Evaluating Web Software Reliability Based on Workload and Failure Data Extracted from Server Logs

Jeff Tian*†, Sunita Rudraraju and Zhao Li
Southern Methodist University, Dallas, Texas, USA

Abstract

In this paper, we characterize usage and problems for web applications, evaluate their reliability, and examine the potential for reliability improvement. Based on the characteristics of web applications and the overall web environment, we classify web problems and focus on the subset of source content problems. Using information about web accesses, we derive various measurements that can characterize web site workload at different levels of granularity and from different perspectives. These workload measurements, together with failure information extracted from recorded errors, are used to evaluate the operational reliability for source contents at a given web site and the potential for reliability improvement. We applied this approach to the web sites `www.seas.smu.edu` *and* `www.kde.org`*. The results demonstrated the viability and effectiveness of our approach.*

Keywords: *World Wide Web (WWW) and Internet, web applications and web server logs, quality and reliability, reliability modeling, workload measurement.*

1 Introduction

With the prevalence of the World Wide Web (WWW, or simply the *web*) and people's reliance on it in society today, ensuring its satisfactory reliability is becoming increasingly important. Various techniques exist today to characterize workload for general software and computer systems, and to measure and assure their reliability [9, 12, 21]. However, the web environment presents many new challenges [6, 14], and requires adapted or newly developed techniques based on the characterization of the web, its usage, and related problems.

For web applications, various log files are routinely kept at web servers. In this paper, we extract web usage and failure information from these log files to evaluate web software reliability and the potential for reliability improvement. This approach has been applied to the web sites `www.seas.smu.edu` and `www.kde.org` to demonstrate its viability and effectiveness.

The rest of the paper is organized as follows: Section 2 analyzes the general characteristics of the web and its reliability problems. Section 3 examines the contents of web logs and their use in evaluating web site workload and web software reliability. Section 4 presents our initial results for `www.seas.smu.edu`, which are cross-validated by the results for `www.kde.org` in Section 5. Conclusions and perspectives are presented in Section 6.

2 Reliability and the Web

We next examine the general characteristics of the web and common problems in web applications to set the stage for us to evaluate web software reliability.

2.1 Defining reliability for web applications and their components

The *reliability* for web applications can be defined as the probability of failure-free web operation completions. We define web *failures* as the inability to correctly obtain or deliver information, such as documents or computational results, requested by web users. This definition conforms to the standard definition of failures being the behavioral deviations from user expectations [5]. Based on this definition, we can consider the following failure sources:

*Final version published in *IEEE Trans. on Software Engineering*, 30(11):754-769, Nov., 2004.

†For correspondence, contact Dr. Jeff Tian, Computer Science & Engineering Dept., Southern Methodist University, Dallas, Texas 75275. Phone: (214) 768-2861; fax: (214) 768-3085; e-mail: tian@engr.smu.edu.

- *Host, network, or browser failures* that prevent the *delivery* of requested information to web users. These failures are similar to failures in regular computer systems, network, or software, which can be analyzed and assured by existing techniques [9, 12, 21].

- *Source content failures* that prevent the *acquisition* of the requested information by web users because of problems such as missing or unaccessible files, trouble with starting JavaScript, etc. These failures are closely related to the specific web-based services that a site provides, and possess various characteristics unique to the web environment [6, 14].

- *User errors*, such as improper usage, mistyped URL, etc., may also cause problems, which can be addressed through user education, better usability design, etc. These failures are beyond the control of web service or content providers.

The end-to-end reliability defined earlier, which measures the probability of failure-free completions of web operations, includes all the problems listed above in its reliability evaluation. However, as also noticed above, many of these problems can be either addressed by existing approaches or are simply beyond the control and responsibility of the local web content providers. In addition, ensuring reliability defined this way would require concerted quality assurance effort over the whole Internet by the global community.

On the other hand, web site software problems, or web source content problems noted above, are a significant part of the overall problems for web operations. In addition, they can generally be addressed locally at the web site by the content providers. Consequently, we focus on the web source content failures and the related web software reliability in this study.

Also worth noting is the differences between web software reliability we restrict ourselves to and web site availability. Normal maintenance activities and network problems may make a web site temporarily unavailable. However, such problems are generally perceived as less serious by web users than web software problems, because the users are more likely to succeed in accessing required information after temporary unavailability, while software problems would persist unless the underlying causes are identified and fixed. This fact also partially justifies our focus on web software reliability.

2.2 Measuring web software reliability and workload

In general, the failure information alone is not adequate to characterize and measure the reliability of a software system, unless there is a constant workload [9, 12]. Due to the vastly uneven web traffic observed in previous studies [1, 15], we need to measure both the web failures and related workload for reliability analyses. Specific characteristics that make web workload measurement different from that for traditional software systems include:

- *Massiveness and diversity*: Web applications provide cross-platform universal access to web resources for everyone with an Internet access. The massive user population, the diverse hardware/software configurations, and the varied usage patterns need to be reflected in the selected workload measures.

- *Document and information focus*, as compared to the computational focus for most traditional workload. Although some computational capability has evolved in newer web applications, information search and retrieval still remain the dominant usage for most web users. A fundamental difference exists between these two workload types.

These characteristics require us to measure actual web workload to ensure its satisfactory reliability instead of indiscriminately using generic measures suitable for traditional computation-intensive workload. Due to the nature of uneven web workload, only usage-dependent workload measures among the traditional ones, such as CPU execution time runs, and transactions, need to be considered for reliability evaluation [9, 12]. However, the user focus and substantial amount of idle time during browsing sessions make any variation of execution time unsuitable for web workload measurement. Similarly, the dominance of non-computational tasks also makes computational task oriented transactions unsuitable for web workload measurement. Instead, other workload measures, such as those we derive in Section 3, may be more suitable for characterizing workload at web sites.

2.3 Basics of reliability analysis and modeling

Both the failure information and the related workload measurements provide us with data input to various software reliability models [9, 12, 18]. The output of these

```
129.119.4.17 - - [16/Aug/1999:00:00:11 -0500] "GET /img/XredSeal.gif HTTP/1.1" 301
328 "http://www.seas.smu.edu/" "Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)"
129.119.4.17 - - [16/Aug/1999:00:00:11 -0500] "GET /img/ecom.gif HTTP/1.1" 304 -
"http://www.seas.smu.edu/" "Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)".
```

Table 1. Sample entries in an access log

```
[Mon Aug 16 13:17:24 1999] [error] [client 207.136.6.6] File does not exist:
/users/seasadm/webmastr/htdocs/library/images/gifs/homepage/yellowgradlayers .gif
[Mon Aug 16 13:17:37 1999] [info] [client 199.100.49.104] Fixed spelling:
/img/XredSeal.gif to /img/xredSeal.gif from http://www.seas.smu.edu/
```

Table 2. Sample entries in an error log

models can help us evaluate the web software reliability and the potential for reliability improvement.

Two basic types of software reliability models are: input domain reliability models (IDRMs) and time domain software reliability growth models (SRGMs). IDRMs can provide a snapshot of the web site's current reliability. For example, if a total number of f failures are observed for n workload units, the estimated reliability R according to the Nelson model [13], one of the most widely used IDRMs, can be obtained as:

$$R = \frac{n-f}{n} = 1 - \frac{f}{n} = 1 - r$$

Where r is the failure rate, which is also often used to characterize reliability. When usage time t_i is available for each workload unit i, the summary reliability measure, mean-time-between-failures (MTBF), can be calculated as:

$$\text{MTBF} = \frac{1}{f}\sum_i t_i$$

When the usage time t_i is not available, we can use the number of workload units as the rough time measure. In this case,

$$\text{MTBF} = \frac{n}{f}$$

If discovered defects are fixed over the observation period, the defect fixing effect on reliability (or reliability *growth* due to defect removal) can be analyzed by using various software reliability growth models (SRGMs) [9, 12]. For example, in the widely used Goel-Okumoto model [4], the failure arrival process is assumed to be a non-homogeneous Poisson process. The expected cumulative failures, $m(t)$, over time t is given by the formula:

$$m(t) = N(1 - e^{-bt})$$

where the model constants N (total number of defects in the system) and b (model curvature) need to be estimated from the observation data. SRGMs can also be used to assess the potential for reliability improvement.

