Abstract

After examining the specific problems of testing and quality assurance for web-based applications, we propose a strategy by integrating existing testing techniques and reliability analyses in a hierarchical framework. This strategy combines various usage models for statistical testing to perform high level testing and to guide selective testing of critical and frequently used subparts or components using traditional coverage-based structural testing. Reliability analysis and risk identification form an integral part of this strategy to help assure and improve the overall reliability for web-based applications. Some preliminary results are included to demonstrate the general viability and effectiveness of our approach.

keywords: Web usage and modeling, quality and reliability, statistical testing, structural testing, operational profile, Markov chain, test coverage.

1. Introduction

Web-based applications provide cross-platform universal access to web resources for the massive user population. The users typically employ different hardware equipments, network connections, operating systems, middleware and web server support, and web browsers. Although some computational capability has evolved in newer applications, document and information search and retrieval still remain the dominant usage for most web users.

The combination of these unique characteristics that distinguish web-based applications from traditional software systems also make traditional coverage-based testing [11] inappropriate or inadequate for web-based applications. Instead, statistical testing techniques [7, 10] can be used to selectively test those components or usage patterns frequently used by the massive number of users under diverse usage environments. For example, the close resemblance between web-based applications and the state transition mechanism in Markov chains makes statistical testing based on actual usage patterns and frequencies captured in Markov chains a natural choice [3]. On the other hand, loosely related collections of web pages can be more appropriately represented and tested by using some simpler usage models based on a flat list of operations and associated probabilities [7]. The introduction of statistical testing strategies is not to replace traditional testing techniques, but to use them selectively on important or frequently used functions or components.

In this paper, we propose a three-tiered hierarchical strategy for testing web-based applications and for analyzing and assuring their reliability. The top tier is modeled by a flat operational profile; the middle tier is modeled by a set of Markov chains; and the bottom tier is modeled by traditional structural testing models. Analyses based on various software reliability models are integrated into the above three-tiered framework for reliability evaluation, risk identification, and quality improvement.

2. Problems and Existing Approaches

Quality assurance and testing for web-based applications generally focus on the prevention of web failures or the reduction of chances for such failures. We define web failures as the inability to obtain or deliver information, such as documents or computational results, requested by web users. Problems with information delivery often involve host, network, or browser failures, which can be analyzed by various existing techniques and addressed by the “global” web community. User errors may also cause problems, which can be addressed through user education, better usability design, etc. On the other hand, web source or content failures are typically directly related to the services or functions that web-based applications are trying to provide. Therefore, we will focus on web source failures and adapt existing testing techniques and related analyses to ensure reliability, which is defined to be the probability of failure-free operations [7], for web-based applications.

Traditional testing techniques can be classified into two broad categories: black-box (or functional) testing and white-box (or structural) testing. The former ver-
ifies the correctness of external functions, or whether the observed behavior conforms to user expectations or product specifications; while the latter verifies the correct implementation of internal units, structures, and relations among them [1].

Most traditional testing techniques use coverage as the stopping criteria, with the implicit assumption that higher coverage means higher quality or lower levels of defects. On the other hand, product reliability goals can be used as a more objective and more direct criterion to stop testing [7]. The use of this criterion requires the testing to be performed under an environment that resembles actual usage by target customers so that realistic reliability assessment can be obtained, resulting in the so-called statistical usage-based testing [7, 10].

Another important factor that affects the choice of testing techniques is the increasing size and complexity of the software systems, such as the web, which makes many coverage goals infeasible or impractical to achieve. Statistical testing can help us prioritize testing effort based on usage scenarios and frequencies for individual functions and navigation patterns to ensure product reliability. This general strategy can be carried out in three steps:

1. Construct the statistical testing models or usage models based on actual usage scenarios and frequencies.
2. Use these models for test case construction, selection, and execution.
3. Analyze the test results for reliability assessments and predictions, and for decision making.

Two primary types of usage models are: flat operational profiles (OPs) [7] and the Markov chain based models (or Markov OPs) [10]. A flat OP presents commonly used operations, such as frequently visited pages or navigation patterns for web-based applications, in a list, a histogram, or a tree-structure, together with the associated occurrence probabilities. The main advantage of the flat OP is its simplicity, both in model construction and usage.

A Markov usage model presents commonly used operational units in a Markov chain, where the state transition probabilities are history independent. Complete operations or navigation patterns can be constructed by linking various states together following the state transitions (links), and the probability for the whole path is the product of its individual transition probabilities. Most web-based applications consist of various components, stages, or steps, visible to the users, and typically initiated by them. Markov models based on state transitions can generally capture such navigation patterns better than flat OPs.

