Analysis of Software Process Effectiveness Based on Orthogonal Defect Classification

LI Zhi-bo\textsuperscript{a}\textsuperscript{*}, HOU Xue-mei\textsuperscript{a}, YU Lei\textsuperscript{a}, DU Zhu-ping\textsuperscript{a}, XU Bing\textsuperscript{a}\textsuperscript{*}

\textsuperscript{a} National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450000, China

Abstract

Measuring the software process effectiveness is the premise to evaluate the quality of software, so it becomes the key of the problem to how to correctly quantify software process effectiveness. In this paper, orthogonal defect classification (ODC) technology is used in the entire process of software development. Analyzing software process effectiveness by ODC attribute measurement method, finally software process improvements can be given from the analysis.

Keywords: ODC(orthogonal defect classification);software process effectiveness; measure.

1. Introduction

Software quality has become the focus of the software field. How to improve software quality, and enhance software reliability becomes the key issues troubled the people.

In recent years, controlling the quality of software by the defect data-centered becomes a research focus. Orthogonal defect classification is between a range of qualitative and quantitative methods of analysis. The basic process is as follows: First, defects are classified according to their different orthogonal property category, and then quantitative analyzed. Finally, the process improvement means can be gained from analysis result.

Currently there are already some studies on the ODC, such as \[1\] the application and implementation of ODC was analyzed and summarized, the literature \[2\] proposed a measure of defects in ODC-based reliability measure for software; the literature \[3\] introduced the basic theoretical concepts and related...
processes. These studies have focused on the basic theory with the ODC and the implementation process, while the ODC process to improve the effectiveness of the software applications do not have too much. This paper describes ODC by theory, analyses the classification of its attributes and the effectiveness of software process by the ODC measure method.

2. Orthogonal Defect Classification

ODC was introduced by IBM TJWatson Research Center, Ram Chillarege, PhD in 1990 [4]. After the ODC concept proposed, the research center also spent 7 years studying and perfecting the technology. Until 1997, the construction of the ODC theoretical system was basically completed. It can help determine and adjust the defect removal strategies, help the establishment and adjustment of defect prevention strategies, guide test design, provide feedback to the developers, help system structure for product assessment, make process analysis and the maintenance cost management. The biggest advantage is not dependent on the process, has the anti-interference ability of normative process. So it is suitable for low process stability, and some process not yet clearly defined, or clearly defined, but implementation has not been standardized Case [5].

2.1. Concepts and Features of ODC

Orthogonal Defect Classification provides a key information extracted from the defect measurement example to evaluate the software development process and make the right process improvement program. ODC attributes used to describe the defects in the eight characteristics: activity of defects found, defect trigger, defects impact, target, defect type, defect qualifier, Defect source, defect age. The eight attributes are pairwise orthogonal, which means there is no correlation, independent and not repeat redundant information between attributes. The purpose is to enable a quantitative classification of defects no ambiguity [6].

2.2. architecture of ODC

2.2.1 Orthogonal defect classification attributes

![Fig. 1. ODC attributes](image)

The detail of ODC attributes can be found in [4]. There are two stages (Figure 1)-defect found stage and defect fixed stage, when collect and classify defect data by ODC. When defect is found, defect exposed environment and the impact of defect on users is obvious. Three attributes can be determined at this time. There are activities of finding defect, defect trigger, and defect impact. When defect is repaired, defect type and defect range can be identified. It can determine the defect in the remaining five attributes: target, defect type, defect qualifier, source and age [7].
Defect activity: The actual implementation of activities when the defect is found. For example, in the functional testing phase, you may do a code review. This phase is functional testing, but the activity is code review.

a. Design Review: review of design or comparison of design and the known demand
b. Code Inspection: Check the code or compare code and the baseline of design.
c. Unit Test: White box testing or execution based on detailed knowledge of the code internals.
d. Function Test: Black box testing based on functional specifications
e. System Test: Testing or execution of the complete system, in the real environment, requiring all resources
g. Review of the graphical user interface (GUI Review): Review of visual interface.

Defect trigger: Describe the environment or conditions of potential defect to be exposed. For example, C language to determine the equivalent symbol "==" was mistakenly written in the assignment symbol "=" during reviewing software code, then the defect trigger attribute shall be taken as language dependency. Because this defect is found when Compiling code related to language examination. Different activities of test have different methods of finding defect, so the mapping between activity and trigger is shown in Table 1.

<table>
<thead>
<tr>
<th>Triggers</th>
<th>Design Conformance</th>
<th>Logic/Flow</th>
<th>Backward Compatibility</th>
<th>Lateral Compatibility</th>
<th>Concurrency</th>
<th>Internal Document</th>
<th>Language Dependency</th>
<th>Side Effect</th>
<th>Rare Situations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Defect impact: If the defect escape, it will be the impact on users. The attribute have thirteen values: Installability, Integrity/Security, Performance, Maintenance, Serviceability, Migration, Documentation, Usability, Standards, Reliability, Requirements, Accessibility and Capability.

