Engineer Output Assessment Practice Technical Report

xxx, xxx, xxx, xxx, xxx, xxx

xxxxxx Technologies Co., Ltd. & Nanjing University

1 Introduction

In order to ensure the quality of software development and products, the enterprise has adopted many technical tools and best practices, such as static scanning, code review, and so on. However, the software quality at this stage still does not meet expectations, which has become a major pain point for the enterprise. It is not enough to rely solely on technical means to ensure software quality. As participants in the software development process, software engineers have a significant impact on software development and products and services. Effective technical tools can only help engineers, not replace their contributions to the software development process. The enterprise hopes to help engineers and managers identify problems and make further improvements in a timely manner through automated and quantitative engineer output assessment.

The enterprise currently has more than 200,000 employees from more than 100 countries and regions, which makes the output assessment of software engineers face many challenges. After a year of hard work by practitioners and researchers, trial and error in multiple development teams, and improvement, the automated output assessment method suitable for the enterprise was finally completed. At present, the engineer output assessment method has been deployed in the enterprise information system to continuously provide services for engineers and managers. This report details the implementation steps and technical rationale of the output assessment practice.

2 Practice Overview

In order to reinforce the software quality, a number of practitioners and researchers have collaboratively designed the practice. This section describes the concerns and principles in designing the assessment metrics and methods to meet the needs of the enterprise.

2.1 Concerns on Assessment Metrics

The metrics construction process follows the following principles:

Quantification. The design of output metrics must be quantitative. First of all, in the ICT enterprise's environment, it is obviously difficult to qualitatively assess the output of so many engineers, which consumes a lot of time for engineers and managers and brings a series of negative impacts. Second, the results of qualitative assessments also difficult to guarantee objectivity and accuracy, which may lead to disagreements between engineers and assessors. In addition, quantitative assessment can also be automated in the collection of metrics, that is, the system automatically captures the output of engineers, reducing the cost of implementing the assessment.

Wide Coverage. The output of engineers is complex, usually involving many activities in the software development process, and incomplete capture will make the final assessment results inaccurate. Therefore, when defining metrics, it is necessary to cover all the quantifiable outputs of engineers as much as possible. First, the collected output metrics should run through the various processes of the engineer's work, such as development, review, and defect fixing. At the same time, output metrics can reflect the activities of engineers from multiple perspectives, such as quality, productivity, etc.

2.2 Concerns on Assessment Usages

Once output metrics have been identified, engineers' outputs can be captured and reported on in the system on a regular basis. However, this way is not acceptable in this ICT enterprise. First of all, the enterprise has taken into account the comprehensive quantitative outputs as much as possible in the design of the metrics, and it is obviously difficult to understand the display of so many records. Also, since the practice requires real deployment in the enterprise to serve engineers and managers, practicality has to be considered. It is necessary to provide

enterprises with an easy-to-understand and practical assessment method under specific scenarios. The enterprise constructs two assessment dimensions, i.e., cross-monitoring and self-monitoring.

Team-assessment. This assessment dimension is designed to capture the activities and contributions of engineers in their teams during a certain period of time. For some reason, the activities of different engineers have different effects on the team's development process and the quality of the final product. It is not enough to measure the activities of engineers, it makes sense when capturing the activities and contributions of engineers in the context of the team. Therefore, the output assessment of this dimension will provide engineers and managers with quantitative measurement and analysis data in a team scenario.

Individual-assessment. This assessment dimension is designed to capture the activities and changes of engineers throughout the whole development process, that is, how their output has fluctuated as time goes by. The dimension of self-monitoring assessment can provide feedback on personal work status and promote further improvement. By employing self-monitoring assessment, an engineer can answer such questions: Did I write more code lately? Has the quality of the code I wrote improved? In which week did I have relatively unusual output?

3 Implementation Steps

This subsection introduces the implementation processes of assessing software engineer outputs. Figure 1 shows an overview of the processes. First, the enterprise obtains a complete set of output metrics through multiple iterations, then collects the outputs and assesses them in two dimensions according to the metrics set, and finally reports the assessment results to engineers and managers.

