Available online at www.sciencedirect.com

ScienceDirect

The Journal of Systems and Software 80 (2007) 795–804

The Journal of Systems and Software

www.elsevier.com/locate/jss

Web error classification and analysis for reliability improvement

Li Ma, Jeff Tian *

Southern Methodist University, Computer Science and Engineering Department, Dallas, Texas 75275, USA

Received 7 June 2006; received in revised form 4 October 2006; accepted 8 October 2006
Available online 16 November 2006

Abstract

In this paper, we adapt an existing defect classification and analysis framework, orthogonal defect classification (ODC), to analyze web errors and identify problematic areas for focused reliability improvement. Based on information extracted from existing web server logs, web errors are classified according to their response code, file type, referrer type, agent type, and observation time. We also introduce an analysis procedure to identify high-risk/high-leverage sub-classes of problems and consolidate analysis results to recommend appropriate followup actions. Results applying our approach to the `www.seas.smu.edu` and `www.kde.org` web sites are included to demonstrate its applicability and effectiveness.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Orthogonal defect classification (ODC); Web errors; Web reliability; Web server logs; Web error classification and analysis

1. Introduction

With the prevalence of the Internet and WWW, web sites are now providing important services to potentially worldwide users. However, the shortened development cycles and constant evolution make it more difficult to assure their quality and reliability (Offutt, 2002). The competition introduced by the open Internet environment and the broader user population also demand an urgent solution to the web quality and reliability problem.

In general, defects are negatively correlated to the quality and reliability of software systems (Card, 1998; Nelson, 1978). To reduce the number of defects and their impact, extensive resources are needed for defect prevention, identification, and resolution (Kan, 2002; Tian, 2005). Given the uneven distribution of defects in software systems, various techniques have been used to identify and characterize defect-prone areas for focused quality improvement (Boehm and Basili, 2001). Orthogonal defect classification (ODC) (Chillarege et al., 1992) is a general framework for software defect classification and analysis that has been successfully used in various traditional software products (Bhandari et al., 1993; Chaar et al., 1993; Chillarege, 1995; Lutz and Mikulski, 2004; Tian and Henshaw, 1994).

The confirmation of the uneven distribution of web errors (Li and Tian, 2003; Ma and Tian, 2003) points to the potential applicability and effectiveness of existing defect classification and risk identification techniques for web applications. In this paper, we adapt ODC to web applications, to identify and characterize problematic areas based on information extracted from existing web logs. We also provide procedural guidance for different analyses and consolidate analysis results to recommend appropriate corrective or preventive actions for web reliability improvement.

In Section 2, we characterize the web environment and common problems, and examine the availability of defect information recorded in existing web logs. Key web error attributes are identified in Section 3. Our general analysis procedure and specific analysis results for the web sites `www.seas.smu.edu` and `www.kde.org` are described in

* Corresponding author. Tel.: +1 214 768 2861; fax: +1 214 768 3085.
E-mail address: tian@engr.smu.edu (J. Tian).
URL: www.engr.smu.edu/~tian (J. Tian).

Section 4. General discussions are included in Section 5. Our overall conclusions are presented in Section 6.

2. Web problems and data availability

We next examine the general characteristics of web problems, relevant information recorded in web server logs and candidate web sites for our study.

2.1. Characterizing web problems and reliability

We can adapt the standard definition of software reliability (Lyu, 1995; Musa et al., 1987) to define the *reliability for web applications* as the probability of failure-free web operation completions, and define the related *web failures* as the inability to obtain and deliver information, such as documents or computational results, requested by web users (Kallepalli and Tian, 2001). Based on these definitions, we can consider the following failure sources:

- *Host, network, or browser failures*: these failures are similar to regular system, network, or software failures, which can be analyzed by existing techniques. Therefore, they are not the focus of our study.
- *Source or content failures* caused by web pages themselves and sometimes the underlying information sources: these failures possess various characteristics unique to the web environment and are the focus of this study.
- *User errors* may also cause problems: although these failures are not the focus of this paper because they are beyond the control of web contents providers, we will encounter some related problems and outline our recommended solutions in Section 4.

The number of observed failures can be normalized by the time interval or usage instances to obtain the failure rate, which also characterizes the reliability of the software (Lyu, 1995; Musa et al., 1987). For web applications, the number of user requests or hits can be used to measure failure rates and reliability for a given web site, because it provides a good characterization of overall web site usage (Kallepalli and Tian, 2001; Tian et al., 2004). For example, if n hits to a web site result in f failures, the failure rate r is then given by $r = f/n$. The reliability R is related to r by the equation $R = 1 - r$ in the Nelson reliability model (Nelson, 1978). Therefore, we directly use failure rate in this paper to characterize web reliability.

2.2. Web access logs and their contents

A "hit" is registered in the access log if a file corresponding to an HTML page, a document, or other web content is explicitly requested, or if some embedded content is implicitly requested or activated. Most web servers record relevant information about individual accesses in their access logs. Despite minor variations in log format, the following fields are always included:

- The "time" when the request is processed by the server.
- "Requested URL", or the URL of the destination file requested by the "hit".
- "Response code" that provides status information of the "hit".
- "Referring URL", or the web page that provided the link followed by the "hit" to the currently requested file.
- The "agent" used by the user to access the web site. It includes the client operating system and browser, or the application identification.