Defect target: Description of the high-level entity to be fixed. This attribute has six values: requirements, design, code, build / package, information development, national language support.

Defect type: Describe the actual corrective actions taken. Different defects target correspond to different domain of defect types.

Defect qualifier: Apply to defect type. Indicator capture a non-existent, incorrect or unrelated to the implementation of the elements. The attribute has three values: missing, indicating that the defect is caused by negligence; error, indicating that the defect is caused by the error; or Extraneous, indicating that the defect was due to something not relevant or pertinent to the document or code.

Defect source: Refer to the defect introduced by the source of the most fundamental. Possible sources are: Developed In-House, Reused From Library, outsourced and ported.

Defect age: Describe the defect is caused by newly added code or reuse code. Defect is a part of the product, or the defect is introduced in the current project when you create a new feature, or redesign and / or rewrite the introduction of the old function, or is provided to solve the previous defect solution, which introduce new defect.

2.2.2 Measure of orthogonal defect classification
Orthogonal defect classification based on defects in the application process measurement (Figure 2) always have several steps [8]:

(1) Preparation stage: In-depth understanding of each attribute of ODC, ODC method is adopted to obtain the approval and support, and access to the development team and testing team allowed. Investigate the defect attributes of the current project, and assign the role of ODC to developers and testers. Finally, deploy ODC plan with a defect tracking tool.

(2) Draw up the measure plan: Determine the defect classification attributes, the project is divided into components. Determine the assessment for the ODC project checkpoints, create planning documents and review ODC plans.

(3) The defect data collection and validation: Before input defect data, ensure that all developers and testers have a clear understanding of the meaning of each attribute. In the data entry process, the data format should be control by tools. These procedures should be consistent with defects in the state of transition. After input defect data, it is necessary to complete confirmation.

(4) Classify defect data by ODC: After confirm and validate defect data, the defect data should be classified by ODC.

(5) Process effectiveness assessment: According to the collected defect data, generate a series of statistical analysis of the chart and analyze these charts, so as to get the defect, which is incomplete coding at the software development process.

(6) Software process improvements: Based on the analysis of assessment results, obtain to further improve the project implementation program.

3. Orthogonal Defect Classification in the software process effectiveness analysis

Taking a project process asset library management system for example. The defect property includes the ODC defect of the eight attributes, as well as some general properties of the necessary defects, such as time to find and fix defects, severity and so on.

With different ODC attributes combinations, you can work in many ways to assess the completion of the test. For example, use testing stage and activity attribute to assess whether the defect should be found in a test phase, but the defect was discovered in the next testing phase. Use activity and trigger attributes to assess whether each activity is used in a sufficient number of corresponding triggers to detect defects. Use time and qualifier attributes to assess whether the code is stable and so on.

Assess test validity from the following three aspects of the testing process: the adequacy of testing, whether the means of testing is sufficient complex and product code is complete.

(1) Test adequacy
Activities at all stages of software development process to assess correspondence between the adequacy of testing, through statistical defect activity found in each testing phase to determine whether defect should have discovered in the testing phase, but the defect was left to the next test phase. In order to assess the completion of testing, detect whether exist escape defect in the test.

Figure 3 shows that a considerable proportion of the defect activity discovered in the system testing phase of the project is functional testing. This shows that the defect should be found in functional testing phase, but was left to the system test phase was discovered. It means that testing in the functional testing phase is insufficient, defect was left to be discovered in the system testing phase. Because the later the defect found, the higher the cost of repair. So the defects should be found in functional testing and unit testing are found in the system testing, making the development costs increased substantially. Therefore, test cases should be increased to improve the functional test coverage in functional testing phase. Or modify the exit criteria in the functional testing phase, for example, the number of defects must be found to enter the system testing phase and so on.

(2) Testing whether use sufficiently complex test means

As shown in Figure 3, there is still insufficient in the system functional testing. We can analyze whether use enough complex testing means in the functional testing, namely whether use enough triggers in the functional testing activity.
sufficient in the functional testing activity. We can moderately increase coverage triggered test cases in the functional testing activities of the testing improvement program.

(3) the integrity of the product code

![Fig. 5. Qualifier VS Time](image)

Fig. 5. Qualifier VS Time

Figure 5 shows the trend of the number of missing and incorrect progress with the project changes. For a gradually stable product, the defects of the missing qualifier should be gradually reduced. Because any new code without testing will increase the potential risks.

However, from Figure 5, we can see the stability of this instance is not optimistic. The defects of missing qualifier are not reduced along with the development of project. This shows that the code of the project is still insufficiency, and need to be further improved.

**Conclusion**

Select the appropriate software process measurement method, it is conducive to better evaluate the effectiveness of software process and provide the basis for software process improvement. ODC provides a new software process measurement to improve the measurement of the software development process, and provides a reasonable quantitative standard. It is more important how to choice the ODC measure mode corresponding to the different measure objectives to evaluate the software development process and software quality.

**References**