3 Analyzing Web Logs for Reliability Evaluation

Monitoring web usage and keeping various logs are necessary to keep a web site operational. Two types of log files are commonly used by web servers: Individual web accesses, or hits, are recorded in *access logs*, with sample entries given in Table 1; related problems are recorded in *error logs*, with sample entries given in Table 2. Analyzing information stored in such logs can help us evaluate web site workload and web software reliability, as discussed below.

3.1 Error log analysis

Error logs typically include details about the problems encountered. The format is simple: a time-stamp followed by the error or warning message, such as in Table 2.

Common problems or error types are listed in Table 3. Notice that most of these errors conform closely to the source content failures we defined in Section 2. Therefore, they can be used in our web software reliability evaluation.

Questions about error occurrences and distribution can be answered directly by analyzing error logs. However, as discussed in Section 2, evaluation of web software reliability also needs the measurement data for web usage or workload. The web usage information and the

type	description
A	permission denied
B	no such file or directory
C	stale NFS file handle
D	client denied by server configuration
E	file does not exist
F	invalid method in request
G	invalid URL in request connection
H	mod_mime_magic
I	request failed
J	script not found or unable to start
K	connection reset by peer

Table 3. Error types

related workload measurements can be extracted from web server access logs, as described below.

3.2 Analysis of access log contents

A "hit" is registered in an access log if a file corresponding to an HTML page, a document, or other web content is explicitly requested, or if some embedded content, such as graphics or a Java class within an HTML page, is implicitly requested or activated.

Information recorded in access logs typically includes: the requesting computer, user name and identity information for authentication, date and time of the request, name and size of the requested file, HTTP status code, referral page, and client name. Specific information useful to our workload analysis recorded in access logs includes:

- The reverse-DNS hostname or IP number of the machine making the request.
- Date and time that the transfer took place.
- Total number of bytes transferred.

They are recorded as the 1st, 4th, and 7th field respectively of each hit entry in the access log, as illustrated in Table 1. If the value for any field is not available, a "−" is put in its place.

3.3 Defining workload measures and extracting them from access logs

As mentioned in Section 2, various software reliability models relate observed failures to usage time for reliability evaluation. From the perspective of web service providers, the usage time for web applications is the actual time spent by every user at the local web site. However, the exact time is difficult to obtain and may involve prohibitive cost or overhead associated with monitoring and recording dynamic behavior by individual web users [15]. One additional complication is the situation where a user opens a web page and continues with other tasks unrelated to the page just accessed. In this situation, the large gap between successive hits is not a reflection of the actual web usage time by this user. To approximate the usage time, we can use various workload measures considered below.

The most obvious workload measure is to count the number of hits, because 1) each hit represents a specific activity associated with web usage, and 2) each entry in an access log corresponds to a single hit, thus it can be extracted easily. In fact, spent using the web site this has already been done for statistical web testing and reliability assurance [6], which also demonstrated that hit count is a viable candidate for the evaluation of web site workload and web software reliability.

Overall hit count defined above can be misleading if the workload represented by individual hits shows high variability. Consequently, we can choose the number of bytes transferred, or byte count, as the workload measure of finer granularity, which can be easily obtained by counting the number of bytes transferred for each hit recorded in access logs.

User count is another alternative workload measure meaningful to the organizations that maintain the web sites and support various services at the user level. When calculating the number of users for each day, we treat each unique IP address as one user. So, no matter how many hits were made from the same computer, they are considered to be made by the same user. This measure gives us a rough picture of the overall workload handled by the web site.

One of the drawbacks of user count is its coarse granularity, which can be refined by counting the number of user sessions. In this case, along with the IP address, access time can be used to calculate user sessions: If there is a significant gap between successive hits from the same IP address, we count the later one as a new session. In practice, the gap size can be adjusted to better reflect appropriate session identification for the specific types of web applications.

The number of user sessions per day may be a better measure of overall workload than the number of users, because big access gaps are typically associated with changes of users or non-web related activities by the same user. Each user who accesses the same web site

Error type	A	B	C	D	E	F	G	H	I	J	K	Total
Number of errors	2079	14	4	2	28631	0	1	1	1	27	0	30760

Table 4. Summary of total recorded errors by type for SMU/SEAS

from the same computer over successive intervals will be counted by user sessions, as long as such a gap exists in between. Even for a single user, a significant access gap is more likely to be associated with different usage patterns than within a single time burst. Therefore, by using user sessions, we can count the users' active contribution to the overall web site workload more accurately.

To summarize, the web workload measures at different levels of granularity and from different perspectives that we can extract from web server access logs include:

- Number of hits, or hit count.
- Number of bytes transferred, or byte count.
- Number of users, or user count.
- Number of user sessions, or session count.

In our previous study of statistical web testing and reliability analysis [6], we implemented utility programs in Perl to count the number of errors, number of hits, and frequently used navigation patterns. We extended these utility programs here to extract and analyze the workload data defined above. We also used a commercial tool, S-PLUS[1], and related utility programs to perform statistical analysis on the collected data, fit reliability models, and present results in graphical forms.

4 Initial Results and Discussions

We next present our initial results for `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 [2], a popular choice among many web hosts, and shares many common characteristics of web sites for educational institutions. These features make our results and observations meaningful to many application environments. Server log data covering 26 consecutive days in 1999 were used. Limitations due to the use of this web site and the specific data are addressed in Section 5.

[1] S-PLUS is a trademark of Insightful, Inc.

4.1 Error distribution and trend analysis

Table 4 gives the summary of different types of errors for SMU/SEAS. The error types and their brief descriptions were given in Table 3. Among the different error types, the most dominant ones are type A errors ("permission denied") and type E errors ("file does not exist"). These two types together account for almost all (99.9%) of the recorded errors.

Type A errors, accounting for 6.8% of the total recorded errors, involve improper access authorization or problems with the authentication process. These errors are more closely related to security problems instead of reliability problems we focus on in this paper. Further analyses of them may involve the complicated authentication process. In addition, type A errors also account for much less of a share of the total recorded errors as compared to type E errors. Therefore, we decided not to use these errors in our web software reliability analysis here.

Type E errors usually represent bad links. They are by far the most common type of problems in web usage, accounting for 93.1% of the total recorded errors. This is in agreement with survey results from 1994$\sim$1998 by the Graphics, Visualization, and Usability Center of Georgia Institute of Technology (see `http://www.gvu.gatech.edu/user_surveys/`). The surveys found that broken link is the problem most frequently cited by web users, next only to network speed problem. Therefore, type E error is the most observed web content problem for the general population of web users. Further analysis can be performed to examine the trend of these failures, and to provide an objective assessment of the web software reliability.

We performed a preliminary analysis of the originators of these bad links [10] and discovered that the majority of them are from internal links, including mostly URLs embedded in some web pages and sometimes from pages used as start-ups at the same web site. Only a small percentage of these errors are from other web sites (4.3%), web robots (4.4%), or other external sources, which are beyond the control of the local site content providers, administrators, or maintainers. Therefore, the identification and correction of these problems represent realistic opportunities for improved web software relia-

