Including the Human Factor in Dependability Benchmarks

Aaron B. Brown, Leonard C. Chung, and David A. Patterson
Computer Science Division, University of California at Berkeley
387 Soda Hall #1776, Berkeley, CA, 94720-1776
{abrown,leonardc,pattrsn}@cs.berkeley.edu

Abstract

We describe the construction of a dependability benchmark that captures the impact of the human system operator on the tested system. Our benchmark follows the usual model of injecting faults and perturbations into the tested system; however, our perturbations are generated by the unscripted actions of actual human operators participating in the benchmark procedure in addition to more traditional fault injection. We introduce the issues that arise as we attempt to incorporate human behavior into a dependability benchmark and describe the possible solutions that we have arrived at through preliminary experimentation. Finally, we describe the implementation of our techniques in a dependability benchmark that we are currently developing for Internet and corporate e-mail server systems.

1. Introduction

Dependability benchmarks are a crucial factor in driving progress toward highly reliable, easily maintained computer systems [3] [9]. Well-designed benchmarks provide a yardstick for assessing the state of the art and provide the framework needed to evaluate and inspire progress in research and development. To achieve these goals, benchmarks must be accurate, realistic, and reproducible; in the case of dependability benchmarks, this means that they must evaluate systems against the same set of dependability-influencing factors seen in real-life environments.

One of the most significant of these factors is human behavior. A system's human operators exert a substantial influence on that system's dependability: they can increase dependability via their monitoring, diagnosis, and problem-solving abilities, but they can also decrease dependability by making operational errors during system maintenance. The human error factor is particularly important to dependability: anecdotal data from many sources has suggested that human error on the part of system operators accounts for roughly half of all outages in production server environments [2]. Recent quantitative studies of Internet server sites and of the US telephone network infrastructure numerically confirm the significance of human error as a primary contributor to system failures [4] [12].

Most existing work on dependability benchmarks has ignored the effects of human behavior, positive or negative; this is unfortunate, but perhaps not surprising given that human behavior is typically studied by psychologists or HCI specialists, not systems benchmarkers. In this paper, we present our first steps at bringing consideration of human behavior into the dependability benchmarking world, and describe our work-in-progress toward building a human-aware dependability benchmark. Although our methodology begins with a reasonably traditional dependability benchmarking framework, we expand on existing work by directly including human operators in the benchmarking process as a source of system perturbation.

Humans add significant complications to the benchmarking process, and much of our research focus is on how to include humans while keeping the benchmarks efficient and repeatable. A key insight is to measure the human dependability impact *indirectly*: our benchmarks measure the system, not the human, and we deduce the human dependability impact indirectly through its effects on the system. Other simplifying techniques that we will discuss include approaches for choosing and preparing human operators for our tests (Section 3), selecting human-dependent metrics that can be automatically collected (Section 2), developing an appropriate workload for the human operator (Section 2), and managing the inherent variability introduced by human operators (Section 3). We consider these approaches in the concrete example of an e-mail benchmark in Section 4.

2. Methodology

Traditional dependability benchmarks measure the impact of injected software and hardware faults on the performance and correctness of a test system being subjected to a realistic workload [3] [9]. For example, the system's performance might fall outside its window of normal behavior while it recovers from a hardware fault; the length of the recovery process and the magnitude of the perfor-

mance drop are measures of the system's dependability in response to that fault. Typically, dependability benchmarks are run without human operator intervention in order to eliminate the possible variability that arises when human behavior is involved. But as dependability emerges from a synergy of system behavior and human response, ignoring either component or their interactions significantly limits the accuracy of the benchmark; both system and operator must be benchmarked together.

To accomplish this joint measurement of system and operator, we extend the traditional dependability benchmarking methodology by allowing the human operator(s) to interact with the system during the benchmark. The interaction takes two forms. First, the operator plays an active role in detecting and recovering from injected failures, just as they would in a real environment with real failures. Second, the operator is asked to carry out pre-selected maintenance tasks on the system (for example, backups/restores, software upgrades, system reconfiguration, and data migration), again to simulate real-world operator interaction with the system. We then measure the system's dependability as usual; unlike the traditional approach, the dependability result now reflects the net dependability impact of the human operator, be it positive or negative.

