to this result, we can recognize these projects have Pss of Trend Patterns. Hence, we can identify that these projects' states and product qualities do not change.

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

In this case study, we can identify the problem about test and time from detection to correction of defects. On the other hand, we do not 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.

Additionally, although we do not assess whether these projects are able to achieve project goals or not, we confirmed that our approach is able to recognize that maven-android-plugin projects has Puu of Trend Patterns and both of Project A and Project B have Pss of Trend Patterns. However, we define thresholds without project stakeholders. Hence, these thresholds may not be appropriate. Thus, we define thresholds with project stakeholders, then we should inspect whether our approach and GQM Plugin can recognize that project achieve the goal or not.

Furthermore, we perform case study with only three domains. Hence, we do not identify the effectiveness of our approach and GQM Plugin for other domains. Therefore, we should adopt our approach and GQM Plugin to other domain projects to assess the effectiveness.

We define 21 metrics in GQM model to recognize a project state and product quality. However, according to questioner survey result, our approach motivate to improve metrics values, but metrics are insufficient to recognize a product quality. Thus, we should we extend GQM Plugin to collect more kinds of metrics information. Additionally, a period using our approach is too short to assess the effectiveness for identifying the tendency of project and product.

V. RELATEDWORK

The Software Project Control Center (SPCC) introduced in [8] is useful for systematic quality assurance and management of software development projects [9]. 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 [9], [10]. 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 [11]. Similar to our approach, interpretation and visualization are based on the GQM method. On the other hand, our approach also includes the trend pattern, the tendency of project and product is easier to determine.

The Empirical Approach to Software Engineering (EASE) project developed project measurement platform called Empirical Project Monitor (EPM) [12], [13]. EPM collects project management data automatically from some tools such as configuration management system, mailing list management system, and issue trucking system. The collected information includes changes for source code, fault report, fault correction report, and so on [14]. EPM analyzes and visualizes these data and feedbacks to project stakeholders. In [15], Monden creates the model of detecting causes of main software project delay using the GQM method and EPM. Using EPM involves in using other tools. On the other hand, GQM Plugin uses only one platform, Jenkins. Hence, GQM Plugin can be introduced more easily into a software development project.

In [16], the Continuous Inspection pattern is presented. Continuous Inspection pattern has four steps. First of all, project stakeholders create and modify source code. Then, some tools analyze this code automatically. Next is that some reports are generated. Finally, some feedbacks are presented to improve the code. These steps communicate with a continuous integration server. Additionally, it is also presented how to adopt the continuous inspection pattern in [16]. However, trend patterns of project state and product quality are not described in this publication.

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 Patterns. Also we implement a GQM Plugin as Jenkins plugin. This plugin collects metrics information, and evaluates metrics and questions based on GQM model. After evaluating, GQM Plugin establishes 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 achieving project goals.

In an experiment with OSS, we can recognize the problem about test, time from detection to correction of defects, and potential defects. Additionally, we can recognize the project has Puu of Trend Patterns. This indicates that project state and product quality are improved.

In a case study, we can recognize the problem about time from detection to correction of defects. Additionally, we can recognize the project has Pss of Trend Patterns. This indicates that project state and product quality are not change.

However, metrics which are defined in GQM model and a period using GQM Plugin are insufficient to identify the tendency of project and quality. Hence, although do not assess whether these projects are able to achieve project goals from Trend Patterns, we confirmed that our approach can recognize the problem and identify which Trend Patterns project has.

As our future work, we adopt our approach and GQM Plugin to software development project continuously to assess the effectiveness of them in long term, and to confirm whether they can identify an achievement of project goal or not. Additionally, we also adopt them to other domain projects to assess versatility. On the other hand, we should extend GQM Plugin to collect more kinds of metrics information (e.g., cohesion, coupling, and code clone) 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 project stakeholders for their valuable comments and feedback for our approach and GQM Plugin.

References

[1] H. Nakai, et al., "Initial Industrial Experience of GQM-based Product-Focused Project Monitoring with Trend Patterns", Poster at APSEC 2014, 2014.

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

[3] H. Washizaki, et al., "Reusability Metrics for Program Source Code Written in C Language and Their Evaluation," PROFES 2012, pp, 89-103, 2012.

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

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

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

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

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

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

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

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

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

[13] 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.

[14] 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.

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

[16] P. Merson, et al., "Continuous Inspection : A Pattern for Keeping your Code Healthy and Aligned to the Architecture," Asian PLoP 2014, 2014.