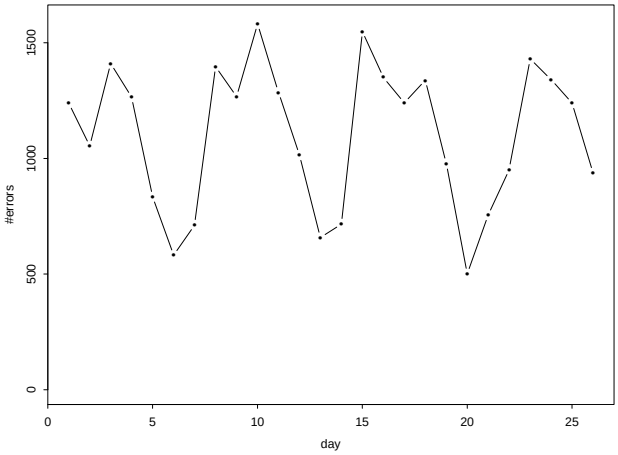

Figure 1. Error profile over time for SMU/SEAS

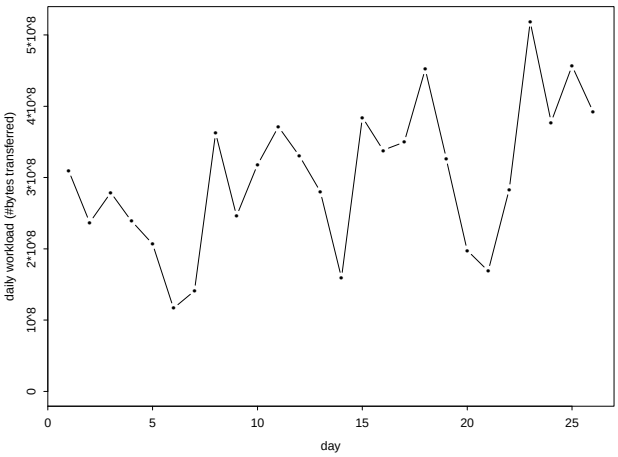

Figure 2. Daily bytes transferred over time for SMU/SEAS

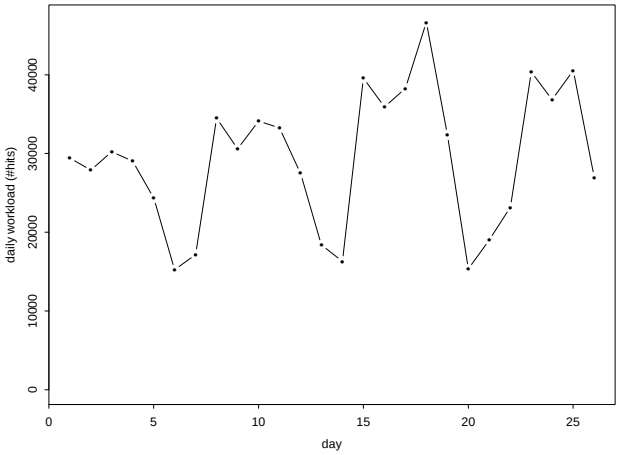

Figure 3. Daily hits over time for SMU/SEAS

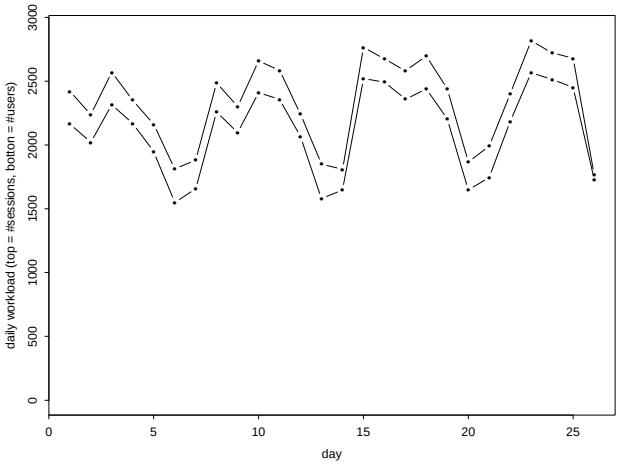

Figure 4. Daily sessions (top curve) or users (bottom curve) over time for SMU/SEAS

bility based on local actions.

Figure 1 plots the number of type E errors over time under normal operations. These daily errors vary considerably. However, the number of problems encountered per day is closely related to the actual usage intensity. As mentioned in Section 2, a workload profile with considerable variability, such as the web traffic in general [1, 15], is a clear indication that measuring failures alone over calendar time is not suitable for reliability analysis. Various usage or workload measurements, such as the ones defined in Section 3, are needed for web software reliability evaluation.

4.2 Workload measurement results

The workload measurement results are plotted in Figure 2 for daily bytes transferred, in Figure 3 for daily hits, and in Figure 4 for daily sessions (top curve) and daily users (bottom curve). We used the standard two-hour gap [11] to identify user sessions here. On the average, each day is associated with 301.6 Mbytes, 29,345 hits, 2,338 sessions, and 2,120 users; each user is associated with 13.8 hits; each user session is associated with 11.6 hits; and each hit is associated with 10,279 bytes. No matter which workload measure is used, the daily workload shows several apparent characteristics, as fol-

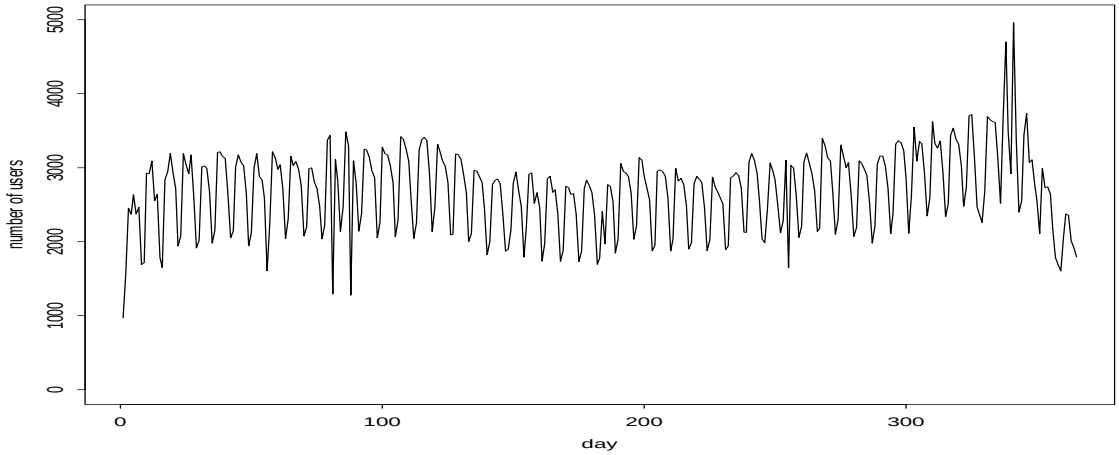

Figure 5. Daily users over long-term for SMU/SEAS

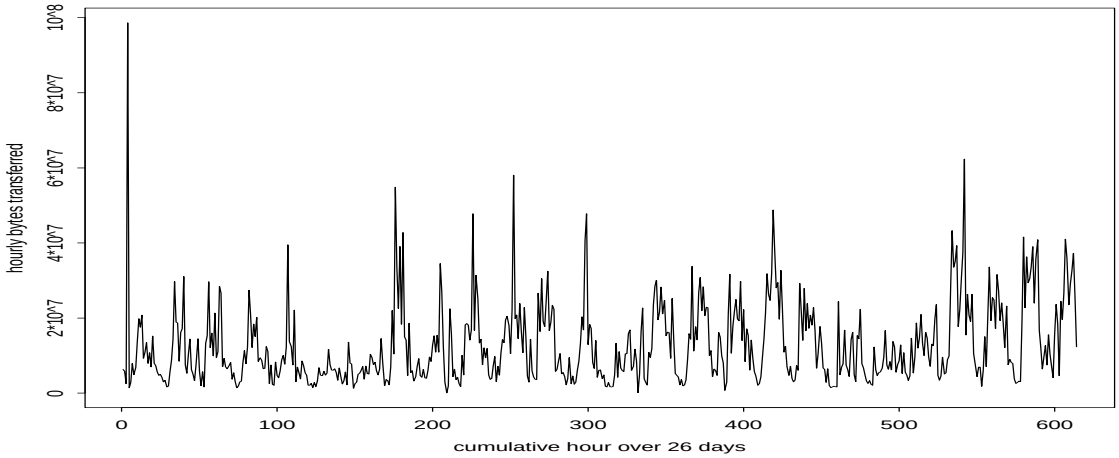

Figure 6. Hourly bytes transferred for SMU/SEAS

lows:

- *Uneven distribution and variability*: The distribution is highly uneven and varies from day-to-day, as represented by the peaks and valleys in these workload plots, which conforms to previously observed traffic patterns [1, 15]. Among the four workload measures, daily bytes and daily hits show larger variability in relative magnitudes than daily users or daily sessions. This result indicates that although the number of users or user sessions may remain relatively stable, some users may use the web much more intensively than others, resulting in larger variations in detailed web site workload measurements over time.