In essence, our approach is to create a standard system dependability benchmark with the human operator as a new source of perturbation, in addition to the standard perturbations injected in the form of software and hardware faults. We can classify the human perturbation based on how it arises. *Reactive perturbation* results from unscripted human action in response to an injected hardware or software fault, and reflects the operator's ability to detect and repair failures. If the operator is quick to respond and recovers the system efficiently, the perturbation will have a positive impact on dependability; if the operator makes mistakes, is slow to respond, or simply fails to respond at all, the impact will be negative. In contrast, *proactive perturbations* arise as the operator performs system maintenance tasks unrelated to failure occurrences. These too can have a negative or positive dependability impact depending on how well the operator performs the task, how many errors are made, and how the maintenance task itself affects the system.

An important change in the benchmark semantics arises when we include proactive perturbations. In traditional dependability benchmarks the system is expected to have a constant level of fault-free dependability; in contrast, with our methodology this baseline can change as the result of maintenance on the system (for example, an "upgrade" maintenance task could increase performance or redundancy). While this complicates the interpretation of benchmark results, it is more indicative of real-world dependability, where maintenance is common. It does require some care in cross-system benchmarking to ensure that similar maintenance is performed on all tested systems.

While the above description of our methodology implies that human operators must participate in the benchmark process, one might wonder if we could simulate the human perturbations and thus eliminate the human. Unfortunately, this reduces to an unsolved problem—if we were able to accurately simulate human operator behavior, we would not need human system operators in the first place! While the HCI community has developed techniques for modeling human behavior in usability tests [5], even in those approaches human involvement is required at least initially to build the model, and the resultant models are typically highly constrained and system-specific, making them inappropriate for use in a comparison benchmark.

Thus we are left with the approach of using live human operators in the benchmarks; this is the only way to truly capture the full unpredictable complexities of the human operator's behavior and the resulting impact on a system's dependability. To flesh out the approach, we must consider how to choose operators for the benchmarks, what maintenance tasks to give them, and what metrics we should use for the final dependability scores. We must also confront the challenges of dealing with human variability, performing valid cross-system comparisons with different operators, and structuring benchmark trials so that the number of human operators is minimized. We discuss workloads and metrics in the balance of this section, and return to the remaining challenges in Section 3.

2.1. Human operator workload

The human operator workload consists of two parts: a pre-specified set of maintenance tasks, and the interactions that arise naturally as the operator repairs injected faults. Since the reactive part of the workload is unscripted and depends on the particular operator's approach to the failures, we do not specify it in advance, and we will not consider it further here.

The pre-specified set of tasks that the operator carries out during the benchmark should be representative of the real-world maintenance carried out in production installations of the type of system under test. Note that we define "maintenance" rather broadly: any *operator* (non-end-user) interaction with the system that is not an immediate reaction to a failure is considered maintenance.

The ideal way to obtain a representative set of maintenance tasks is to carry out a "task analysis" study in which the experimenter shadows real operators as they run a production system similar to that being benchmarked [8]; recording how these operators spend their time provides a list of tasks ranked by importance or frequency. The drawback of task analyses is that they are time consuming and often impractical, especially when the type of system being benchmarked has never been deployed in production.

In such cases, a satisfactory set of maintenance tasks can be selected by an expert familiar with the target system's application domain, using published studies of what system administrators do as a guide [1] [6] [7]. An analysis of these studies suggests a set of general categories of maintenance tasks that apply to most systems:

Initial configuration: setting up new systems, including hardware, operating system, and application installations. Increasing the capacity of an existing system.

Reconfiguration: a broad category covering everything from small configuration tweaks to significant reconfigurations like hardware, OS, or application upgrades.

Monitoring: using monitoring tools or probes to detect failures, security incidents, and performance problems.

Diagnosis and repair: recovery from problems detected by monitoring tasks. Diagnostic procedures, root-cause analysis, and recovery techniques like hardware repairs, software reinstallation/configuration, security incident response, and performance tuning. Unlike "Reconfiguration" tasks, these are unplanned and unscheduled.