Recently, we developed unified Markov models (UMMs) to support statistical testing, performance evaluation, and reliability analysis [3]. As compared to individual Markov chains in previous work [10], we used a set of hierarchical Markov chains in our UMMs. The hierarchical nature of our UMMs allows us to achieve state and transition coverage for Markov chains at the top level and frequently used sub-chains near the top level, and more selective testing for other lower level Markov chains. In addition, test efficiency can be improved by avoiding redundant executions once a subpart has been visited already by using lower level models and following different execution paths.

Both these types of OPs will be integrated into our hierarchical strategy below. The results from statistical testing can be analyzed by using various software reliability models.

3. A Hierarchical Strategy

Our hierarchical strategy integrates various existing techniques for software testing and reliability analysis to work for web-based applications. The key components include:

1. Development of the high-level operational profile (OP), which enumerates major functions to be supported by web-based applications and their usage frequencies by target customers. This list-like flat OP will be augmented with additional information and supported by lower level models based on our unified Markov models (UMMs). The additional information includes grouping of related functions and mapping of major external functions to primary web sources or components.

2. For each of the high-level function groups, a UMM can be constructed to thoroughly test related operations and components. Our UMMs capture desired behavior, usage, and criticality information for web-based applications, and can be used to generate test cases to exhaustively cover high-level operations and selectively cover important low-level implementations. The testing results can be analyzed to identify system bottlenecks for focused remedial actions, and to assess and improve system performance and reliability.

3. Critical parts identified by our UMMs can be thoroughly tested using lower level models based on traditional testing techniques. Other quality assurance alternatives, such as inspection, static and dynamic analyses, formal verification, preventive actions, etc., can also be used to satisfy user needs and expectations for these particular areas.

3.1. Obtaining Information for OP Construction

OP accuracy not only affects the testing to be performed, but also affects the reliability of the delivered software product. To maximize OP accuracy, we can combine differ-
ent means to extract appropriate information from different sources using different tools. At the same time, we can try to minimize cost by making effective use of existing information sources, tools, and techniques.

To construct a flat OP, the function list needs to be identified first and then the associated probabilities need to be obtained. To construct a UMM, the basic states and state transitions need to be identified first and then state transition probabilities need to be obtained. Both these steps for both these variations of OP construction can be carried out using expert opinion, customer survey, or actual measurement.

Expert opinion is the starting point for our OPs, which can be cross-validated later on when customer survey and measurement information is obtained. Many of the existing product and system documentations, such as architectural and design documents, technical memos, and documents of understanding with customers and partners, as well as product specifications and relevant standards, also represent expert opinions. From these information sources, high level structures, major components, interconnections among these components, as well as rough usage frequency and importance information, can be obtained.

Customer surveys can provide more objective and potentially more accurate information regarding a product’s usage. However, because of the massive user population for most web-based applications, careful planning is necessary to reduce the survey cost. We can identify a few selected target customers, and conduct detailed surveys of limited scope. In addition, we can perform automated analysis of existing customer feedback information, in connection with similar analyses of existing logs and system records to be discussed later.

Actual measurement provides ultimate accuracy of customer usage scenarios and associated probabilities, although it is also typically the most expensive among the above three alternatives. Fortunately for most web-based applications, existing logs and system records, typically maintained for normal system operations, billing, and other purposes, can be used to construct our OPs. Recently, we have extracted information from existing server access and error logs to construct our UMMs and to evaluate the web site reliability [3].

### 3.2. Model construction based on log analysis

Based on log analyses similar to the one described in [3], we can produce the following reports:

- **Top access report** (TAR) which lists frequently accessed (individual) services or web pages together with their access counts.
- **Call pair report** (CPR) which lists call pairs (transition from one individual service to another) and the associated frequency.

TAR is important because many of the individual services can be viewed as stand alone ones in web-based applications, and a complete session can often be broken down into these individual pieces. This report, when normalized by the total access count or session count, resembles the flat OP by Musa [7]. For example, a variation of TAR called page hit report for web-based applications was obtained from web access logs in [3], giving us the frequently visited pages and the associated hit counts in ranking order. Notice that each service unit in a TAR may correspond to multiple pages grouped together instead of a single page. Such results provide useful information to give us an overall picture of the usage frequencies for individual web pages, but not navigation patterns and associated occurrence frequencies.

CPR connects individual services and provides the basic state transitions and transition probabilities for our UMMs. We can traverse through CPR for strong connections among TAR entries, which may also include additional connected individual services not represented in TAR because of their lower access frequencies or because they represent lower level service units. A UMM can be constructed for each of these connected groups. In this way, we can construct our UMMs from TAR and CPR, in consultation with expert opinion and customer survey results.