- *A periodic pattern that synchronize with error profile*, which is characterized by weekdays normally associated with heavier workload than during the weekends. This pattern seems to conform to the self-similar web traffic [3]. In addition, this periodic pattern are correlated or synchronized with daily error profile in Figure 1. This fact indicates that these workload measures are viable alternatives for web software reliability evaluation, because of the direct relation between usage and possible failures for the web site's source contents demonstrated in such synchronized patterns.

- *A long term stability for the overall trend*, which can be cross-validated by examining the trend over a longer period instead of for just 26 days, such as we plotted in Figure 5 for the number of daily users over a year. The other workload measures traced over the whole year also showed the same long-term stability. This is probably due to the stable

enrollment for SMU/SEAS and web site stability where no major changes were made to our web-based services over the observation period.

Of the four workload measures, hits, users, and sessions can be extracted from access logs easily and consistently. However, byte counting was somewhat problematic, because "byte transferred" field was missing not only for the error entries but also for many other entries. Further investigation revealed that most of these missing entries were associated with files or graphics already in the users cache ("file not modified", therefore no need to resend or reload). Since the total number of entries with missing bytes information represented about 15% of the total number of entries (hits), we simply used the rest to calculate the number of bytes transferred in this paper. The impact of this counting scheme is discussed in connection to future work in Section 6.

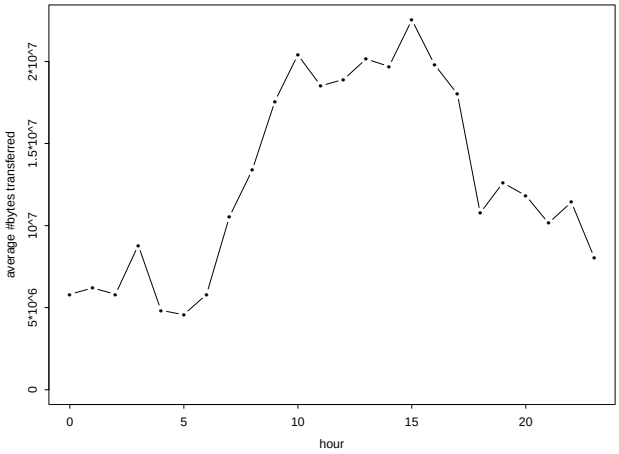

Figure 7. Average hourly bytes transferred in a day for SMU/SEAS

4.3 Hourly workload measurement results

Workload measurements associated with periods shorter than calendar days, such as by the hour or by some short bursts, can also be used in reliability analysis. With time-stamped individual hits and related information available in the access log, such measurements can be easily obtained.

Figure 6 plots hourly bytes transferred for the 26 day period. We can see numerous peaks and valleys associated with different hours and days. Huge workload differences exist between some peaks and valleys, similar to other Internet traffic [1, 15]. We also examined the other hourly workload measures, but they do not show such big differences between their peaks and valleys, although the same pattern remains.

There is a remarkable consistence in the overall hourly patterns from day to day. Figure 7 plots the average hourly bytes transferred for different hours of a day, averaging over the 26 day period. Other plots we produced for average hourly workload also show similar patterns but with slightly less variability. This pattern looks remarkably similar to previously observed patterns [1], which indicates that daily web workload profile over different hours has remained fairly stable over the years. Knowing the normal web site workload patterns over the hours of a day can help us schedule system maintenance tasks to minimize possible disturbance to normal web-base services.

As for reliability analysis, data with less variability, usually through data grouping or clustering, are generally preferable, because they typically produce more stable models that fit the observations better and provide better reliability assessments and predictions [16, 17, 20]. In our case, daily data have much less variability, as shown by comparing Figure 6 to Figure 2, yet give us enough data points (26) to be used in statistical analyses and model fitting. Consequently, we only use daily data in subsequent reliability analyses.

4.4 Analysis of operational reliability

When we combine the measurement results for the web failures, in this case type E errors extracted from the error log, and workload measured by the number of users, sessions, hits, and bytes transferred, we can perform analyses to evaluate web software reliability.

As observed earlier, the peaks and valleys in errors represented in Figure 1 generally coincide with the peaks and valleys in workload, such as in Figures 2, 3, and 4. This close relationship between usage time and failure count can be graphically examined as in Figure 8, plotting cumulative errors vs. cumulative bytes transferred over the observation period. An essentially linear relation can be detected between the two. Similar observations can be obtained if we plot cumulative errors vs. cumulative hits, users, or sessions.

This relationship can also be characterized by the daily failure rate, as defined by the number of errors divided by the workload measured by bytes transferred, hits, users, or sessions for each day. These daily failure rates also characterize web software reliability, and can be interpreted as applying the Nelson model [13] mentioned in Section 2 to daily snapshots. Table 5 gives the

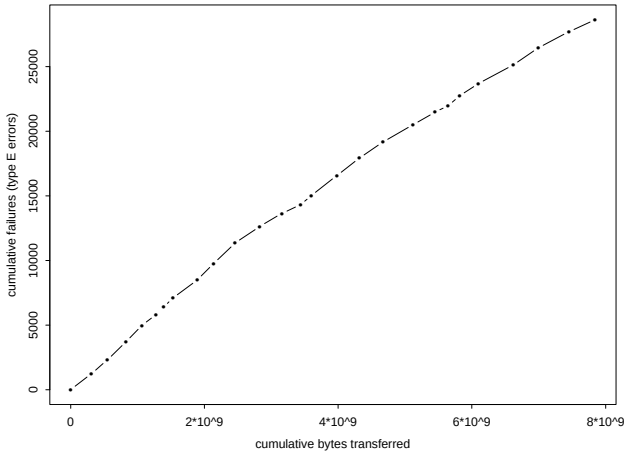

Figure 8. Cumulative errors vs. cum. bytes transferred for SMU/SEAS

range (min to max), the mean, and the standard deviation (std.dev), for each daily error rates defined above. Because these rates are defined for different workload measurement units and have different magnitude, we use the relative standard error, or *rse*, defined as: *rse = std.dev / mean*, to compare their relative spread in Table 5. We also included the daily error count for comparison. All these daily error rates fall into tighter spread than daily error count, which indicates that they provide more consistent and stable reliability estimates than daily error count.

Since individual web failures are directly associated with individual hits, we can use the Nelson model [13] described in Section 2 to evaluate the overall web software reliability using failures and hits for the complete 26 days. This gives us the site software reliability of $R = 0.962$, or 96.2% of the individual web accesses will be successful. This model also give us an MTBF = 26.6 hits, or averaging one error for every 26.6 hits.

Modeling with other workload measures is also possible. For example, the above MTBF can be recalculated for other workload units, giving us an MTBF = 273,927 bytes, an MTBF = 1.92 users, or an MTBF = 2.12 sessions. That is, this site can expect, on the average, to have a problem for every 273,927 bytes transferred, for every 1.92 users, or for 2.12 sessions. The web software reliability R in terms of these workload measures can also be calculated by the Nelson model. However, result interpretation can be problematic, because web failures may only be roughly associated with these workload measures. For example, because of the missing byte transferred information in the access logs for failed requests, the failures can only be roughly placed in the sequence of bytes transferred, resulting in imprecise reliability assessments and predictions. On the other hand, individual web failures may be roughly associated with certain users or user sessions through the particular hits by the users or within the sessions. In this case, each user or session may be associated with multiple failures, and appropriate adjustments to modeling results might be called for. For example, it might be more appropriate to separate failure-free sessions from sessions with failures, instead of comparing the number of failures in a session.

4.5 Evaluating potential reliability improvement

Under the idealized environment, the fault(s) that caused each observed failure can be immediately identified and removed, resulting in no duplicate observations of identical failures. This scenario represents the upper limit for the potential reliability improvement if we attempt to fix operational problems on-line or if we attempt to test the system and fix problems under simulated customer operational environment. This upper bound on reliability growth may not be attainable under many circumstances because of the large number of transient faults that usually take place whose origins are usually very difficult to be identified and removed because of their dependency on the context. Nevertheless, this upper limit gives us an idea about the potential reliability growth. Should quantitative information become available about the faults that are hard to fix, it can be used to fine tune the above limit to provide more accurate estimation of reliability growth potential.