Common response codes are listed in Table 1 (for complete definitions, see www.w3c.org/Protocols/). Response code "2xx" indicates successful request; "3xx" indicates that redirection is needed; "4xx" indicates bad request; and "5xx" indicates server error. Bad requests conform closely to the source or content failures we defined above when they are triggered by embedded links. These problems and related user problems are discussed further in Sections 3.5 and 4.4.

In case of failures, or those with response code 4xx or 5xx, the specific response code represents different types of problems experienced by users. In subsequent discussions, we refer to these failures as *errors* of corresponding *error types*, to conform to the commonly used terminology in the web community. The information associated with access log entries is used to analyze and classify web errors in this paper. If additional information about web errors is available, such as recorded in various error logs and some platform/language/technology-specific logging tools, it can also be used in our analyses, such as in (Li and Tian, 2003).

Various analysis tools are available, such as FastStats (www.mach5.com/fast), Analog (www.analog.cx), Webalizer (www.mrunix.net/webalizer) and various web analytics tools (see www.webanalyticsassociation.org). However, these tools focus on web traffic instead of problems in web contents, and problem

Table 1
Common response codes

Response code	Description
200	OK
206	Partial content
301	Page moved permanently
302	Page moved temporarily
304	Resource modified
400	Syntax error
401	Lack of authentication data
403	Access is forbidden
404	File does not exist
405	Method not allowed
408	Request time out
414	Requested URL too large
416	Requested range too large
500	Server internal error

classification beyond what is present in the raw data is left to the user. Therefore, our web error classification and analysis supported by our own utility programs go far beyond the basic analyses supported by existing tools.

2.3. Web sites for our case studies

Our approach for web error classification and analysis was applied to two web sites from different application domains. One is `www.seas.smu.edu`, the official web site for the School of Engineering and Applied Science at Southern Methodist University (SMU/SEAS). This web site utilizes Apache Web Server (Behlandorf, 1996), a popular choice among many web hosts, and shares many common characteristics of web sites for educational institutions. These features make our results meaningful to this general environment.

The second web site is `www.kde.org` for the open source KDE project. This web site is different from the SMU/SEAS web site in terms of traffic volume, maintenance cycle, application domain and other measurable characteristics (Tian et al., 2004). All these factors make it a good choice to cross-validate our results. The KDE web site also uses Apache Web Server and the same data format, which makes our data extraction and analysis easy.

We obtained access log data covering over a year in 1999–2000 for SMU/SEAS and several months in 2003–2004 for KDE. However, data stability problems due to product releases and significant changes (Kallepalli and Tian, 2001; Tian et al., 2004) reduced our data sets to consistent and stable data covering 26 consecutive days for SMU/SEAS and 22 days for KDE.

3. Identifying web error attributes

In this section, we adapt ODC (orthogonal defect classification) (Chillarege et al., 1992) to the web environment.

3.1. Basic ideas of ODC

Analysis of defect data can help us detect and remove potential defects, prevent injection of similar defects in subsequent projects, and manage risk better by planning early for product support and services (Boehm and Basili, 2001; Tian, 2005). The defect analysis techniques form a broad spectrum (Chillarege, 1995; Kan, 2002): statistical defect models, such as various software reliability models (Lyu, 1995; Musa et al., 1987), can predict defect count, trend, or product reliability. Alternatively, qualitative analyses of defects, such as causal analysis and related defect prevention process (Card, 1998; Mays et al., 1990), examine the logical link between defects and their causes in an attempt to prevent similar defects by eliminating the identified causes.

There is a wide gap between statistical defect models and qualitative defect analyses. Orthogonal defect classification (ODC) bridges this gap with a systematic classification and analysis of defect data (Chillarege et al., 1992). ODC has a rich and extensive category of defect attributes, so that related data and analysis results can be used to provide valuable insight and in-process feedback to the software development and maintenance processes (Chillarege, 1995). Key ODC attributes include:

- Defect *impact*, or failure type, indicates the kind of problem caused by the defect, such as capability, performance, usability, or reliability problems.
- Defect *type*, or fault type, represents the type of internal problem or fix applied, such as timing, interface, or algorithm problems.
- Defect *source* indicates the general location of code that is corrected.
- Defect *trigger* indicates what facilitated the software fault to surface and result in a failure, such as via unit testing, system testing, inspection, etc.
- Defect *severity* is based on a combination of the difficulty caused by the defect and the urgency for a fix.
- *Phase found* reflects the development phase when a defect is found.

Defect analyses in ODC typically include: (1) one-way analysis that examines one attribute at a time for its overall distribution or its trend over development phases; and (2) two-way analysis that examines the cross-interaction of two attributes. Higher order analyses are also possible, such as using tree-based modeling on all the ODC attributes (Tian and Henshaw, 1994).

One fundamental assumption in all ODC analyses is that there exists an expected defect profile, so that the actual results are compared it to identify anomalies by their ranked differences (Bhandari et al., 1993). These anomalies are analyzed and corroborated by development personnel to initiate appropriate corrective or preventive actions. However, under many application environments, such a defect profile does not exist. Uniform distribution is suggested as the starting point, and an expected profile can be gradually constructed for future use (Chaar et al., 1993).