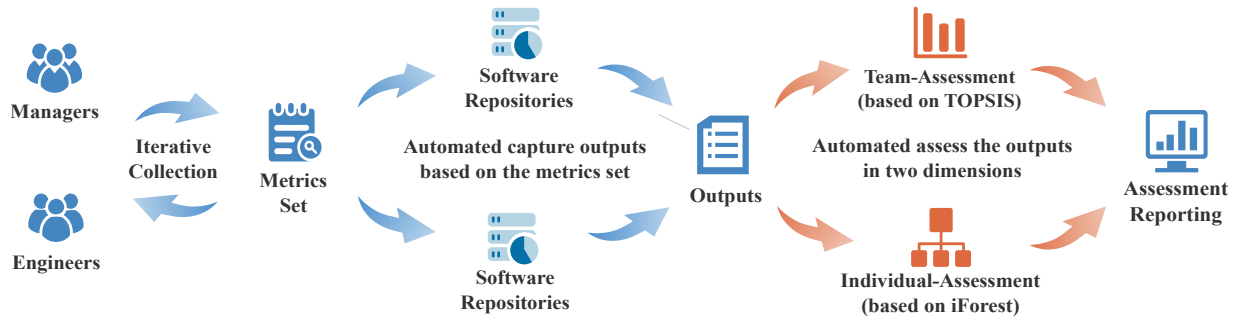

Figure 1: An overview of assessing software engineer outputs

3.1 Metrics Collection

There are many factors to be considered when developing output metrics to reflect an engineer's activities. In order to avoid the time and effort-consuming process of manual participation and minimize the subjectivity introduced by qualitative assessment, output metrics should first be quantitative and then be collected or calculated automatically. Therefore, the metric set is built based on the software repositories of the enterprise and includes as many aspects as possible of software engineer outputs. More than 20 engineers and managers from different divisions familiar with the ICT enterprise's development processes and information systems have collected all measurable metrics related to engineers' activities throughout the development processes.

Figure 2 shows all the quantitative metrics. In addition to basic metrics such as lines of code, the number of requirements, and the number of times for conducting code reviews, the outputs also include quality-related metrics such as code quality and the number of defects. The construction of this quantitative output metric set benefits from the enterprise's original mature and comprehensive data collection system, lots of data have already been collected and recorded in a standardized manner during the software development processes.

3.2 Team-assessment

To conduct this assessment, the enterprise develops team-assessment method based on the *"Technique for Order Preference by Similarity to an Ideal Solution" (TOPSIS)* Hwang and Yoon (1981) method to make full use of the