This limit on potential reliability improvement can be measured by the reliability change (or *growth*) through the operational duration or testing process where such defect fixing could take place. Under the web application environment, each observed failure corresponds to a recorded type E error in the error log, and the idealized defect fixing would imply no more observation of any duplicate type E errors. In other words, failure arrivals under this hypothetical environment would resemble the sequence of unique type E errors extracted from the error log, which can be calculated by counting each type E error only once at its first appearance but not subsequently.

In general, reliability growth can be visualized by the gradual leveling-off of the cumulative failure arrival curve, because the flatter the part of the curve, the more time it takes to observe the next failure. To visualize

error rate	min	max	mean	std.dev	rse
errors/bytes	2.35×10^{-6}	5.30×10^{-6}	3.83×10^{-6}	9.33×10^{-7}	0.244
errors/hits	0.0287	0.0466	0.0379	0.00480	0.126
errors/sessions	0.269	0.595	0.463	0.0834	0.180
errors/users	0.304	0.656	0.5103	0.0859	0.168
errors/day	501	1582	1101	312	0.283

Table 5. Daily error rate (or failure rate) for SMU/SEAS

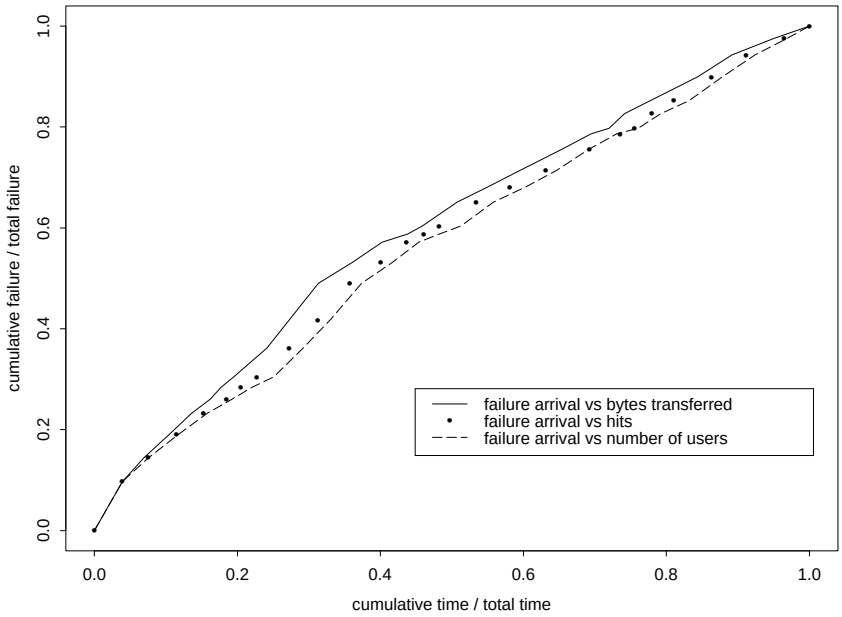

Figure 9. Reliability growth comparison for different workload measures for SMU/SEAS

this, we plotted in Figure 9 cumulative unique failures versus different workload measurements we calculated above. Relative scale is used to better compare the overall reliability growth trends. The individual data points in the middle depict the failure arrivals indexed by cumulative hits. The (top) solid line depicts failure arrivals indexed by the cumulative bytes transferred. The (bottom) dashed line depicts failure arrivals indexed by the cumulative number of users. The user session measurement resulted in almost identical curve shape as that for the number of users, thus was omitted to keep the graph clean. As we can see from Figure 9, there is an observable effect of reliability growth for this data, with the tail-end flatter than the beginning for all three curves.

Quantitative evaluation of reliability growth can be provided by software reliability growth models (SRGMs), which commonly assume instantaneous defect fixing [9, 12]. In this paper, we use a single measure, the purification level ρ [19] to capture this reliability change:

$$\rho = \frac{\lambda_0 - \lambda_T}{\lambda_0} = 1 - \frac{\lambda_T}{\lambda_0}$$

where λ_0 and λ_T are the initial and final failure rates, respectively, estimated by a fitted SRGM. Complete elimination of all potential defects would result in $\rho = 1$, and no defect fixing would result in $\rho = 0$. Normal reliability growth is associated with ρ values ranging between these two extremes, with larger ρ values associated with more reliability growth. $(1 - \rho)$ gives us the ratio between λ_T and λ_0, or the final failure rate as a percentage of the initial failure rate.

We fitted various SRGMs to relate cumulative unique failures to cumulative workload measurements. Figure 10 is an example of such a fitted model, with cumulative bytes transferred as the usage time or workload measurement. The widely used Goel-Okumoto (GO) model [4] introduced in Section 2 was used here to evaluate the potential reliability improvement. The results include:

time/workload measurement	model parameters & estimates						reliability growth ρ
	N	b	SSQ	λ_0	λ_T	MTBF	
bytes	3674	1.76×10^{-10}	54960	6.45×10^{-7}	1.63×10^{-7}	6.14×10^{7}	0.748
hits	4213	1.38×10^{-6}	60880	0.00583	0.00203	493	0.632
sessions	4750	1.42×10^{-5}	66553	0.0675	0.0284	35	0.579
users	4691	1.60×10^{-5}	65063	0.0752	0.0311	32	0.587

Table 6. Reliability modeling results for SMU/SEAS

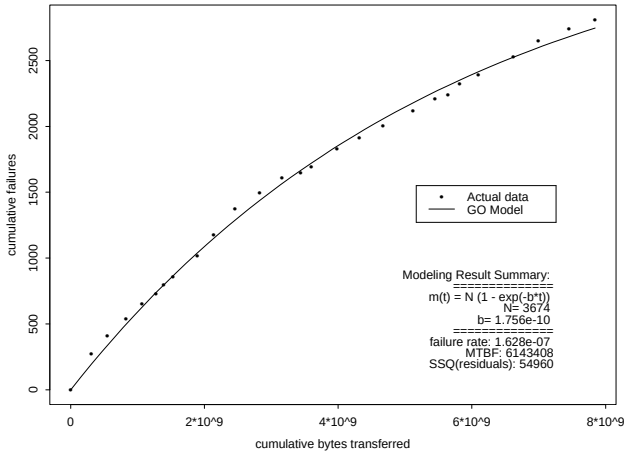

Figure 10. A sample SRGM fitted to SMU/SEAS data

- Fitted model parameters N and b that link expected failure observations $m(t)$ to usage time t in the formula: $m(t) = N(1 - e^{-bt})$.
- Model goodness-of-fit measure, sum of residual squares, or SSQ.
- Various estimates that can be derived from the fitted model: initial failure rate (λ_0), failure rate (λ_T) and MTBF at the end of testing, and purification level (ρ).

Table 6 summarizes these modeling results. The ρ values based on models using different workload measurements indicate that potential reliability improvement ranges from 57.9% to 74.8% in purification levels. In other words, effective web testing and defect fixing equivalent to 26 days of operation could have reduced the failure rate to between slightly less than half ($1-57.9\%$) and about one quarter ($1-74.8\%$) of the initial failure rate. Other SRGMs we tried also yield similar results: A significant reliability improvement potential exists if we can capture the workload and usage patterns in log files and use them to guide software testing and defect fixing.

5 Cross-Validation

We next identify the limitations of our initial study presented above and describe a new study to address these limitations and to cross-validate the above results.

5.1 Addressing the limitations of our initial study

Similar to most previously studies that overwhelmingly focus on academic sites [15], our results above were obtained for an academic site. Despite the factors that contribute to the general validity of our results described at the beginning of Section 4, the most serious limitation of our initial study is that the SMU/SEAS web site may not be a good representative for many non-academic web sites. For example, most of the SMU/SEAS web pages are static ones, with the HTML documents and embedded graphics dominating other types of pages [8], and the web site operates under fairly light traffic. In e-commerce and other applications, workload types may be more diverse, with dynamic pages and context-sensitive contents play a much more important role, and traffic volume can be significantly larger [15]. A related limitation of our initial study is the two-hour gap we used to identify user sessions, which may be suitable for academic sites but not others. For example, most e-commerce applications use 15-30 minutes to time out sessions, and the previously observed average session time-outs are also significantly shorter than two hours [15].

