2009 Fifth International Joint Conference on INC, IMS and IDC

A Flexible Metric-driven Framework for Software Process

[1]LIN Cheng, [2]HUANG Zhongdong
College of Computer Science
Zhejiang University
Hangzhou, P.R. China
[1]lcfrcw@hotmail.com, [2]hzd@cs.zju.edu.cn

***Abstract*—In software engineering, metrics are always used for measuring the quality of software process to improve software development or maintenance. In fact, there are several metric frameworks available to make metrics work efficiently, but only for the specific industrial region. In this paper, a general metrics framework is proposed, which is following the Goal/Question/Metric paradigm of Basili and Weiss (1984), to support most of the metrics. Furthermore, a sample application based on this framework is presented to demonstrate its effect on software process. The results provide some useful insights into project management practices. Finally, some enhancements are proposed for future work.**

Keywords- Metric; Metric-driven; Software Metrics Framework; GQM; Software Process; Software engineer

I. INTRODUCTION

With a great progress made in software process, metrics become a critical role in effective and efficient software development and maintenance. For example, the effectiveness of defect removal during development, the pattern of testing defect arrival, and the response time of the fix process [1]. But according to result in [12], Dutta et al. analyzed data on almost 400 companies in 20 European countries to find out that the average adoption of metrics is only 45 percent. It's urgent that we should find a general solution for metrics implement.

Obviously, we can draw a conclusion that the effectiveness and efficiency of metrics depend on following two factors: one is selection of metrics which are due to different intention for the software process, the other is the architecture or framework where metrics are deployed.

Fortunately, there is one well-known approach that is Goal Question Metric (GQM) . The authors of GQM (Basili and Weiss) proposed a methodology which could be applicable to anywhere for metrics. Furthermore, an extended GQM approach was presented by Patrik Berander and Per Jönsson, which facilitates identification of and focuses on the most important measurements for an organization [3]. There is another approach which contains three levels as Factor/Criteria/Metrics. In this solution, the measurements are taken during the development effort in order to provide an indication of the progression toward a desired level of quality [11].

Our proposed framework, based on GQM approach, and combined with the merits in Factor/Criteria/Metrics approach, can fit into most of current metrics. Moreover, it provides flexible interface for new metrics to be introduced and indicator-based reports components. The framework and the sample application have made an important step towards the goal of higher maturity of CMMI [2] which has define the Measurement and Analysis as an independent process area.

The organization of the paper is as follows. First, we discuss related works in section 2. Then, the essential layers and components of framework are presented in section 3. A sample application is provided to demonstrate the feasible of this framework in section 4. Finally, we present the conclusion and the future research work in section 5.

II. RELATED WORK

Metric framework research is always one of the most significant areas to improve the ability for metrics with executable and maintainable.

Currently, most of the current software metrics tools intend to analyze the code of the software. For example, the representative tools: LOGISCOPE used for embedded application, Panorama, Cantata++, Test2Work Advisor and etc [4]. In addition, there are some great metric-based systems like TAME [5], Ginger [6], SME [7], AMS [8], and Amadeus [9]. The main problem with the approaches which exist on software industry lies on the inability to provide a flexible interface for all kinds of metrics in various software projects. In abstract of [13], it presents three fundamental principles to make measurement active. First one is integrating measurement and process. Second is user-specifiable interpretation to allow a representative basis for project analysis. Third is to provide an extensible integration framework to support synergistic application. All above give birth to our framework.

III. FRAMEWORK OVERVIEW

Fig. 1 gives a big picture of our metric-driven framework.

There are three layers in this framework. At the bottom is Data Layer. Its main functionality is collecting, formatting and transferring the data from various reports and bug tracking system such as Bugzilla (developed by Mozilla Foundation), JIRA (developed by Atlassian), etc.

Firstly, Data Collection component is initialized by metric driver which lies on Metric Layer, it collects all necessary source data as request, then process analyse and storage executions.