Preventative maintenance: non-urgent tasks that maintain a system's integrity, redundancy, and performance, or that adapt the system to changes in its workload.

For an example of how these task categories were specialized for a dependability benchmark for e-mail server systems, see Section 4.

2.2. Metrics

Traditional dependability benchmarks use performance and correctness measures to quantify dependability. Dependability scores are produced by examining how these measures deviate from their expected norms as the system is perturbed by injected faults. We can use this same approach to quantify dependability in our human-aware dependability benchmarks, since we are considering human operator involvement as just another perturbation source to the system. Dependability as measured by this approach reflects the net impact of the human operator: human error that affects performance or correctness will be manifested as reduced dependability, whereas human ingenuity in efficiently repairing problems or performing maintenance will manifest as improved dependability.

The major advantage of this approach is that it vastly simplifies the benchmark process compared to the alternative of trying to directly measure the human impact on dependability. Because there is no need to directly measure human error rate or the dependability impact of individual human actions, the collection of dependability results can be automated. Furthermore, it is easier to design benchmarks for cross-system comparison, as there is no need to match operator actions on one system to equivalent actions on another (often an impossible task). Of course, there is nothing preventing the benchmarker from collecting additional data on human error rates, error severity, or recovery time; such data can prove useful in evaluating a system's *maintainability*, although they are not needed for a dependability evaluation.

3. Reproducible benchmarks with humans

The inherent variability and unpredictability of human behavior makes it a challenge to achieve reproducibility when we include humans in our benchmarks. A crucial part of our human-aware benchmarking methodology is to manage the variance introduced by our human operators, both within a single benchmarking experiment and across benchmark runs on different systems or over time.

Variability in human operators comes from at least three sources. First, different prospective operators will have different backgrounds and base skill levels (compare, for example, an experienced sysadmin to a CS student). Second, operators may have different levels of experience with the system and the benchmark tasks. This is a particularly acute problem when benchmarks are carried out iteratively, as each iteration of the benchmark process increases the operator's experience with the system and can alter his or her behavior on subsequent iterations. Finally, there is a level of inherent variability in human behavior: two operators with identical experience and identical training given identical benchmark tasks may still behave differently.

3.1. Managing variability: single system runs

We propose a two-pronged approach for managing variability in one-off, single-system benchmarks. First, we appeal to statistical averaging, deriving the final dependability result from multiple iterations of the benchmark with different operators participating in each iteration. Second, we attempt to minimize the pre-averaging variability by selecting the participating operators from a set of people with approximately-equal levels of background and experience, and by providing training and support resources to further equalize their knowledge bases.

Results from our pilot studies suggest that between 5 and 20 operators (iterations) will be needed to gain a statistically-sufficient averaging effect; work from the UI community confirms these estimates and suggests that 4 or 5 operators maximizes the benefit/cost ratio [11].

3.1.1. Choosing operators

We can significantly reduce the variance between operator-participants by controlling for their background and skill levels. Because real operators vary greatly in their skills and experience, and because real installations have different demands for operator quality and dependability, we cannot establish a single set of selection criteria for all dependability benchmarks. Instead, we define several

classes of operators, and allow the benchmarker to choose those which best match the target environment of the tested system. With this approach, results from one benchmark run should be comparable to results from other benchmarks using the same class of operators; benchmarks using different classes of operators might also be comparable if the operator level is used as a "handicap" on the results.

We observe at least three classes of benchmark operators (from highest to lowest qualification):

Expert: The operators have intimate knowledge of the target system, unsurpassed skills, and long-term experience with the system. These are operators who run large production installations of the target system for their day jobs, or are supplied by the system's vendor. Benchmarks involving these operators will report the best-case dependability for the target system, but may be realistic only for a very small fraction of the system's potential installed base.

Certified: The operators have passed a test that verifies a certain minimum familiarity and experience with the target system; ideally the certification is issued by the system vendor or an independent external agency. Benchmarks involving these operators should report dependability similar to what would be seen in an average corporate installation of the tested system.

Technical: The operators have technical training and a general level of technical skill involving computer systems and the application area of the target system, but do not have significant experience with the target system itself. These operators could be a company's general systems administration or IT staff, or computer science students in an academic setting. Benchmarks involving these operators will report dependability that is on average similar to that measured with certified operators, but there may be more inter-operator variance and more of a learning curve factor.