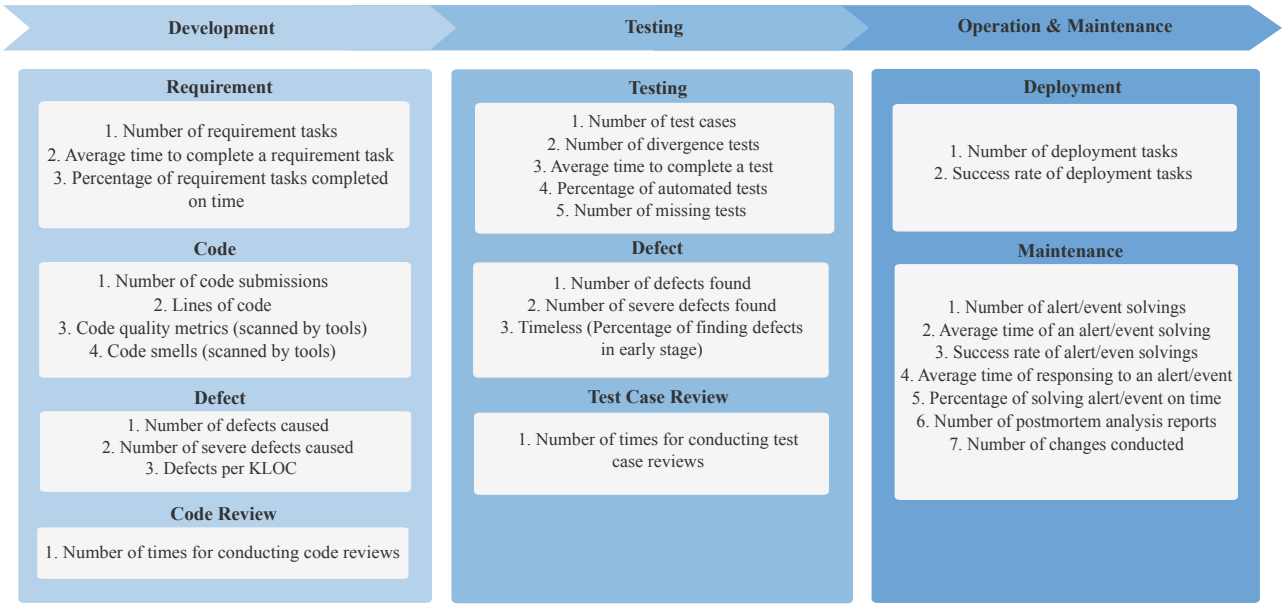

Figure 2: Output metrics

engineers' outputs and distinguish the assessment results. Note that this method is not the only solution, it is just a practice for us to achieve this assessment dimension.

TOPSIS is a ranking method that was developed by Hwang and Yoon in 1981 Hwang and Yoon (1981), and since then has been widely used in conception and application areas. The standard *TOPSIS* method attempts to choose alternatives that simultaneously have the shortest distance from the positive ideal solution and the farthest distance from the negative ideal solution. The positive ideal solution maximizes the benefit criteria and minimizes the cost criteria, whereas the negative ideal solution maximizes the cost criteria and minimizes the benefit criteria Hwang and Yoon (1981); Pavić and Novoselac (2013). *TOPSIS* has been applied in multiple areas such as human resources management, supply chain management, and logistics Behzadian et al. (2012). The following describes the steps of the team-assessment based on the *TOPSIS* method.

a. *Metrics Process: TOPSIS* requires a normalized matrix to calculate the score for each instance. All output metrics are constructed to an original output matrix $A_{m\times n}$, where m is the number of software engineers and n is the number of output metrics.
Then execute the following normalization steps:

$$c_{i,j} = \frac{a_{i,j}}{\sqrt{\sum_{i=1}^{m} a_{i,j}^2}} \tag{1}$$

Where $a_{i,j}$ and $c_{i,j}$ are original and normalized value of output matrix respectively.

b. *Weight Setting:* To customize the method for real-world projects, managers should set their expected weights for each output metric. The enterprise uses the *Analytic Hierarchy Process (AHP)* Golden et al. (1989) to ensure the weight could meet the teams' concerns. *AHP* is an effective tool for dealing with complex decision-making, which could convert the unstructured problem into hierarchical forms of elements and aid the decision-maker to set priorities and make the best decision by reducing complex decisions to a series of pairwise comparisons Saaty and Kearns (1985). Here follow the *AHP* steps to conduct the weight setting process and obtain the weight vector w.

c. *Scores Calculation:* The best and worst values are selected in each output column and constitute the positive ideal instance (c_j^+) and negative ideal instance (c_j^-). Then the distances from both positive ideal instance and negative ideal instance for each engineer is calculated.

$$d_i^+ = \sqrt{\sum_{j=1}^{n} w_j (c_j^+ - c_{i,j})^2} \tag{2}$$

$$d_i^- = \sqrt{\sum_{j=1}^{n} w_j (c_j^- - c_{i,j})^2} \tag{3}$$

Where w_j denotes the weight for metrics j.
At last, the relative closeness coefficient to the ideal solution for each engineer is calculated.

$$s_i = \frac{d_i^-}{d_i^+ + d_i^-} \tag{4}$$

d. *Improvements to the TOPSIS method:* According to the experiments, it was found that the traditional *TOPSIS* method could not reflect the actual scenarios in the development process, that is, the scores obtained by *TOPSIS* were not accurate in some special cases. When some engineers specialize in one work for a period of time, they end up with very high metrics for the work they focus on, while all other output values are very low. In this case, the score calculated by the *TOPSIS* method will be very low. Obviously, it's not appropriate to give such a low assessment to an engineer who has an excellent output in a certain task. Since the *TOPSIS* method combines multiple metrics to weaken the contribution of a certain metric to the score, its incompatibility is reflected here. To solve this problem, *Linear Programming* is introduced in the *TOPSIS* method. A linear programming problem may be defined as the problem of maximizing or minimizing a linear function subject to linear constraints Dantzig (2016); Dantzig and Thapa (2003). It can be clearly seen that the parts under the square root of equations 2 and 3 in the *TOPSIS* implementation can be converted into linear formulas. Taking formula 2 as an example, $(c_j^+ - c_{i,j})^2$ is a constant (represented by t_j), and now the weight is regarded as a variable, the part under its square root can be converted into the standard form of the *Linear Programming* formula:

$$target = w_1t_1 + w_2t_2 + w_3t_3 + ... + w_nt_n \tag{5}$$

Obviously, when the value of Equation 5 is smaller, the outputs of the engineer will eventually be closer to the optimal vector, so that the engineer will get a higher score. Instead of setting a definite weight for each output metric, the managers should set different weight ranges based on importance. In this way, the constraints of *Linear Programming* are as follows:

$$s.t. \begin{cases} x_1 < w_1 < y_1 \\ x_2 < w_2 < y_2 \\ \qquad\qquad ..... \\ x_n < w_n < y_n \\ \sum_{j=1}^{n} w_j = M \end{cases} \tag{6}$$

Where x_1 and y_1 represent the minimum and maximum weights that can be given to output metric 1, x_n and y_n represent the minimum and maximum weights that can be given to output metric n. In addition, M specifies the limit of the sum of all weights, that is, the sum of the weights given to each output metric in *Linear Programming* cannot exceed M. In this way, according to the above constraints, the highest score of an engineer under the limited conditions can be obtained. This method amplifies the contribution of higher output metrics to the overall score to a certain extent, so that an engineer who has a specific work output over a period of time will not be rated too low.
In summary, the enterprise made the following two improvements when using the *TOPSIS* method for output assessment:
- *Set a range for the weight of each output metric, not a fixed value.*
- *Use Linear Programming to obtain the optimal value when calculating the assessment score, constrained by the range of weights and the sum of weights.*

3.3 Individual-assessment

The individual-assessment is designed to capture the activities and changes of engineers throughout the whole development process. The enterprise uses isolation forest Liu et al. (2008) to show output fluctuations and anomalous time points in the time series. Note that this method is not the only solution, it is just a practice for us to achieve this assessment dimension. Here introduce the assessment process based on isolation forest.

Anomalies are data patterns that have different data characteristics from normal instances Liu et al. (2012). They are usually much easier to isolate from the group data. Liu et al. (2008) proposed an anomaly identification algorithm called isolation forest in 2008 and it was proved more accurate and faster than other anomaly identification algorithms. In this algorithm, isolation trees are built by partitioning instances recursively until all instances are isolated or the current height reaches the limit. Anomalous instances with distinguishable attribute values are

more likely to be separated in early partitioning. The average height in an isolation forest which is composed of a set of iTrees (the trees constructed for segmenting outliers, the construction process is shown in Figure 3) will be calculated and contributes to the anomaly scores of each instance.

a. *iForest Construction:* To develop an *iForest*, multiple iTrees should be built by partitioning the attributes randomly selected from the attribute set. Figure 3 shows the training process of an iTree. Continue to perform the division steps shown in the figure until all instances are isolated or the current height reaches the limit. The sooner the instance is divided, the more likely it is to deviate from the group, that is, the orange dots in the figure are more likely to be considered anomalous.