3.2. Identifying generic web error attributes

Inspired by ODC, we next identify web error attributes while considering the specific characteristics of web applications and information availability from web access logs:

- *Response code*: the "response code" in access logs tells us whether a given request resulted in a success or failure. In case of a failure, the response code provides information about the specific problem experienced by the user, or the *impact* of the failure to the user. As we observed in our previous studies (Kallepalli and Tian, 2001; Li and Tian, 2003) and in Section 3.3, "missing files" dominate all other failure types. The corresponding *interfaces* need to be fixed, either by providing the missing destination file or modifying the file making the request. Therefore, this attribute also gives us information of the defect *type* or fix type.

- *File type* of the "missing file": if a requested file is missing, it can be considered as a defect *source* and characterized by its file type. For some web sites, it can be further characterized by its owner type, directory level and other location information. All of them can be derived from the "requested file" field of the access log. In addition, defect *severity* corresponds to the importance of the "missing file". This needs to be individually determined, although file type and location may also provide some clues.
- *Referrer type* and *agent type*: for the web environment, all failure observations recorded in access logs are *triggered* through operational use or specific requests. Each request can be characterized by its *referrer* and *agent*:
  - *Referrer type* is a classification of the referring URL concerning its relation with the web site under study. It tells us if an internal, external, or other source provided the address of the requested file.
  - *Agent type* represents the type of software agent the client used.

  The analysis of referrer type and agent type can help us distinguish problems triggered by different users and their usage patterns.
- *Time period* an error was found: although all errors recorded in access logs were *found* in the field rather than in the development phases, the associated timestamp values can be used to group them into different time periods to provide useful information.

All these attributes can be extracted from web access logs, either directly, as for *response code*, or through some simple processing for all other attributes. We have developed utility programs written in PERL to automate this activity. We next individually examine these web error attributes, their possible values and the computation required to extract them from web access logs.

3.3. Web error attribute: response code

Table 2 shows the distribution of response codes for the SMU/SEAS and KDE web sites directly extracted from their access logs. "File does not exist" errors, or "broken links", with response code 404, accounted for about 91.6% of the total errors for SMU/SEAS and 99.2% for KDE, which represent 3.8% and 5.8% of the total accesses, respectively. All other error types either account for a negligible share, or are closely related to security or other problems instead of the reliability problems we focus on in this study. The surveys by the Graphics, Visualization and Usability Center of Georgia Institute of Technology also found that problems with "broken links" are the problem type most frequently cited by web users, next only to network speed problems (see www.gvu.gatech.edu/user_surveys/). Therefore, further study in this paper will focus on "404 errors", the most dominant type of recorded errors, and the most observed web content problems for the general population of web users.

3.4. Web error attribute: file type

Different file types provide different information for users, serve different functions and may cause different problem to users when they are missing. For most web sites, HTML *pages* are the main information sources and provide the main navigation facilities. If a page is missing, users will not only miss the information in the page, but also encounter navigation problems due to the missing hyperlinks. Other files, such as *graphic, multimedia* and *document* files, do not usually contain navigation facilities.

Table 2
Error distribution by response codes

Response code	Description	SMU/SEAS		KDE	
		Frequency	%	Frequency	%
200	OK	526,577	69	8,026,896	80
206	Partial content	4,476	1	9,241	0
301	Page moved permanently	70,396	9	12,754	0
302	Page moved temporarily	391	0	296,865	3
304	Resource modified	125,121	17	1,083,607	11
400	Syntax error	158	0	408	0
401	Lack of authentication data	108	0	0	0
403	Access is forbidden	1,953	0	642	0
404	File does not exist	28,650	4	583,315	6
405	Method not allowed	3	0	12	0
408	Request time out	417	0	3,489	0
414	Requested URL too large	0	0	8	0
416	Requested range too large	0	0	3	0
500	Server internal error	0	0	212	0
All		758,250	100	10,017,452	100

Therefore, the impact of such a missing file is limited to the specific file alone.

A file's "type" is identified by or closely associated with its suffix in most modern operating systems. However, there are more than a hundred or more different file suffixes, even for a moderately-sized web site such as www.seas.smu.edu, (Li and Tian, 2003). We need to reduce them into a few meaningful and manageable types by grouping files of similar contents and functions into a single *file type*, resulting in the following categories:

- *Page* type includes files with suffix ".html", ".htm" and other variations, or directories that correspond to their index pages. For some web sites, dynamically generated pages may be considered separately.
- *Graph* type includes graphic files, e.g., files with suffix ".gif", ".jpg", etc. Dedicated graphic files can also be considered separately from embedded ones.
- *Document* type includes files with suffix ".txt", ".pdf", ".ps", etc., which usually provides plain information but not navigation facilities.
- *Multimedia* file type includes audio and video files in different formats.
- Files not belonging to the above categories are classified into the *other* type.