Another limitation is the period covered in our data: 26 days in 1999. Two appealing alternatives are: 1) using more *recent* data, and 2) using data covering a *longer* period of time. However, drastic changes or restructuring of the web site will overshadow the normal trend of observed failures over time, and data covering such periods should be avoided for web software relia-

bility analyses, unless one is specifically addressing the change impact on reliability. The SMU/SEAS web site was overhauled shortly after the period covered in this study, and the ripple effect of the changes lasted a long period. In the meantime, we obtained more recent data from the KDE project to cross-validate our results for SMU/SEAS. On the other hand, longer term data may not be suitable for some of the analyses we performed. For example, the reliability growth analysis assumes that none or few new faults are injected, which can not be true for the dynamic web environment for an extended period of time. Even for the analysis of operational reliability, a stable web site is assumed. Consequently, our use of longer term data is limited to using the one year data from 1999-2000 for the SMU/SEAS web site for only the analyses that are meaningful, such as overall workload trend over longer period.

To overcome these limitations, we obtained and analyzed recent web logs from the KDE project and used different session cut-off values to cross-validate our initial results, as described below.

5.2 KDE web site and logs

For our cross-validation study, we initially planned to analyze some public domain web logs, such as from the Internet Traffic Archive at `ita.ee.lbl.gov` or the W3C Web Characterization Repository at `repository.cs.vt.edu`. However, we discovered that most of the information and log files therein are tailored towards overall Internet and WWW traffic, and are typically older than the SMU/SEAS data we used.

Fortunately, through our recent work with open source projects [7], we learned about the possibility of using measurement data and web logs from open source projects, and obtained web logs from the KDE project web site at `www.kde.org`, with the help of KDE project personnel. The brief description about this project from the KDE web site is given below:

- KDE is a network transparent contemporary desktop environment for UNIX workstations.
- The KDE project is a large open group of developers consisting of several hundred software engineers from all over the world committed to free software development.

Besides providing various information about the KDE project, the KDE web site also supports online download of released documentation and software, and provides online development facilities (including bug database, source reference, WebCVS, etc.). The overall user population and traffic volume are significantly large than the SMU/SEAS web site. Changes are continuously committed to the web site in order to provide the developers and users with the most up-to-date information. These characteristics that differentiate the KDE web site from the SMU/SEAS web site make it a good choice for our validation study. In addition, this web site also uses Apache Web Server [2], which makes our data extraction and analysis easy due to the same data format used.

For each day, the KDE web server generates a log file that contains the access information starting from around 5:05am of the previous day until 5:05am of the current day. Only access logs are used, but not the error logs. However, from the HTTP response code, we can extract the general error information. For example, type E (missing file) error in the error logs is equivalent to access log entries with a response code 404.

To protect the identity and privacy of individual users, the available log data were transformed by KDE personnel using an 1-to-1 mapping for the original IP addresses to make it impossible for us to identify individual users or computers. After some initial problems with data recording and transformation, we obtained consistent data covering over two months in 2003. However, for our analysis, we would like to avoid drastic web changes associated with product releases (which are quite frequent for open source products, such as KDE), web restructuring, and other major events. Therefore, we selected data that fit into an one-month window (31 days) for our subsequent analyses.

5.3 Results for KDE web logs

The access logs for the 31 days recorded more than 14 million hits, of which 793,665 resulted in errors. 785,211 hits resulted in response code 404 (file-not-found), which accounted for 98.9% of all the errors. The next most reported error type was of response code 408, or "request timed out", which accounted for 6225 or 0.78% of all the errors. This dominant share of 404 errors, which is equivalent to type E errors for SMU/SEAS, justifies our focus on this type of errors in our reliability evaluation.

The error profile and three workload profiles using the workload measurements, bytes, hits, and users, are presented in Figure 11. Two different variations of session count were used in dealing with the KDE data: the same two-hour gap cut-off we used for the SMU/SEAS web site previously (labeled s1), and the 15 minutes cut-

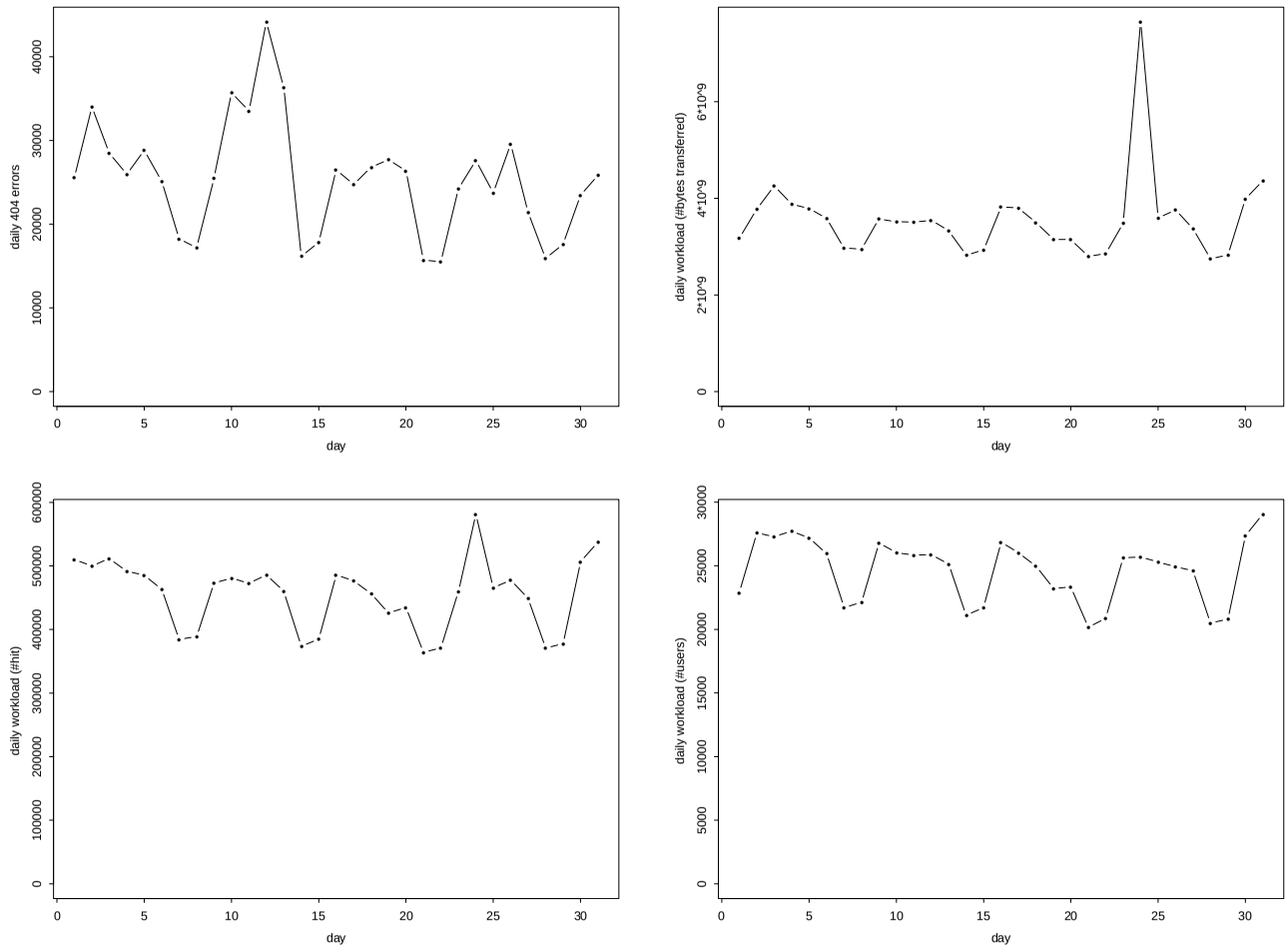

Figure 11. Error and workload profiles over time for KDE (#errors: top-left; #bytes: top-right; #hits: bottom-left; #users: bottom-right)