Notice that multiple OPs, particularly multiple UMMs in addition to TAR, our top level OP, usually result for a single set of web-based applications using the above approach. This implementation of our integrated strategy in a hierarchical form is discussed below:

- At the top level, TAR can be used directly as our flat OP for statistical usage-based testing, much like in [7].
- Entries in TAR can be grouped according to their connections via CPR, and a UMM can be constructed for each of these groups, forming our middle level usage models, or our individual UMMs.
- The hierarchical nature of our UMMs will allow us to have lower level UMMs as well as other lower level testing models to thoroughly test selected functional areas or web components.

This hierarchical implementation of our integrated strategy is graphically depicted in Figure 1. We focus on testing frequently used individual functions or services at the top level, testing common navigation patterns and usage sequences at the middle level, and covering selected areas at the bottom level. Specific low level UMMs or other coverage-based testing models can be built to thoroughly test the related features or critical components in the higher
level flat OPs or UMMs. Coverage, criticality, and other information can also be easily used to generate test cases using lower level models under our OPs.

3.3. Reliability analysis and improvement

Reliability models and analysis techniques can 1) provide an objective assessment of current product reliability and to determine when to stop testing, inspection, or other quality assurance activities, 2) predict future reliability and time or resources to reach a reliability goal, or 3) identify problematic areas or software components for focused reliability improvement [7, 9]. We integrate these analyses into our hierarchical strategy to support the following:

- **Reliability snapshot and growth over time**: Once $n$ usage-based statistical test cases are executed with $f$ failures, the reliability can be estimated by the average success rate $(n - f)/n$ according to the Nelson model [8], or the weighted success rate for runs of unequal workload [9]. Similar analysis of data from inspection and other quality assurance activities can also be carried out. Such measurement results over time can also be analyzed by software reliability growth models (SRGMs) [7].

- **Bottleneck identification and reliability extrapolation**: Because failures and component reliability can be associated with specific entries in our flat OPs or nodes and links in our UMMs, such information allows us to identify reliability bottlenecks and offers us the flexibility to extrapolate existing reliability analysis results to different configurations or usage probabilities.

- **Reliability composition and improvement**: We are investigating different configurations and designs, so that the system reliability can be maximized for given component reliability. Our tree-based reliability models [9] can be expanded to include UMM information for problem identification and cost-effective reliability improvement.

- **Defect analysis and reduction**: Once defect information is collected, it can be related to various other product and process characteristics for effective problem identification and defect reduction.

3.4. Low-level testing and other QA activities

Multiple variations of testing can be supported by our hierarchical strategy. However, quality assurance for any large software system should not be limited to testing alone. Various other alternatives, such as inspection, formal verification, fault tolerance, etc., also need to be used in conjunction with testing. Our hierarchical strategy can also accommodate these quality assurance activities, as follows:

- Support for software inspection can be provided at two levels: 1) selective inspection of critical web components at the top level by relating frequently used services to specific web components; 2) scenario based inspection guided by our UMMs where usage scenarios and frequencies can be used to select and construct inspection scenarios.

- Selective formal verification can be carried out with the help of our flat OPs or UMMs as well, similar to the way inspection can be aided, as described above. In particular, specific formal verification models can be associated with highly critical parts of UMMs, and much of the information can be shared between lower level UMMs and formal verification models.

- The ability to identify bottlenecks in performance and reliability can also help the selective design and implementation of fault tolerance features. Similarly, many preventative actions, system analyses, damage containment efforts, etc., can also be focused on those critical parts in our hierarchical strategy.

4. Preliminary Results

Our hierarchical strategy above originated from our earlier work summarized in [3]. Various on-going work has
since been reported in [4, 5, 6]. Most of these results are based on using www.seas.smu.edu, the official web site for the School of Engineering and Applied Science, Southern Methodist University (SMU/SEAS), as the testbed. We next summarize some of these preliminary results to demonstrate the general viability and potential effectiveness of our overall strategy.

4.1. Model construction and usage

We examined the characteristics of high level operations, and found that many items are loosely connected. Therefore, independent and individual probabilities used in flat operational profiles are appropriate for high level statistical testing. For example, most of the pages in TARs are index pages for individual academic units within SMU/SEAS, not tightly linked with each other and with little cross-references. The exception is the hierarchical structure reflected, where there are numerous “up” and “down” references between the index page for SMU/SEAS and those for its individual departments and other academic units. Further analysis of expanded list of frequently visited pages follow similar patterns. In addition, the usage frequencies are very unevenly distributed, thus justifying the use of statistical testing and risk identification techniques to focus on high-risk/high-leverage areas.

We investigated the transition from top level operational profiles to middle level Markov chains. We noticed some natural clusters, which would be handled by individual Markov chains. There is a close link and high cross-reference frequencies within a cluster but low visit frequencies across pages from different clusters. For example, within each academic department or unit, there are numerous cross-references but few across boundaries. Consequently, a Markov chain can be associated with each academic unit in statistical testing of this web site, with an additional high-level Markov chain to test the interaction between SMU/SEAS and its academic units.