Should human-aware dependability benchmarks reach widespread commercial use (like the TPC database benchmarks [15]), they will probably use expert operators. Expert operators offer the lowest possible variance, are unlikely to make naïve mistakes that could make the system look undeservedly bad, yet still provides a useful indication of the system's dependability and maintainability. Published results from benchmarks like TPC often already involve a major commitment of money and personnel on the part of the vendor, so supplying expert operators should not be a significant barrier.

For non-commercial use of dependability benchmarking where experts are unavailable (as in academic or internal development work), using certified operators is ideal since certification best controls the variance between non-expert operators. As it may be difficult to recruit certified operators, it is likely that technical operators will be often be used in practice; we believe that accurate dependability measurements can still be obtained in this case by providing suitable resources and training as described below.

3.1.2. Training operators

We can reduce any remaining variance within a chosen class of operators by using standardized training to bring all operators to the same level of understanding of the test system. It has been our experience that this training must be done at a conceptual level to help the operator build a mental model of the system. The alternative, training on specific tasks that appear in the benchmark, leads to the unrealistic situation of operators follow checklists during the benchmark rather than relying on ingenuity, exploration, and problem-solving, as they would in real life.

Our initial experiments have suggested that an effective method for conceptual training is to first provide a high-level overview of the system's purpose and design, then have the operator carry out a simple maintenance task that requires exploration of the system's interfaces (for example, changing a configuration parameter that is buried deep in an unspecified configuration file or dialog box). If the initial task is well-designed, the operator will have built up enough of a mental model of the system and its interfaces to proceed with the benchmark. With this approach, very little formal training need be given, simplifying the deployment of the benchmark.

3.1.3. Resources for operators

Even with training, different operators may have different gaps in their knowledge that show up during the benchmark. To mitigate the resulting variance, operators should be provided with resources to fill these gaps. Two effective forms of resources are documentation and expert help.

Documentation provides a knowledge base upon which the operator can draw while performing the benchmark tasks. For maximum realism, we believe the operator should be provided with the unedited documentation shipped with the testbed system and be given access to the Internet and its various search engines. If at all possible, the documentation should be provided electronically so that its usage can be monitored automatically.

When documentation fails in real life, operators turn to experts for help. It is important to provide a similar, but standardized, option in the benchmarking process, both to increase realism and to provide an "out" should the operator get stuck or frustrated with a task. We propose to do this by making available a single "oracle" or expert during all runs of the benchmark. The expert must be intimately familiar with the system and the operator tasks; oftentimes the benchmarker can play this role.

A challenge is making the oracle available in such a way that it remains an appeal of last resort; if overused, the oracle becomes the target of the benchmark, not the operator. One approach used successfully in user studies is to make the oracle available via email [16], an approach that

also reduces the demands on the oracle's time. Other possibilities include providing only a limited number of calls to the oracle, imposing an artificial time penalty for using the oracle, or implementing the oracle as an automated "I give up" button that simply completes the current task automatically or restores the system to a known state.

3.2. Managing the learning curve effect

One of the most challenging problems with using live human operators in dependability benchmarks is the *learning curve* effect: at the end of a benchmark iteration, the operator has learned something about the system, and will likely use that experience to perform better on subsequent iterations. This is particularly a problem in cross-system comparison benchmarks or iterative benchmarking of the same system, where the cost of using a fresh set of operators for each system/iteration would be excessive.

Compensating for the learning curve effect is a challenging problem that we are only beginning to address. A simple approach for comparison benchmarks is to randomize the order in which each operator uses the test systems; with a large enough pool of operators, the learning curve effects will be averaged across the systems. Alternately, the effect of the learning curve can be estimated and factored out by benchmarking each system repeatedly until its dependability results stabilize.