3.5. Web error attributes: referrer type and agent type

In the access log entry corresponding to a user access to page p of site S through a hyperlink provided in page q, q's address is recorded as the "referring URL". Based on q's relationship to S, we can classify it as an *internal*, *external*, or *empty* referrer:

- Page q may belong to S, indicating that the user navigated inside S, resulting in q's classification as an *internal* referrer.
- Page q may belong to another web site T, indicating that the user jumped from T to S, resulting in q's classification as an *external* referrer.
- *Empty* referrer type refers to the case where the referrer URL is not available, with a " – " recorded for its "referral URL" field. User accesses by typed addresses or from bookmarks, as well as accesses from web robots, all belong to the *empty* referrer type.

There are two primary types of agents that can be directly extracted from the "agent" field of access logs: browsers and robots. *Browser* agents are usually used by human users to browse the web. *Robot* agents, such as search engine robots, off-line browsers, link-checkers, etc., are programs that automatically retrieve files from web servers.

3.6. Web error attribute: time period

Requests in different time periods may be subjected to different traffic volumes and associated with different types of users or usage patterns. For example, a web site focused on localized users, such as www.seas.smu.edu, may experience a significantly lowered traffic volume during late night, while a web site of international focus, such as www.kde.org, may not have such big traffic volume variations (Tian et al., 2004). The traffic volume has a great impact on the performance and reliability of the web site. Sometimes, a performance problem in the developers' point of view may well be a reliability problem from the users' perspective. The analysis of the *time period* attribute can yield useful information for problem classification and reliability improvement.

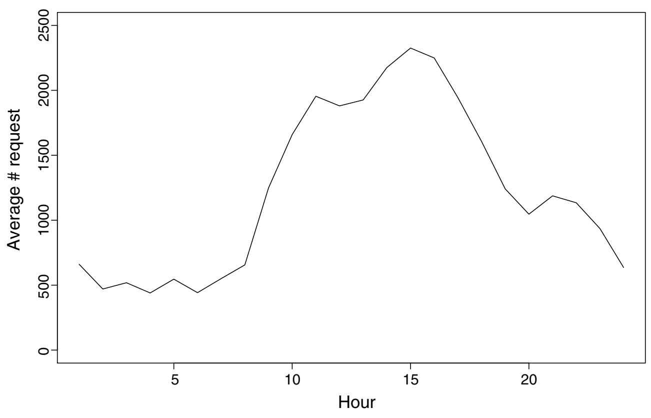

Fig. 1. Average number of hourly requests in a day for SMU/SEAS.

The time when the server finished processing a request or when the error was detected for a failed request is recorded in access logs. The web error attribute *time period* can be easily extracted from these logs by grouping entries accordingly. For example, Fig. 1 shows the hourly accesses for the SMU/SEAS web site, averaging over the 26 days (Kallepalli and Tian, 2001). Based on this distribution, two time periods can be identified: from 7:00 a.m. to midnight as *day time* and from midnight to 7:00 a.m. as *night time*.

Other classification schemes are also possible to accommodate special needs. For example, classification by peak, normal and low traffic hours may be a good choice for web sites more sensitive to traffic volume variations. Classification by weekdays and weekends is another choice for web sites with substantial differences between these time periods.

4. Web error classification and analysis

Using the above classification scheme, web errors can be classified and analyzed to identify problematic areas and to provide feedback and recommendations.

4.1. General procedure and activities

Our web error analysis procedure expands one-way and two-way analyses from the original ODC (Chillarege et al.,

1992) into a general procedure consisting of the following five stages:

(1) In the *pre-classification* stage, preliminary analysis of the given web site and its log data is carried out to define error measurements, to establish an analysis baseline and to customize the classification scheme.
(2) One-way analysis examines one attribute at a time to identify problematic areas for corrections or for further analyses. Detailed error trend and reliability analyses could also be performed, if necessary, by combining time-stamped error data with detailed usage measurements (Kallepalli and Tian, 2001; Tian et al., 2004).
(3) Two-way analysis examines the interaction between two error attributes to address the problems identified but not clarified in the analyses above.
(4) Multi-way analysis focuses on the remaining problems by the combination of three or more attributes.
(5) The above analysis results are *consolidated* to provide feedback and recommendations.

Some analysis stages can be skipped when the previous stages have already provided all the information we need or have clearly explained the problems identified. We next describe these five stages individually and present our analysis results for the SMU/SEAS and KDE web sites.

4.2. Stage 1a: defining measurements and baseline

When web accesses are classified, the number of hits and the number of errors in each category can be counted. Two error measurements can be derived from these basic counts:

- Error share (ES%) – This is the percentage of a given class of errors. A higher ES% indicates a greater leverage on the overall reliability.
- Error rate (ER%) – This is the ratio of errors over hits for a given class of hits, or the failure rate linked to reliability via the Nelson model (Nelson, 1978) in Section 2.1. A higher ER% indicates lower reliability.

These measurements can be used to characterize the quality for each subset defined by specific web error attribute values. Because there is no pre-existing expected error profile for web applications to serve as the baseline for comparison, we use uniform distribution as the baseline for web error analysis. This would imply that the subsets with extremely high ES% or ER% will be identified as anomalies regardless of whether they are expected or not. Once enough classified web error data are accumulated, an expected error profile for a given web site or for similar web sites can be constructed. Such profiles can be used as a more realistic and useful baseline to help with result interpretation and derivation of corrective actions. For example, only those with "unexpectedly" high ES% or ER% are identified as anomalies for corrective actions. These profiles can also be generalized to help the defect prevention for new web sites or new web services.