978-0-7695-3769-6/09 $26.00 © 2009 IEEE
DOI 10.1109/NCM.2009.189

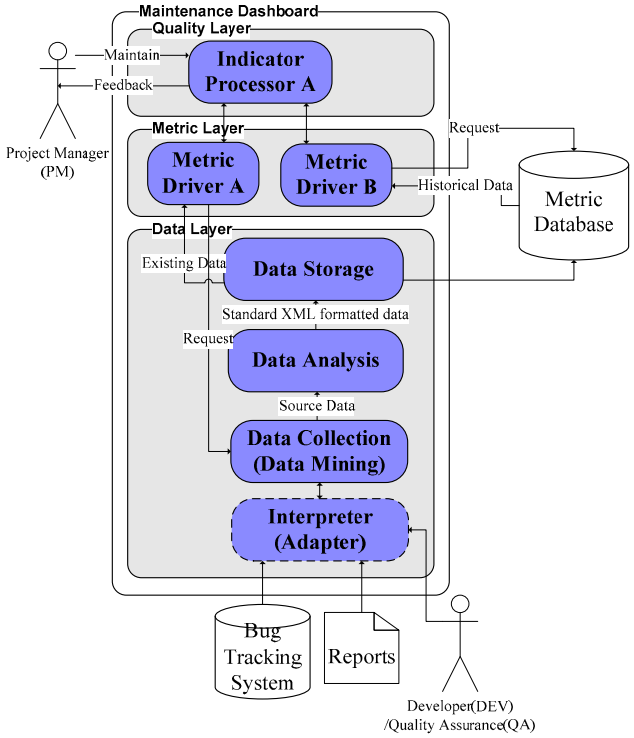

Figure 1. Framework overview

Secondly, metric driver receives the formatted data from Data Layer or Metrics Database. Then, it carries out the pre-defined algorithm that is provided by metrics.

Finally, Indicator Processor on Quality Layer receives the metrics data and generates expected reports. In addition, an essential maintenance dashboard is provided to focus on the maintenance job for metric hierarchy.

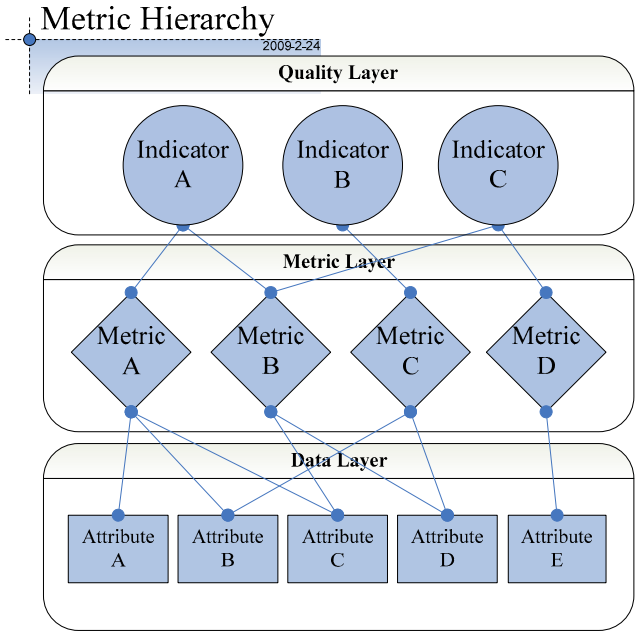

Figure 2. Metric Hierarchy

From above, we have the basic knowledge about our solution. It not only provides the flexible functionality from data selection to report generation, and also distribute the different jobs in software metrics process to various roles. To make our framework as a criterion, we also provide a Metric Hierarchy shown on Fig. 2. Indicators are the key metrics portfolio maintained by managers to fulfill their management responsibilities. The concept of "key" means the metric is considered important by the users in managing job responsibilities. Normally the key indicator is created from many different measures. For example, the Dow Jones Average is created from the combining of stock prices from approximately 40 different stocks. In the following sections, we will introduce the detailed information about how this hierarchy works efficient with our framework.