For iterative benchmarking of a single system (for example, during system development), other techniques are needed. The most promising approach is to create a system-specific model of human operator behavior, describing how long operators take to respond to problems, what kinds of responses are used, and what kinds of errors are made. Although general simulation of a human operator is intractable, it should be possible to achieve a system- and task-specific model using techniques developed in the HCI community. In particular, models might be created by observing live human operators in an initial benchmark iteration, or perhaps by using expert analysis in a cognitive walkthrough [5]. The benchmarker could use the model to simulate the operators' behaviors in later iterations, either manually or automatically. An open question is how long such a model would remain valid; after major changes to the system or its interfaces, it is likely that the model would need to be rebuilt.

4. An example: benchmarking e-mail

Our first target for evaluating our human-aware dependability benchmarking methodology is e-mail. E-mail has grown from its origins as a best-effort convenience to what is today a mission-critical enterprise service with stringent dependability needs. Surprisingly, no existing e-mail benchmark attempts to quantify dependability.

Our approach follows the general methodology described in Section 2. The benchmark applies a workload, injects perturbations, and collects metrics while the system is under the supervision of a human operator. The benchmark treats the e-mail service as a black-box for generality.

The workload of the benchmark consists of three components: performance, perturbation, and human workloads. The performance workload consists of a realistic simulation of e-mail traffic injected using standard protocols (SMTP and POP3). The simulated workload is based on the SPECmail2001 workload parameters [14] but is fully parameterizable to allow the user to explore system behavior under different load scenarios (for example, load spikes, which are an increasingly relevant dependability threat to Internet services). The perturbation workload has not yet been finalized, but we will likely start with two main types of fault injection: coarse-grained hardware and software faults. For example, we will inject storage system failures (corrupt data, failed disks, timeouts), network failures (corruption, transient connectivity loss, routing anomalies), and OS-level software faults (terminated processes, driver hangs, erroneous return values), among others.

Finally, the human workload consists of maintenance tasks chosen from the categories defined in Section 2.1 and arranged in three steps of increasing difficulty. The first step is a warm-up task consisting of a simple software reconfiguration such as changing the default domain of unqualified e-mail addresses; this step also serves as a "training" step, allowing the operator to become familiar with the system. The second step is a moderately difficult task such as installing and configuring a server-side e-mail virus filter. The third step is a challenging task such as moving a group of users from one server to another.

During each task, the benchmark measures the overall service dependability. Our dependability metrics consist of e-mail delivery delays and errors, the number of dropped/corrupted e-mails, and service performance in fault-free, induced-fault, recovery, and service overload scenarios.

Due to the difficulty of finding certified operators in an academic setting, we intend to use technical-level operators in our benchmark experiments as described in Section 3.1.1. We plan to automate the benchmark as much as possible, including the workload generator and instrumentation. Through these and other techniques, we hope to be able run operators through without a benchmarker present, except perhaps to serve as the on-call oracle.

5. Related work

Our perturbation-based benchmark methodology follows in the footsteps of existing work on dependability benchmarking and extends our earlier work on availability benchmarking, which measured the availability of RAID systems by perturbing them with simulated disk failures

[3]. Our methodology also fits into the dependability benchmarking framework defined by Madeira and Koopman [9], with our human-operator-induced perturbations making up the "upsetload" in their terminology. Where our methodology is unique is in its inclusion of the human operator as a perturbation source: we are not aware of any dependability benchmarks to date that include the human component in their dependability measurements.

Many of the techniques, issues, and proposed solutions in this paper are adaptations of traditional behavioral research techniques for human-computer interaction, such as those described in Landauer's excellent survey [8]. However, unlike the HCI approaches, it is our goal to measure the *system's* behavior rather than the human's—in our benchmarks, the human operator is not there to be directly observed or measured, but to provide realistic perturbation and stimulus to the system. In that sense our work is most similar to work in the security community on the effectiveness of security-related UIs, such as Whitten and Tygar's study of PGP [16]. While we can (and do) borrow advice on topics like selection of operators, task analysis, and experiment logistics from the HCI community, their standard experimental designs and metrics do not directly apply to our dependability benchmarking task.

Finally, our proposed e-mail benchmark differs from other widely-used e-mail benchmarks in that it measures dependability as well as performance. In particular, the two major email benchmarks in production use today (SPEC's SPECmail2001 [14] and Netscape's Mailstone [10]) focus only on performance, do not include facilities for injecting perturbations, and do not measure dependability beyond a simple count of dropped client connections. While some research e-mail systems have been evaluated under simple perturbation (*e.g.*, Porcupine [13]), none have included consideration of the human operator.