4.3. Stage 1b: customizing classification

As discussed in Section 3.3, our analyses focus on the dominant "404 errors" only. The *response code* attribute was not analyzed further and excluded from the lists of attributes in Table 3 below.

For both of our web sites, a few individual files, or *exceptions*, were responsible for the extremely high error rates and the high error share of the categories they belong to. For example, for the SMU/SEAS web site, a single "`.ico`" file and a few "`.mp3`" files contributed to the extremely high error rates of 100% and 73% for "`.ico`" and "`.mp3`" files, respectively; and a few "`.class`" files and a single graphic file, respectively, accounted for 17% and 21% of the total errors. Similarly for the KDE web site, a few "`.ico`" files accounted for about 58% of the total errors, and a single file "`kdenews.rdf`" and its variations accounted for more than 31%. Errors associated with all exceptions accounted for about 51% of the total errors for the SMU/SEAS web site and 91% for KDE.

These exceptions are usually associated with extremely high usage frequencies, which explains their strong influence on the overall reliability. They would also dominate and significantly skew the error analysis results. Therefore, they need to be excluded from the classification scheme and subsequent analyses. On the other hand, the identification of these exceptions ensures that they are dealt with. Therefore, excluding them in subsequent stages will not diminish the valuable information provided by them in this stage.

When used to analyze a given web site, the specific error attributes and particularly their possible values may vary slightly, which should be customized in this pre-classification stage. For example, *multimedia* files were rare for the SMU/SEAS web site, especially after excluding all the "`.mp3`" files under a single directory due to their identification as exceptions. Therefore, the rare *multimedia* file type is subsumed under the *other* file type in Table 3. There is also a clear distinction between *official* and *personal* web pages for this web site. The former category is maintained by dedicated professionals and contains "official"

Table 3
Customized error attributes and their possible values for SMU/SEAS and KDE

Attribute	Web site	Possible values
File type	SMU/SEAS	Page, graph, document, other
	KDE	Static page, dynamic page, graph, style, other
Referrer type	Both	Internal, external, empty
Owner type	SMU/SEAS	Official, personal
Time period	Both	Day, night
Agent type	Both	Browser, robot

information about the school and its academic units; while the latter category contains self-maintained personal web pages. Therefore, a *owner type* attribute is introduced in Table 3 to reflect these differences.

For the KDE web site, there were considerable amount of dynamic contents, so the *page* type was divided into *static page* and *dynamic page* types in Table 3. In addition, files with suffix "`.css`" contributed a considerable amount of accesses and represented important characteristics of the KDE web site. Therefore, a new *style* file type was also introduced in Table 3.

4.4. Stage 2: one-way analysis

Our most interesting one-way analysis results are by *referrer type* presented in Table 4. Because *internal* referrers are the responsibility of the local contents providers and *external* referrers are usually beyond their control, we should focus on fixing *internal* referrer problems, particularly if they represent a significant share of the problems or have a relatively high error rate. Fortunately for both the SMU/SEAS and KDE web sites, the error rates for *internal* referrers are lower than that for *external* ones, indicating that they do not have a severe quality problem internally.

For the SMU/SEAS web site, *empty* referrer type led to a higher error rate with a considerable amount of errors. Consequently, it is identified as a problematic area and will be analyzed further in subsequent analyses.

For the KDE web site, although *internal* referrer type contributed about half of the errors, its error rate is considerably lower than *external* and *empty* referrer types. Compared with the SMU/SEAS web site, the KDE web site also has a significantly better reliability with *internal* referrers. Such evidence for the superior inherent quality of the KDE web site, combined with the fact that 91% of the errors have already been identified and explained in the pre-classification analysis earlier, makes it unnecessary to perform further error analyses. In effect, we can skip the rest analyses for KDE and jump directly to the final result consolidation stage.

Another interesting one-way analysis is by *file type* for the SMU/SEAS web site presented in Table 5. The *other* files have the highest error rate but the smallest error share, so they can be dealt with individually. The *page* files contributed the majority of errors and had a fairly high error rate, indicating their significant impact on the overall reliability. Combined with the importance of *page* files as the backbone of the SMU/SEAS web site, we need to analyze this file type further in subsequent analyses.

Table 4
1-way error analysis by *referrer type*

Referrer type	SMU/SEAS		KDE	
	Error share (%)	Error rate (%)	Error share (%)	Error rate (%)
Internal	60.39	1.41	49.18	0.51
External	10.02	1.99	24.74	4.35
Empty	29.59	3.75	26.08	1.39

Table 5
1-way error analysis by *file type* for SMU/SEAS

File type	Error share (%)	Error rate (%)
Page	61.06	3.80
Graph	33.84	0.90
Document	3.54	2.51
Other	1.56	10.16

4.5. Stage 3: two-way analysis

Two-way analysis can be done either through the general joint distribution analysis or the following focused conditional analyses, similar to conditional probability analysis: for a problem characterized by attribute A with value v in a one-way analysis, two-way analysis can be applied as the conditional analysis of other attributes under the condition that attribute A takes value v. By such conditional analyses, more information associated with the original problem is exposed, possible causes can often be identified, leading to additional suggestions or recommendations.