A. *Data Layer*

Data Layer deals with the selecting interested data from various inputs, validating the source data, analyzing the source data, and transferring the formatted data. As mentioned above, Basili and Weiss (1984) proposed a data collection methodology (GQM) that could be applicable to anywhere. The suggested schema consists of six steps with considerable feedback and iteration occurring at several places:

1) *Establish the goal of the data collection*
2) *Develop a list of questions of interest*
3) *Establish data categories*
4) *Design and test data collection forms*
5) *Collect and validate data*
6) *Analyze data*

The importance of the validation element of a data collection system or a development tracking system cannot be overemphasized [1].

According to GQM, we have managed to divide the expected data into several elements as attributes for the foundation of metric. This requirement also contributes to introduction of Metric Maintenance Dashboard in order to maintain all the configuration parameters in this framework.

Based on the configuration for metric driver, Data Collection component starts to receive the request from metric driver, and collect all the desirable data from related source. While collection does accomplish, it triggers the processes in Data Analysis component where the data is structured into standard XML data format as shown on Fig. 3.

All the attributes in XML data file have name, format, value and the destination path on the Metric Database. Due to the advantage of standardization with XML format, we could easily transform it into database, web, and all kinds of reports through Data Storage component. The most important role for Data Layer in framework is to provide formatted data as expected to Metric Layer for further process.

B. *Metric Layer*

Metric Layer lies at the center part in framework, and plays a core role in whole framework. It's composed by a bunch of metric drivers which are the main execution

engineers. The metric driver receives formatted data from the layer below or database, and generates the desired results through various scenarios.

Sometimes, the owners of the metric need the history metric data for the purpose such as checking the potential risk for the software process. Therefore, the framework provides the data channel between Metric Database and Metric Driver. Those valuable metric data which is collected on previous job is saved by Data Storage component.

After processed by the defined algorithms, the intermediate results are generated by each driver and sent to the upper layer.

C. *Quality Layer*

Quality Layer is the most user-oriented component, and its main functionality is to present valuable reports mostly to Project Managers. To make the service more centralized and organizable, Indicator Processors mechanism are provided in this framework. It receives the metric objects from Metric Layer, and then presents with the benchmark data in order to give the managers an overall idea of the status and performance with software process.

Many indicators existed for the software process, so managers always select several indicators as portfolio for their quality orientation. The framework provides multi-processors architecture to make this functionality work.

```
<?xml version="1.0" encoding="UTF-8"?>
<Attribute_Lists>
    <Metirc_Attribute>
        <Name>Project_Name</Name>
        <Format>String</Format>
        <Value>Project A</Value>
        <Table>tblProject</Table>
        <Field>Name</Field>
    </Metirc_Attribute>
    <Metirc_Attribute>
        <Name>Defect_Number</Name>
        <Format>Integer</Format>
        <Value>256</Value>
        <Table>tblDefect</Table>
        <Field>Number</Field>
    </Metirc_Attribute>
    <Metirc_Attribute>
        <Name>KLOC</Name>
        <Format>Double</Format>
        <Value>3.5</Value>
        <Table>tblProject</Table>
        <Field>KLOC</Field>
    </Metirc_Attribute>
</Attribute_Lists>
```

Figure 3. Standard XML data format

Besides the processor, Maintenance Dashboard maintains the benchmark and the mapping relationship between indicators with metrics. In this way, the users could easily manage the whole process in a friendly way.

IV. A SAMPLE APPLICATION

Based on the framework mentioned above, we have implemented this solution into our Quality Assurance (QA) team with several finance related applications.

It seems extremely important to manage the risk exists in the software process for the finance service under trend of world economic decline. Managers lever up the quality standard to meet the competition with the competitors. This framework is born under this background.

In order to cope with rising quality request, we develop a small application named Metrics Management System (MMS). MMS provides an overall functionality to manage the measurable test result, to generate the test metrics report and to analyze the test report for the projects in QA team.