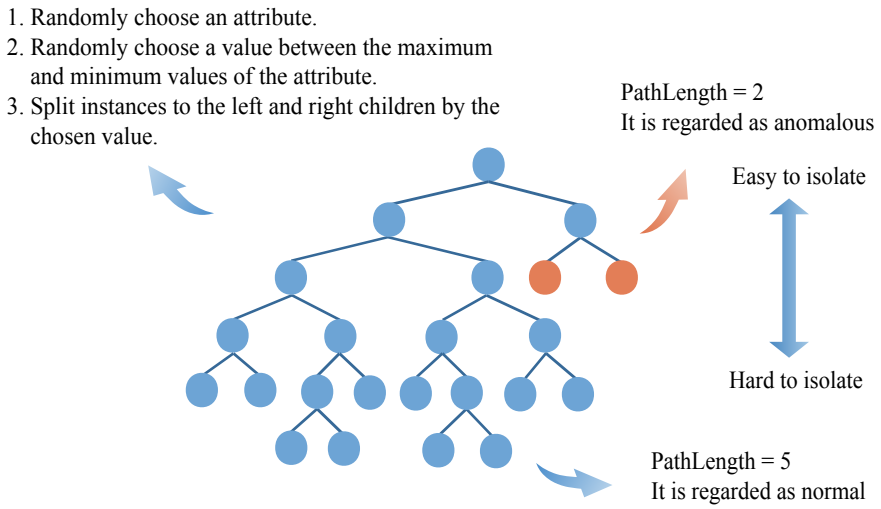

Figure 3: An Overview of Building iTree

b. *Anomaly Analysis:* After training an *iForest*, the isolation score for each time point can be calculated by the following formula:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{7}$$

Where x denotes the instance to be detected, $E(h(x))$ denotes the average height which is obtained when finding instance x on each iTree in *iForest*, and $c(n)$ denotes the average path length of unsuccessful search in an isolation tree. The calculation of $s(x, n)$ provides an isolation score for each instance. The earlier the instance is separated, the smaller its average height is in the forest, that is, the isolation score will be closer to 1, and vice versa.

To reveal the fluctuations for personal outputs, parameter ξ is set to constrain the isolation threshold. It is used to judge if the output at a time point is an anomaly. When analyzing the self output status, an *iForest* is built for each engineer's historical output set and calculated the isolation score at each point. The threshold can determine whether the output at a certain time point is exceptionally better or worse than the other times. The identification process is shown as Algorithm 1.

3.4 Assessment Reporting

The assessment results of the two dimensions will be reported to the relevant engineers and managers in a timely manner. The following major problems in the assessment reports should be considered: *How to display the results more intuitively and let engineers clearly understand their status and shortcomings? What is the best way for managers to understand the output distribution and growth trajectory of team members promptly? How to ensure that personal privacy data is not publicly displayed to take care of engineers' emotions?* In a nutshell, the enterprise aims to help engineers and managers perceive as comprehensive information as possible, that is, to understand the distribution of output and the individual output status of all engineers. The capture of these engineers' outputs will provide measurement support for the ICT company's software process improvement practices.

Algorithm 1: Anomaly analysis

Input: W - weekly output data set of an engineer
Output: scoreSet - isolation score set of the engineer
initiate scoreSet as a null set
build iForest F for data set W
for w *in* W **do**
    $score \leftarrow Score(F, w)$ {score for w in iforest F}
    add score to scoreSet
end
return scoreSet
Function `Score`(F, w)**:**
    Input: F - isolation forest, w - output in a week
    Output: isolation scores
    $sumPath \leftarrow 0$
    for t *in* F **do**
        $sumPath \leftarrow sumPath + pathLength(w, t)$
    end
    $average \leftarrow \frac{sumPath}{F.size}$
    $score \leftarrow 2^{-\frac{average}{c(iTree.size)}}$
    return score
End Function

References

Behzadian, M., Otaghsara, S. K., Yazdani, M., and Ignatius, J. (2012). A state-of-the-art survey of topsis applications. *Expert Systems with applications*, 39(17):13051–13069.

Dantzig, G. (2016). Linear programming and extensions. In *Linear programming and extensions*. Princeton university press.

Dantzig, G. B. and Thapa, M. N. (2003). *Linear programming: Theory and extensions*, volume 2. Springer.

Golden, B. L., Wasil, E. A., and Harker, P. T. (1989). The analytic hierarchy process. *Applications and Studies, Berlin, Heidelberg*, 2.

Hwang, C.-L. and Yoon, K. (1981). Methods for multiple attribute decision making. In *Multiple attribute decision making (MADM)*, pages 58–191. Springer.

Liu, F. T., Ting, K. M., and Zhou, Z. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):1–39.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the 4th International Conference on Data Mining*, pages 413–422. IEEE.

Pavić, Z. and Novoselac, V. (2013). Notes on topsis method. *International Journal of Research in Engineering and Science*, 1(2):5–12.

Saaty, T. L. and Kearns, K. P. (1985). *The Analytic Hierarchy Process*. Analytical Planning.