For the SMU/SEAS web site, one of the open issues left from the above one-way analyses is the further characterization of *page* files, which is analyzed in this stage by *owner type* in Table 6 through our two-way conditional analysis. We can see that the *personal* pages has a significantly higher error rate and error share than the *official* pages. These results agree with our general expectation of better reliability for the more important *official* part.

Another two-way analysis by the combination of *file type* and *referrer type* focuses on both the *page* file problems and *empty* referrer problems identified in the one-way analyses above. Table 7 compares the result for *empty* referrers with that of *internal* referrers used as the baseline, and shows that both referrer types are comparable in both *page* error shares and *page* error rates. In this case, the result needs to be analyzed further to clarify the similarity via three-way analysis below.

4.6. Stage 4: multi-way analysis

Multi-way analysis can also use conditional analysis as we did in two-way analysis, with the conditions as a set

Table 6
2-way error analysis of *page* files by *owner type* for SMU/SEAS

File type (attribute 1)	Owner type (attribute 2)			
	Official		Personal	
	Error share (%)	Error rate (%)	Error share (%)	Error rate (%)
Page	25.55	2.03	74.45	5.15

Table 7
2-way error analysis of *page* files by *referrer type* for SMU/SEAS

File type (attribute 1)	Referrer type (attribute 2)			
	Internal		Empty	
	Error share (%)	Error rate (%)	Error share (%)	Error rate (%)
Page	50.39	4.41	49.61	4.17

of attribute values characterizing the data subsets. However, this solution is feasible only when there are only a few remaining problems, because combinatorial explosion of the multiple attributes would require too many multi-way analyses. To avoid combinatorial explosion in general, tree-based modeling can be used to analyze the problem further by establishing predictive relations between multiple predictor variables and an identified response variable (Clark and Pregibon, 1993).

After two-way analysis, the only remaining issue at the SMU/SEAS web site is the further characterization of problems associated with *page* file type and *empty* referrers (as compared to *internal* referrers) in Table 7. Table 8 lists the three-way analysis results of the classification of *page* files by the combination of *referrer type* and *owner type*. We can see that for the *official* pages, *internal* referrers trigger a small number of errors with a fairly low error rate; but the *empty* referrers trigger a large portion of errors with a higher error rate. In other words, pages of the *official* part have a fairly good quality, while most of the errors therein are triggered by *empty* referrers. For *personal* pages, more errors were triggered by *internal* referrers, and the error rate of *internal* referrers is even higher than *empty* referrers. We can conclude that the high error rate in *personal* pages is largely due to their own poor quality.

4.7. Stage 5: result consolidation and recommendations

The results for both our case studies are consolidated and summarized below, along with our recommendations for reliability improvement:

- In both case studies, a large number of errors were caused by a small number of files, or *exceptions*, with high usage frequencies. By fixing these few exceptions, the total errors could be reduced by about 51% for SMU/SEAS and 91% for KDE.

  Recommendations: (1) Identification of exceptions must be a mandatory stage in the web error classification and analysis procedure due to their strong influence on overall reliability. (2) Errors occurring with high frequencies should received higher priorities for fixing for effective reliability improvement. (3) Web developers and maintainers must be careful in making changes to the locations and names of frequently updated files that are expected to be requested frequently by web robots, such as "`.rdf`" files for the KDE web site. (4) As a preventive action, files with high usage frequencies should be identified early to allow project managers to plan ahead for reliability assurance and improvement. Usage profiles can be established based on existing web logs to identify high-usage files (Kallepalli and Tian, 2001; Tian and Ma, 2006).
- Empty referrer accesses to SMU/SEAS experienced lower reliability, largely due to the wrong or outdated addresses requested by users or web robots. Many of these requests are closely associated with user initiated operations, such as from user typed URLs, saved bookmarks, or start-ups.

  Recommendations: Although web contents providers couldn't control the errors triggered by *empty* referrers, it is imperative to address some of these problems closely related to user initiated operations in order to maximize user satisfaction. Some preventive measurements, such as redirect facilities and informative error pages, can alleviate the problem impact. More actively, controlling the renaming or relocation of entry pages can reduce the chance of such errors. As a preventive measure for future web-application development, adoption of development standards, naming conventions, proper training and use of development tools, would help us create more robust and reliable web pages. These actions could improve the overall user experience and lead to better user satisfaction.
- The comparison of the error patterns of SMU/SEAS with KDE showed that the overall reliability of the latter is better than the former after the exceptions are excluded. From the analyses of SMU/SEAS, we observed that the overall error rate of the *personal* pages is higher than that of the *official* ones. The error rate of page files in the *personal* part triggered by *internal* referrers is even higher, which indicates inherent poor quality for this part. The different web sites, such as SMU/SEAS vs. KDE, or sub-sites, such as *personal* vs. *official* parts of SMU/SEAS, are likely to be different in many aspects, such as developers' skills and experience, development methods and tools, maintenance cycles and site contents. All these factors may contribute to reliability variations.