In our team, we use JIRA system as our bug tracking system. By way of coordination with JIRA system, we set up a small database using Access 2003.

The application is written in Access with VBA. It provides friendly UI for users to manipulate the data.

We have already implemented six indicators for this tools list as follow: QA Evaluation Indicator, Test Case Quality Indicator, Bug-Omit Rate Indicator, Valid Bug Rate Indicator, Downtime Log Efficiency Indicator and Automation Return on Investment (ROI) Indicator. And we will take first two indicators as examples.

The data source from QA Evaluation Indicator is QA weekly task assignment report (Note: QA leader will have a weekly task assignment at the beginning of each week, and will collect the actual status at the end of the week. This information can be used to calculate the "Deviation Rate".). The method used for this indicator shown in (1):

$$DeviationRate = (ActualHours - EstimatedHours) / EstimatedHours \times 100\% \quad (1)$$

At last, we get report from our application as shown on Fig. 4.

According to the line chart, we can define that the schedule for this project fluctuates a lot at first time. But after the turbulence from the six week, it seems to be stabilized in the following two weeks. For managers, now they could have an overall knowledge of the status with the resource allocation and project plan.

QA Evaluation Indicator shows a good point for the project schedule, now we give another indicator (2) which is worked on this application to track the quality for test case design.

$$TestCaseQuality = FoundbyTestcase(Critical \times 3 + Normal \times 1 + Minor \times 1/3) / Total(Critical \times 3 + Normal \times 1 + Minor \times 1/3) \times 100\% \quad (2)$$

The following Fig. 5 is one of the reports generated by the indicator.

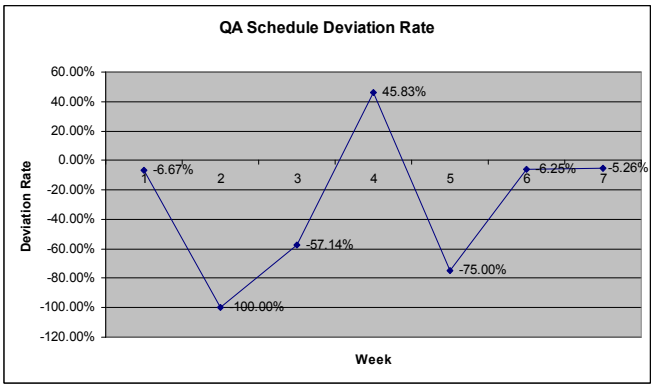

Figure 4. QA Schedule Deviation Rate

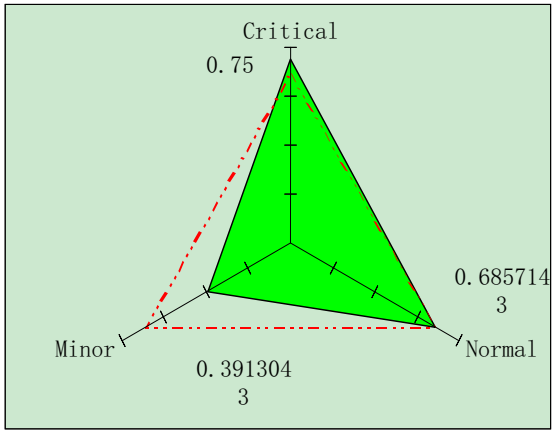

Figure 5. Test Case Quality

What's the exact lesson we learn from this report? It shows that the coverage of the test case should be increased to cover the small proportion with the Minor, or the codes for the project need to be improved for the big proportion with Critical and Normal. The root causes need more analysis, in this ways, more metrics are introduced.

All of those reports are only a tip or warning for managers to take care and make the right decision.

In practice, after analyze all the indicators mentioned above, we fixed our shortcoming, and make a great progress with the improvement. Due to this application, our team does receive great praise for our work as QA.

V. Conclusion and Future Work

The metrics framework that we proposed above introduces a standard way to implement the metrics and the indicators, and we also introduce a new metric hierarchy to enhance connection between layers.

