Analytic Model for Web Anomalies Classification

Nasser Alaeddine and Jeff Tian
Department of Computer Science and Engineering
Southern Methodist University
Dallas, Texas, USA
Email: nasser.alaeddine@verizon.com
Email: tian@engr.smu.edu

Abstract

In this paper, an analytic technique is proposed to improve the dynamic web application quality and reliability. The technique integrates orthogonal defect classification (ODC), and Markov Chain to classify as well as analyze the collective view of web errors. The error collective view will be built from access logs and defect data. This classification technique will enable viewing the web errors in page, path, and application contexts. This technique will help in developing reliable web applications that benefit from the understanding of web anomalies and past issues. The preliminary results of applying this approach to a case study from telecommunications industry are included to demonstrate its’ viability.

1. Introduction

Analysis of anomalies occurring during operations is an important means of improving the quality of current and future software. Although the benefits of anomaly analysis of operational software are widely recognized, there has been relatively little research on systematic classification of dynamic web application errors. An analysis technique based on orthogonal defect classification for static web pages was proposed to extract information from existing web server logs and classify them based on their response code [1].

This paper presents an integrated technique, utilizing ODC and Markov chain, to analyze the anomalies of dynamic web applications. The collective view of failures collected from server access logs and defect data will be used in the proposed technique. The term “defect” generally refers to some problem with the software, either with its external behavior or with its internal characteristics [2]. The server access log contains a trace of the HTTP processed requests and responses, and is hosted on the web server side. The defect data are the software bugs detected during software development or a system maintenance cycle, and are stored in a centralized repository for tracking. Each defect record corresponds to a unique failure.

The proposed systematic analysis technique will include the construction of the model, and then perform analysis to understand the issues in high-risk areas.

2. Analysis technique strategy

The construction of the model is based on the web server access log, defect data, and software requirement specification. The model states will represent the web application pages, and the transition will represent the navigation aspect of the web application.

Once the defect model is constructed, we apply one-way, two-way, and multi-way analysis to classify the web errors at most-used pages, most-used paths, and application context. The classification scheme is based on the following web errors attributes: defect type, defect frequency, page domain space, file type, referrals, and agent type. The measurement will be error share and the error rate. The error share is the percentage of a given class of errors, and error rate is ratio of errors over hits for a given class of hits.

Model Construction
1. Extract the model states (nodes), and state transition (links) from server access logs. Each node will represent a web page.
2. Assign transition probability, which will be extracted from the server access log or based on domain expert.
3. Assign the number of hits, which will be extracted from the server access log, to each page.
4. Define the domain profile scheme of the top used pages. The domain is defined as, "a subset of the input space over which processing done by the system under test is defined" [3]. The domain profile information will be collected from the product specification.
5. Classify the http errors based on the response code from server logs, and then group the HTTP errors per page. Assign the total number of HTTP errors to the pages within the constructed model.
6. Extract the total number of defects per page from the defect repository along with the corresponding domains. Assign the total number of defect to pages within the constructed model.

**Model Analysis**

1. Determine classes with most errors at application level. Classify the errors from defect data and server access log, and determine the top classes of errors.
2. Determine the most k likely paths (transaction), and the total current defects. Classify these path defects using one-way analysis.
3. Determine the most k likely pages, and the total current defects. Classify these page defects using one-way analysis.
4. Classify the defects of most used pages, most used paths, and at application level using two-way analysis.
5. Classify the defects of most used pages, most used paths, and at application level using multi-way analysis.
6. Conduct reliability analysis per page, path and application.
7. Consolidate analysis results, identify problematic areas, and provide recommendations.

**3. Preliminary Results**

A preliminary classification of the server access logs and the defect data detected during development and maintenance cycles shows that 404 errors are the dominant error. Table 1 shows the top classes of web errors. Also, we noticed that a large number of failures were caused by a small number of errors with high usage frequencies. We found the server access log is very effective in identifying errors that may not be detected during testing or reported during operation.

<table>
<thead>
<tr>
<th>Class of errors</th>
<th>% of total</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP 404 errors (Not Found)</td>
<td>33%</td>
</tr>
<tr>
<td>Interfaces</td>
<td>21%</td>
</tr>
<tr>
<td>Logic, computation, and algorithm</td>
<td>16%</td>
</tr>
<tr>
<td>User interface code</td>
<td>14%</td>
</tr>
</tbody>
</table>

Table 1. Top classes of errors

This method will enable the viewing of errors in relation to the web usage pattern. We believe the page domain scheme will be very effective in detecting the code partitions with most troubled areas.

**4. Conclusion**

In this paper, we developed a web-error classification and analysis method. This systematic defect classification and analysis can identify the problem areas, provide development team with better understanding of the existing web errors, implement effective fixes, and prevent such problems from reoccurring in future releases.

The wide availability of web server logs and defect data makes this approach widely applicable for web applications in operation mode, and our formalized analysis procedure makes it easy to implement. This analysis technique provides a high degree of automation for most operations.

**References**