  Recommendations: Since different development settings may have a strong influence on overall reliability, further studies are needed to assess the effect of these

Table 8
3-way error analysis of *page* files by *owner type* and *referrer type* for SMU/SEAS

File type (attribute 1)	Owner type (attribute 2)	Referrer type (attribute 3)			
		Internal		Empty	
		Error share (%)	Error rate (%)	Error share (%)	Error rate (%)
Page	Official	28.6	1.19	71.4	3.2
	Personal	62.1	5.37	37.9	4.59

factors both qualitatively and quantitatively. Causal analyses of the differences can reveal good development practices as well as pitfalls. Quantitative impact assessments would require a large amount of empirical data to help us develop empirical models that predict reliability from development settings. Results from these future studies can guide web contents providers to choose successful methods and avoid pitfalls and to achieve quantifiable reliability improvement.

5. Discussions

The widely available access logs make our approach applicable to most web sites. The use of such field data incorporating both the problem and usage information in reliability assessment reflects the realistic quality expectations from the users' perspective. Identification and correction of the related problems would lead to effective reliability improvement meaningful to web users. Our approach goes beyond the basic analyses based on raw data supported by existing web analytics tools by creating an individualized classification scheme for individual web sites so that problematic areas can be identified for focused reliability improvement. Our formalized analysis procedure also makes our approach easy to use.

We developed a 5-stage analysis process to support more systematic and flexible exploration of available data. Although certain stages can be skipped if previous stages have already clarified the problems, we did not specify an explicit rule for doing so because the decision may depend on the specific problems and the characteristics of the web site under study. For example, the pre-classification analysis of exceptions for the KDE web site identified and explained 91% of the errors. No further analyses would be needed if the resulting reliability after dealing with the exceptions would be adequate. Otherwise, additional stages can be carried out, such as in our one-way analysis in Section 4. The one-way analysis results clearly indicated that the remaining problems for KDE were primarily triggered by *external* and *empty* referrers. The subsequent analyses can be skipped, leading us right to the final consolidation stage.

We also achieved a balance between the need for risk identification and the desire for robust analysis results due to the exclusion of exceptions. We do not simply throw away the exceptions, but rather use them to identify and resolve specific problems and help shaping our classification scheme. Our analysis procedure ensures that all important exceptions are identified, analyzed, understood and acted upon, before we proceed with subsequent analyses excluding them. In each of our analysis stages, we try to identify anomalies or areas that stand out, similar to identifying exceptions in the pre-classification stage, until we have achieved satisfactory understanding of all the results or until we have exhausted the available data.

Direct users of our analysis results are the web development and maintenance personnel. In our case studies, we made our analysis results available to them as post-mortem feedback. In future studies, we plan to initiate more active engagement by holding group discussions with web developers and jointly performing causal analysis (Card, 1998; Mays et al., 1990) on identified problematic areas to identify key opportunities for improving website reliability. We also plan to follow through with such corrective or preventive actions and quantitatively assess their impact on reliability improvement. These activities will, in turn, help us fine tune our approach proposed in this paper for better efficiency and effective in future applications.

There were a few surprises in developing and applying our approach, which led to it current form. The biggest surprise was in the overwhelming impact of the selected few files, or exceptions, on our web error analysis results when we first attempted one-way analysis for SMU/SEAS without excluding such exceptions. This led us to introduce our exception identification in our pre-analysis stage. This worked well for the KDE web site, as shown in the cross-validation study. In fact, exception identification had an even stronger impact on web error analysis for KDE. When combined with only one-way analysis, it can fully identify and explain the problematic areas.

Another significant change to our approach is the introduction of customization of our classification scheme and related categories for individual web error attributes. Our generic web error attributes in Section 3 cover all the fields of the standard web access logs. However, further information, such as error location, referral and timing, could be extracted through additional data processing. For example, besides file type, we can extract owner type, directory level and other information for web error location analysis. Among them, the owner type analysis played an important role in identifying and explaining the problematic areas for SMU/SEAS. Consequently, we recommended the addition of classification scheme customization in our pre-analysis stage, which should be done in consultation with domain experts and/or web site owners. This customization can also be done as a response to feedback from analyses exploring the use of additional web error attributes and categories, such as owner type and directory level mentioned above.

On the other hand, some specific web error attributes, such as time period and agent type, did not provide us with very useful information for problematic area identification. However, several of the identified exceptions are actually related to agent types, such as the "`.ico`" files for our two web sites. Time period could also provide more meaningful analysis results for dynamic web sites with heavier and more varied traffic. As a whole, the collection of our generic web error attributes in Section 3.2, with the addition of attribute and category customization in Section 4.3, worked well with our analysis procedure to identify and explain all the important problematic areas and offer recommendations.

6. Conclusions and perspectives

In this paper we developed a web error classification and analysis method by adapting ODC (orthogonal defect classification) (Chillarege et al., 1992) to the web environment. Web errors are classified accordingly and high-risk areas are identified and analyzed for effective reliability improvement. The wide availability of web server logs makes our approach widely applicable, and our formalized analysis procedure makes it easy to implement.

We have successfully applied our web error classification and analysis method to two case studies from different application domains, and the results demonstrated the applicability and effectiveness of our approach. The high-risk areas we identified from these case studies are associated with files of high usage frequencies and specific file types, usage sequences and development settings. We recommended reliability improvement activities accordingly.