After sample application has been implemented based on this framework, we could see the merits of our framework are list as follows:

1) Flexibility to introduce new metrics through Metric Hierarchy with Maintenance Dashboard.

2) Manageability for metric data by using XML standard data format and Metric Database.

3) Standardization data flow from collection, validation and analysis.

4) Adaptability to the bug tracking system and diversity to reports of input and output.

However, for the framework, there are still some enhancement jobs with enrichment report templates. Another issue is that we should add authorization functionality to control the roles' permission to make sure the veracity of the data.

To sum up, our solution of metric-driven framework for software process seems promising and successful with respect to its purpose.

Acknowledgment

The framework is proposed by authors only and the sample application has implemented in Quality Assurance team of State Street Zhejiang University Technology Center (SSTC). Thanks to the help from Quality Assurance team of SSTC, as well as the support of State Street Corporation in the research. The contents of this paper are the opinions and conclusions of the authors only and do not necessarily represent the position of State Street Corporation or its subsidiaries, affiliates, officers, directors or employees.

References

[1] Stephen H. Kan, "Metrics and Models in Software Quality Engineering, Second Edition", Addison-Wesley Professional, Boston, MA, USA, September, 2002, pp. 85, 118.

[2] CMMI-SE/SW Version 1.1, Copyright 2002 by Carnegie Mellon University, January 11, 2002.www.sei.cmu.edu/cmmi

[3] Patrik Berander and Per Jönsson, "A Goal Question Metric Based Approach for Efficient Measurement Framework Definition", ISESE'06, September 21–22, 2006, pp. 316 - 325

[4] Mingzhi Mao and Yunfei Jiang, "A Coherent Object-Oriented (OO) Software Metric Framework Model", 2008 International Conference on Computer Science and Software Engineering, 2008, pp. 68-72

[5] V.R.Basili and H.D.Rombach, "The TAME project: Towards improvement-oriented software environments.", IEEE Trans. Software Engr., June 1988, pp: 758-773

[6] Koji Torii, Tohru Kikuno, Ken ichi Matsumoto, and Shinji Kusumoto, "A data collection and analysis system Ginger to improve programmer productivity on software development.", Technical report, Osaka University, Osaka, Japan, 1989

[7] W.Decker and J.Valett, "Software management environment (SME) concepts and architecture.", Technical Report SEL-89-003, NASA Goddard, Greenbelt, Maryland, August 1989

[8] J.A.McCall, P.Richards, and G.Walters, "Factors in software quality", Technical Report RADC-TR-77-369, Rome Air Development Center, Griffiss Air Force Base, NY, November 1977.

[9] Richard W. Selby, Adam A. Porter, Doug C. Schmidt, and Jim Berney, "Metric-Driven Analysis and Feedback Systems for Enabling Empirically Guided Software Development", ICSE '91: Proceedings of the 13th international conference on Software engineering, May 1991, pp: 288-298

[10] Richard W. Selby, Adam A. Porter, Doug C. Schmidt, and Jim Berney, "Guide to the CSQA COMMON BODY OF KNOWLEDGE Version 6.2", ICSE '91: Proceedings of the 13th international conference on Software engineering, May 1991, pp: 288-298

[11] J. A. McCall, P. G. Richards, and G. F. Walters. “Factors in Software Quality”, Technical Report Vol. I, NTIS Spring_eld, VA, 1977. NTIS AD/A-049 014.

[12] Soumitra Dutta, Michael Lee, and Luk Van Wassenhove, “Software Engineering in Europe: A Study of Best Practices”, IEEE Software, vol. 1, no. 3, June 1999, pp. 45 – 53.

[13] Richard W. Selby, Adam A. Porter, Doug C. Schmidt, Jim Berney, “Metric-driven analysis and feedback systems for enabling empirically guided software development”, ICSE '91: Proceedings of the 13th international conference on Software engineering, May 1991, pp. 288-298