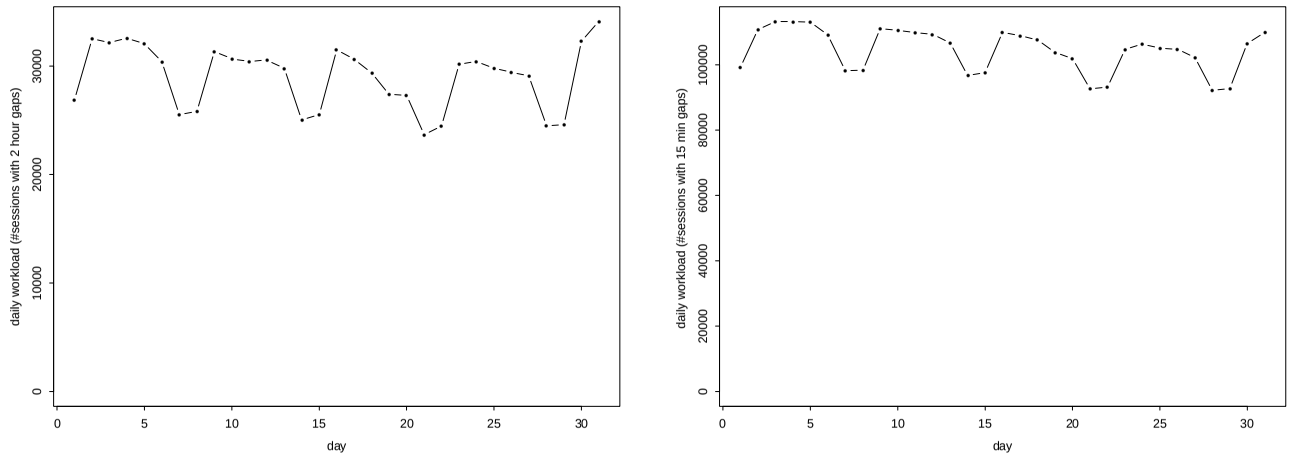

Figure 12. Workload profiles over time for KDE (#sessions with 2hr gaps: left; #sessions with 15min gaps: right.)

error rate	min	max	mean	std.dev	rse
errors/bytes	3.608×10^{-6}	1.246×10^{-5}	7.210×10^{-6}	1.81×10^{-6}	0.251
errors/hits	0.04178	0.09091	0.05519	0.0117	0.211
errors/s1s	0.6335	1.4450	0.8648	0.189	0.219
errors/s2s	0.1665	0.4041	0.2403	0.0554	0.231
errors/users	0.7428	1.7060	1.0180	0.228	0.223
errors/day	15510	44160	25330	6833	0.270

Table 7. Daily error rate (or failure rate) for KDE

off more appropriate for dynamic pages (labeled s2). The session profiles are plotted in Figure 12.

The overall traffic at this web site is significantly heavier than that for SMU/SEAS, with a daily average of 3,563 Mbytes, 455,005 hits, 24,656 users, 29,029 s1 sessions, and 104,490 s2 sessions. However, all the observations about workload and error distribution, trends, and patterns for SMU/SEAS remain valid here, except for the long term stability which was only partially validated by the data covering slightly more than two months due to our lack of longer term data for KDE. In fact, the visual patterns of Figure 11 and Figure 12 look remarkably similar to those of Figures 1 through 4. Although the relative differences for KDE tends to be smaller than that for SMU/SEAS, likely due to the heavier traffic by a larger user population.

As expected, the hourly workload measurement results for the KDE project are similar to that for SMU/SEAS. However, the workload distribution over different hours of the day, plotted in Figure 13, shows some differences to Figure 7 for SMU/SEAS, while maintaining similar patterns. The peak seems to be shifted right, to later hours, and the low activity hours are relatively short. These differences can probably be attributed to the nature of open source projects, where most of the developers work on the projects on their spare time, typically in late afternoons and evenings. The hour with the lowest workload is also the hour they scheduled to cut off the daily access logs and to save the information.

Table 7 gives the evaluation results of operational reliability. Expectedly, the same patterns hold, i.e., all the daily failure rates fall into tighter bands than that for the daily errors, to give consistent and stable assessments of the operational reliability of this web site's contents. The overall reliability values are also roughly the same as that for SMU/SEAS. For example, on average, 5.76% of the hits would result in 404 errors, or the web site was 94.2% reliable as compared to 96.2% for SMU/SEAS.

We also repeated the assessment of reliability growth

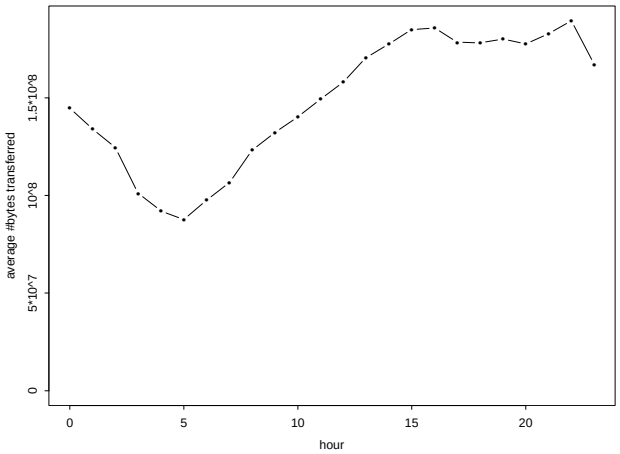

Figure 13. Average hourly bytes transferred in a day for KDE

potential for the KDE web site. However, when we extracted the unique failures (unique 404 errors), we noticed an anomaly at the 24th day, which was associated with more than 10 times the maximal daily unique errors for all the previous days. Further investigation revealed that this is related to a planned beta release of the KDE product, when the web contents were drastically changed and many new faults were injected. Since our reliability growth evaluation is for stable situations where few or none new faults are injected, as is the assumption for all the software reliability growth models [9, 12], we restricted our data to the first 22 days in this analysis.

Figure 14 plots the reliability growth evaluation we carried out for the KDE data. Among the five workload measures we used, bytes, hits, users, s1 and s2 sessions, all produced almost identical results in the reliability growth visualization, when we plotted relative cumulative unique errors against relative cumulative workload, similar to what we did in Figure 9. The comparative visualization is omitted here because all the relative reliability growth curves would closely resemble the actual

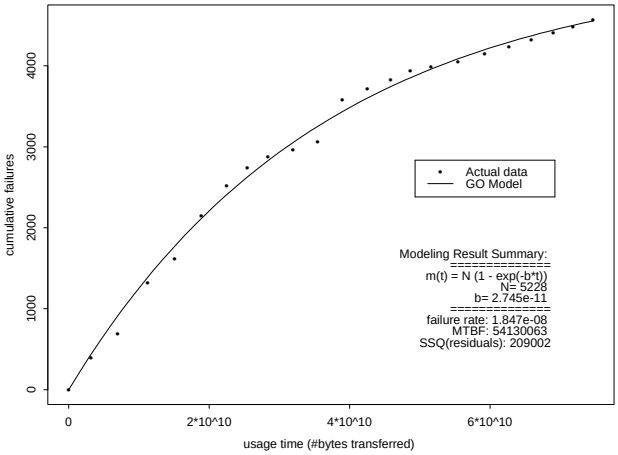

Figure 14. A sample SRGM fitted to KDE data

curve represented by the actual data points in Figure 14. A visual inspection of Figure 14 also revealed more degrees of reliability growth, or more bending of the data trend and fitted curve, than that in Figures 9 and 10. Reliability growth potential as captured by ρ for the KDE web site ranged from 86.7% to 88.9% (with the model in Figure 14 gave us $\rho = 87.1\%$). In other words, effective web testing and defect fixing equivalent to 22 days of operation could have reduced the failure rate to about 11% to 13% (calculated by $1 - \rho$) of the initial failure rate; or, equivalently, almost all the original problems could have been fixed.