Although our analyses are based on the default access logs of web servers for the two web sites of our case studies, where errors concerning execution of dynamic web logic are not recorded, our approach can be extended to analyze dynamic error logs by customizing relevant error attributes. We also plan to apply our method to web sites in other domains, such as commercial web sites with large volumed of dynamic contents, more computational capabilities and two-way traffic, to further extend and validate our approach. Such new applications will also help us establish expected error profiles. These profiles and our related experience can then be generalized to ensure and improve reliability and customer satisfaction for many web applications.

Acknowledgments

This research is supported in part by NSF Grant CCR-0204345 and THECB/ATP Grant 003613-0030-2001. We thank the anonymous reviewers for their constructive comments and suggestions.

References

Behlandorf, B., 1996. Running a Perfect Web Site with Apache, second ed. MacMillan Computer Publishing, New York.

Bhandari, I., Halliday, M., Tarver, E., Brown, D., Chaar, J., Chillarege, R., 1993. A case study of software process improvement during development. IEEE Transactions on Software Engineering 19 (12), 1157–1170.

Boehm, B., Basili, V.R., 2001. Software defect reduction top 10 list. IEEE Computer 34 (1), 135–137.

Card, D.N., 1998. Learning from our mistakes with defect causal analysis. IEEE Software 15 (1), 56–63.

Chaar, J., Halliday, M., Bhandari, I., Chillarege, R., 1993. In-process evaluation for software inspection and test. IEEE Transactions on Software Engineering 19 (11), 1055–1070.

Chillarege, R., 1995. Orthogonal defect classification. In: Lyu, M.R. (Ed.), Handbook of Software Reliability Engineering. McGraw-Hill, New York, pp. 359–400.

Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B., Wong, M.-Y., 1992. Orthogonal defect classification – a concept for in-process measurements. IEEE Transactions on Software Engineering 18 (11), 943–956.

Clark, L.A., Pregibon, D., 1993. Tree based models. In: Chambers, J.M., Hastie, T.J. (Eds.), Statistical Models in S. Chapman & Hall, London,, pp. 377–419 (Chapter 9).

Kallepalli, C., Tian, J., 2001. Measuring and modeling usage and reliability for statistical web testing. IEEE Transactions on Software Engineering 27 (11), 1023–1036.

Kan, S.H., 2002. Metrics and Models in Software Quality Engineering, second ed. Addison-Wesley, Reading, MA.

Li, Z., Tian, J., 2003. Analyzing web logs to identify common errors and improve web reliability. In: Proceedings of the IADIS International Conference on e-Society, Lisbon, Portugal, pp. 235–242.

Lutz, R.R., Mikulski, I.C., 2004. Ongoing requirements discovery in high-integrity systems. IEEE Software 21 (2), 19–25.

Lyu, M.R. (Ed.), 1995. Handbook of Software Reliability Engineering. McGraw-Hill, New York.

Ma, L., Tian, J., 2003. Analyzing errors and referral pairs to characterize common problems and improve web reliability. In: Proceedings of the 3rd International Conference on Web Engineering, Oviedo, Spain, pp. 314–323.

Mays, R.G., Jones, C.L., Holloway, G.J., Studinski, D.P., 1990. Experiences with defect prevention. IBM Systems Journal 29 (1), 4–32.

Musa, J.D., Iannino, A., Okumoto, K., 1987. Software Reliability: Measurement, Prediction, Application. McGraw-Hill, New York.

Nelson, E., 1978. Estimating software reliability from test data. Microelectronics and Reliability 17 (1), 67–73.

Offutt, J., 2002. Quality attributes of web applications. IEEE Software 19 (2), 25–32.

Tian, J., 2005. Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement. John Wiley & Sons Inc. and IEEE CS Press, Hoboken, New Jersey.

Tian, J., Henshaw, J., 1994. Tree-based defect analysis in testing. In: Proceedings of the 4th International Conference on Software Quality, McLean, Virginia.

Tian, J., Ma, L., 2006. Web testing for reliability improvement. In: Zelkowitz, M.V. (Ed.), Advances in Computers, 67. Academic Press, San Diego, CA, pp. 177–224.

Tian, J., Rudraraju, S., Li, Z., 2004. Evaluating web software reliability based on workload and failure data extracted from server logs. IEEE Transactions on Software Engineering 30 (11), 754–769.

Li Ma received the B.S. and M.S. degrees in Computer Science from Xi'an Jiaotong University, China, in 1998 and 2001, respectively. She is currently working toward the Ph.D. degree in Computer Science at Southern Methodist University, Dallas, Texas. Her research area includes software quality and reliability, software testing, software measurement and web engineering. She is a student member of the IEEE Computer Society.

Jeff (Jianhui) Tian received a B.S. degree in Electrical Engineering from Xi'an Jiaotong University in 1982, an M.S. degree in Engineering Science from Harvard University in 1986, and a Ph.D. degree in Computer Science from the University of Maryland in 1992. He worked for the IBM Software Solutions Toronto Laboratory between 1992 and 1995 as a software quality and process analyst. Since 1995, he has been with Southern Methodist University, Dallas, Texas, now as an Associate Professor of Computer Science and Engineering, with joint appointment at the Dept. of Engineering Management, Information and Systems. His current research interests include software testing, measurement, reliability, safety, complexity, and applications in commercial, web-based, telecommunication, and embedded software and systems. He is a member of IEEE and ACM.