On the other hand, completely (or nearly completely) isolated operations can be tested by the top level model alone, or if necessary, can bypass the middle level and go directly to the bottom level model for further testing. This latter finding would require minor adjustments to our initially proposed approach and make our three-tiered testing strategy more flexible. In effect, we have modified our model to provide a bypass from top level OP to bottom level structural testing, totally bypassing the middle level Markov chains to form a two-tiered, instead of a three-tiered, implementation.

We analyzed the usefulness of existing testing techniques as low level testing models. Our initial investigation indicates that state-based testing (or finite-state-machine testing) is probably the most appropriate for link-rich pages, with control flow testing more appropriate for link-poor pages. Data flow testing is probably more useful for write-rich contents than for read-only contents. Of course, existing web checkers can continue to be used for some low level testing.

4.2. Error and reliability analysis

We performed various error and reliability analyses in [3]. The overall reliability of our web site was evaluated and shown to remain fairly steady over time. We also investigated the potential reliability growth under the idealized testing environment for statistical web testing, where each observed failure is immediately removed. Analysis showed that statistical web testing over 26 days would have reduced the number of defects by 66.7%, thus improving the reliability by about the same amount — a significant defect reduction and reliability improvement over a relative short time period of 26 days. This result hypothetically demonstrated the effectiveness of our statistical web testing strategy, where the majority of defects could be removed during the simulated testing period of less than one month, resulting in improved reliability for web-based applications.

As followup to the above work, we extended it to include more varieties of workload measures (users, sessions, and bytes transferred, in addition to hits) and to collect measurements over a longer period of time (1 year instead of 26 days). The overall trend and the general picture remained the same. The results expanded the general validity of our previous conclusions and our overall approach to reliability analysis using web logs.

We also performed defect analysis using Orthogonal Defect Classification (ODC) [2] as guideline to identify problematic areas. In addition to error type information we used in our analyses before, we added the use of referral pairs, nesting levels, etc. [4, 6]. The commonly cited 80:20 rule (80% of the problems were caused by 20% of the components) seem to hold for all these individual distributions, where a few error types, error sources, or referrer types, etc., dominate the overall error distributions. This fact points to the importance for risk identification and the great potential for related remedial actions in producing significant quality improvements. We have also started combining these individual analyses to perform joint error analyses. The initial results showed that when using such combined ODC attributes, we can better identify problematic areas for focused remedial actions. These analysis results also led us to perform additional root cause analyses which can be used directly for defect identification and fixing.
4.3. Other preliminary results

To test the suitability of Markov chains to model web usages, we gathered information about web link usage frequencies from web logs. In particular, we used a small set of tests to check the conformance by these actual usage frequencies to the so-called memoryless property that all Markov chains satisfy. The results provided us with empirical evidence that Markov chains can provide fairly accurate models of web usages [5].

We evaluated the overall changes in web structure and contents and analyzed their effect on testing and quality assurance. In particular, we examined the accesses and error observations resulted from changes vs those resulted from defect removal. The results would have a great impact on web evolution and appropriate choice of testing and quality assurance strategies.

As part of the support tools for the implementation of our strategy, we implemented various ad hoc utility programs to support the above individual activities, which will be integrated into a suite of tools in the near future.

We have also started to investigate the potential use of public domain web logs to further validate our initial results.

5. Conclusions and Perspectives

Our three-tiered hierarchical approach described above offers an integrated framework that provides both a potentially effective testing strategy and a vehicle for reliability assessment and improvement. The user focus of web-based applications are supported in this strategy by testing functions, usage scenarios, and navigation patterns important to and frequently used by end users under our top-tier usage model based on the list-like operational profiles as well as our middle-tier usage models based on Markov chains. On the other hand, internal components and structures for web-based applications can also be thoroughly exercised by using our bottom-tier models based on traditional coverage-based testing, under the guidance of the upper-level usage models. Consequently, this strategy provides a possible solution to an important set of problems facing our information society today.

We performed various case studies using the web site for the School of Engineering and Applied Science, Southern Methodist University (SMU/SEAS) as the testbed. The preliminary results from these case studies demonstrated the apparent viability and effectiveness of our hierarchical approach to the testing and quality assurance for web-based applications.

The techniques and tools currently under development in this project for data collection, model construction and application, and result analysis and presentation will provide automated support that is essential for practical implementation and deployment of our hierarchical strategy in industry. Such automated support will magnify the positive impact of technological advancements. Once we have finished the initial viability study under academic settings, we can work closely with our industrial partners to further validate our approach and to deploy it to reap the full benefit of ensuring high reliability and customer satisfaction for many web-based applications.

Acknowledgments

This research is supported in part by NSF grants 9733588 and 0204345, and THECB/ATP grant 003613-0030-2001.

References