6 Conclusions and Perspectives

By analyzing the unique problems and challenges for the web environment, we have developed an approach for web software reliability evaluation based on information extracted from existing web server logs. This approach has been applied to study the web sites `www.seas.smu.edu` for the School of Engineering and Applied Science at Southern Methodist University (SMU/SEAS) and `www.kde.org` for the KDE project. Our key findings are summarized below:

- *Measure derivation and data extraction*: Specific web software problems related to missing files and four workload measures, bytes transferred, hit count, number of users, and number of sessions, were derived in this paper for web software reliability evaluation. Detailed failure data can be extracted from error logs. When such logs are not available, rough failure data can be extracted from access logs. Hit count, byte count, and user count can be easily extracted from access logs, due to their direct correspondence to access log entries and the embedded data fields "bytes transferred" and "IP address". Session count computation may involve history information for individual users or unique IP addresses, but properly identified user sessions with appropriate time-out values can reflect web usage better than simply counting the users.

- *Assessing the operational reliability for web software*: When used with failure data to estimate failure rate or reliability, all four workload measures proposed in this paper produced more consistent and stable reliability estimates than using daily errors alone. They offer reliability assessments from different perspectives, and each may be suitable for specific situations. For example, byte count might be more suitable for traffic measurement and related reliability interpretations; hit count might be more meaningful to web users as they browse individual pages; while number of users or sessions might be more suitable for high level web site reliability characterization.

- *Assessing the potential for reliability improvement*: Also demonstrated in both case studies is the significant potential for reliability improvement if defect can be fixed under a stable environment, leading to reliability growth ranging from 58% to 89% in purification levels within a month of such improvement actions.

- *Some generalization beyond our two case studies*: Many of the results we obtained and patterns we observed concerning workload measurements for the SMU/SEA and the KDE web sites are remarkably similar to that for other Internet traffic [1, 3, 15], which indicates that web traffic characteristics have remained fairly stable for almost a decade. Although re-confirming these existing results and patterns is not our intention or our focus, this confirmation lends further validity to our primary purpose of using these measurements as part of the data input to evaluate web software reliability.

To summarize, our results demonstrated that workload measures proposed in this paper can be extracted from access logs to capture the overall workload for

these web sites at different levels of granularity and from different perspectives. When used in conjunction with failure measurements, they can provide an objective and stable evaluation of the operational reliability for these web sites' contents. We also showed that a potentially significant reliability gain could be achieved under effective usage-based testing and defect fixing. These results demonstrated the viability and effectiveness of our strategy.

There are also some open issues we plan to address in future studies, including:

- *The impact of web site changes and related fault injections*: Our reliability analyses performed in this paper assumed the stability of the web sites under study, and our evaluation of reliability growth potential additionally assumed that none or few new faults were injected. Therefore, a direct generalization of this study is to study the impact of web changes and injection of new faults on web software reliability.

- *Risk identification for reliability improvement*: The error distribution is highly uneven, as shown in this paper and demonstrated in further studies we performed to examine the error distributions across error types, originators, error sources, page types, etc. [8, 10]. These uneven distributions point out the importance of applying risk identification techniques to identify problematic areas in the future for focused web software reliability improvement.

- *Better ways to count bytes transferred*: Byte counting in this paper ignored about 15% of access log entries with missing information for their "byte transferred" field, which typically correspond to error entries and cached web contents. Treating them as 0's is convenient but runs contrary to the general practice in software reliability engineering [9, 12], where all usage time or activities should be counted regardless of whether the specific usage resulted in successful completions or failures. This fact points to the need for further investigation and possible alternative data treatment when we use bytes transferred for reliability analyses.

In addition, we also plan to identify better existing tools, develop new tools and utility programs, and integrate them to provide better implementation support for our strategy. All these efforts should lead us to a more practical and effective approach to achieve and maintain high reliability for web applications.

Acknowledgments

The work reported in this paper was supported in part by NSF grants 9733588 and 0204345, THECB/ATP grants 003613-0030-1999 and 003613-0030-2001, and Nortel Networks.

We thank our system administrator, Merlin Wilkerson, for his help in gathering and interpreting the web logs for the SMU/SEAS web site, and Dirk Mueller from the KDE project who helped us obtain and interpret the access logs for the KDE web site and deal with various data problems.

We thank Chaitanya Kallepalli and Omar Shahdad for their previous work in developing many of the Perl programs used in this study, and A. Güneş Koru and Li Ma for their help and suggestions.

We also thank the anonymous reviewers whose constructive comments and suggestions led to a much improved paper.

References

[1] M. F. Arlitt and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Trans. on Networking*, 5(5):631–645, Oct. 1997.

[2] B. Behlandorf. *Running a Perfect Web Site with Apache, 2nd Ed.* MacMillan Computer Publishing, New York, 1996.

[3] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Trans. on Networking*, 5(6):835–846, Dec. 1997.

[4] A. L. Goel and K. Okumoto. A time dependent error detection rate model for software reliability and other performance measures. *IEEE Trans. on Reliability*, 28(3):206–211, 1979.

[5] IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. Number STD 610.12-1990. IEEE, 1990.

[6] C. Kallepalli and J. Tian. Measuring and modeling usage and reliability for statistical web testing. *IEEE Trans. on Software Engineering*, 27(11):1023–1036, Nov. 2001.

[7] A. G. Koru and J. Tian. Defect handling in medium and large open source software projects. *IEEE Software*, 21(4):54–61, July 2004.

[8] Z. Li and J. Tian. Analyzing web logs to identify common errors and improve web reliability. In *Proc. IADIS International Conference on e-Society*, pages 235–242, Lisbon, Portugal, June 2003.

[9] M. R. Lyu, editor. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, 1995.

[10] L. Ma and J. Tian. Analyzing errors and referral pairs to characterize common problems and improve web reliability. In *Proc. 3rd International Conference on Web Engineering*, pages 314–323, Oviedo, Spain, July 2003.

[11] A. L. Montgomery and C. Faloutsos. Identifying web browsing trends and patterns. *IEEE Computer*, 34(7):94–95, July 2001.

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

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

[14] J. Offutt. Quality attributes of web applications. *IEEE Software*, 19(2):25–32, Mar. 2002.

[15] J. E. Pitkow. Summary of WWW characterizations. *World Wide Web*, 2(1-2):3–13, 1999.

[16] N. F. Schneidewind. Software reliability model with optimal selection of failure data. *IEEE Trans. on Software Engineering*, 19(11):1095–1104, Nov. 1993.

[17] N. Singpurwalla. Software reliability modeling by concatenating failure rates. In *Proc. 9th Int. Symp. on Software Reliability Engineering*, pages 106–110, Nov. 1998.

[18] R. Thayer, M. Lipow, and E. Nelson. *Software Reliability*. North-Holland, 1978.

[19] J. Tian. Integrating time domain and input domain analyses of software reliability using tree-based models. *IEEE Trans. on Software Engineering*, 21(12):945–958, Dec. 1995.

[20] J. Tian. Better reliability assessment and prediction through data clustering. *IEEE Trans. on Software Engineering*, 28(10):997–1007, Oct. 2002.

[21] K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd Edition*. John Wiley & Sons, Inc., New York, 2001.

7 About the Authors

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.

Sunita Rudraraju received a B.S. degree in Biology from Andhra University, India in 1998, an M.S. degree in Software Engineering from Southern Methodist University, Dallas, Texas in 2001. She worked as a Research Assistant on software reliability analysis at Southern Methodist University during the year of 2001. She worked as a Software Engineer for Fidelity Investments, Westlake, Texas between 2003 and 2004. Since 2004, she has been with Intervoice-Brite as a Software Engineer in the Research and Development Dept.

Zhao Li received the BE and ME degrees in computer science from Xi'an Jiaotong University, China, in 1996 and 2000, respectively. He is currently working toward the PhD degree in computer science at Southern Methodist University, Dallas, Texas. His research area includes software testing, software reliability analysis, and software metrics. He is a student member of the IEEE Computer Society.