6. Conclusions and future directions

As dependability increasingly supplants performance as the essential metric for computer systems, dependability benchmarks are becoming essential tools for system designers and evaluators. Yet to date, dependability benchmarks have ignored the behavior of a computer system's human operators and administrators, a key piece of the dependability puzzle. In this paper we have presented a first attempt at addressing this deficiency: our human-centric benchmarking methodology should provide an effective means of incorporating the effects of human operator behavior into dependability measurements.

What we have presented here is just a first step, however. Our methodology will need to be proven and refined through extensive experimental verification; experimentation will also help explore the extent of cross-operator variability and the tradeoffs involved in issues such as selecting and training operators. We are pursuing this follow-on work in the context of our e-mail dependability benchmark. Other issues that remain to be explored include the development of techniques for pre-evaluating the skill level of participating operators, more advanced dependability metrics that are parameterized by the operator's skill level, and extensions that allow for a direct measure of a system's maintainability and scalability along with the indirect measurements extracted through the dependability metrics. These are all fruitful and important directions for future research, and we look forward to seeing them addressed by the community.

References

[1] Anderson, E. and D. A. Patterson. "A Retrospective on Twelve Years of LISA Proceedings." *Proc. 13th Systems Administration Conference (LISA XIII)*, Seattle, WA, 1999.

[2] Brown, A. and D. A. Patterson. "To Err is Human." *Proc. 1st Workshop on Evaluating and Architecting System dependabilitY (EASY '01)*, Göteborg, Sweden, July 2001.

[3] Brown, A. and D.A. Patterson. "Towards Availability Benchmarks: A Case Study of Software RAID Systems." *Proc. 2000 USENIX Annual Technical Conf.*, San Diego, CA, June 2000.

[4] Enriquez, P. "Failure Analysis of the PSTN." Unpublished talk available at http://roc.cs.berkeley.edu/retreats/spring_02/d1_slides/RocTalk.ppt, January 2002.

[5] Ivory, M. and M. Hearst. "The State of the Art in Automating Usability Evaluation." *ACM Computing Surveys*, 33(4):470–516, December 2001.

[6] Kolstad, R. "1992 LISA Time Expenditure Survey." *;login: the USENIX Association Newsletter*, 1992.

[7] Kolstad, R. "Sysadmin Book of Knowledge." http://ace.delos.com/taxongate.

[8] Landauer, T. K. "Research Methods in Human-Computer Interaction." In *Handbook of Human-Computer Interaction, 2e*, M Helander et al. (ed), Elsevier, 1997, 203–227.

[9] Madeira, H. and P. Koopman. "Dependability Benchmarking: making choices in an n-dimensional problem space." *Proc. 1st Workshop on Evaluating and Architecting System dependabilitY (EASY '01)*, Göteborg, Sweden, July 2001.

[10] Netscape, Inc. *Mailstone Utility.* http://docs.iplanet.com/docs/manuals/messaging/nms41/mailston/stone.htm.

[11] Nielsen, J., and Landauer, T. K. "A mathematical model of the finding of usability problems." *Proc. ACM INTERCHI '93*, Amsterdam, The Netherlands, April 1993, 206–213.

[12] Oppenheimer, D. and D. A. Patterson. "Architecture, operation, and dependability of large-scale Internet services." Submission to *IEEE Internet Computing*, February 2002.

[13] Saito, Y., B. Bershad, and H. Levy. "Manageability, Availability, and Performance in Porcupine: A Highly Scalable Internet Mail Service." *Proc. 17th Symposium on Operating Systems Principles (SOSP '99)*, Kiawah Island, SC, 1999.

[14] Standard Performance Evaluation Corporation. *SPECmail2001*, http://www.spec.org/osg/mail2001/.

[15] Transaction Processing Performance Council Benchmarks. http://www.tpc.org.

[16] Whitten, A. and J. D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0." *Proceedings of the 9th USENIX Security Symposium*, August 1999.