To Err is Human

Aaron B. Brown and David A. Patterson
Computer Science Division, University of California at Berkeley
{abrown,pattrsn}@cs.berkeley.edu

Abstract

We argue for the importance of human error as a significant and oft-overlooked factor in system dependability, and contend that human behavior must be considered in both dependability benchmarks and high-dependability system designs. We support our claims with data illustrating that human-induced failures consistently account for half of all outages, and show that human error accompanies even the simplest system operation and maintenance tasks. We then consider how human behavior might be accounted for in both dependability benchmarks and in the designs of dependable computer systems.

1. Introduction

Dependability is a crucial aspect of modern server systems. As the growth in the Internet increases the direct exposure of data and service delivery to the end user, dependability and its associated metrics of availability, performance, and correctness become real challenges that directly affect users and the corporate bottom line of service providers. Much to the chagrin of those service providers, achieving high dependability is currently a losing battle. Outages remain frequent even among the high-end of service providers, and outage costs remain high in both economic and social terms. A recent survey by InternetWeek revealed that 65% of surveyed sites suffered a customer-visible outage at least once in the previous 6-month period; 25% reported three or more outages during that period [10]. The survey also revealed the extraordinary hourly costs of those outages, ranging from several hundred thousand dollars an hour for e-commerce sites like Amazon.com and Ebay to as much as $6.5 million per hour for an online stockbroker. With all the research in the systems and fault-tolerance communities into building dependable systems, why have we not been able to eliminate these outages entirely?

This work was supported in part by the Defense Advanced Research Projects Agency of the Department of Defense, contract no. DABT63-96-C-0056, the National Science Foundation, grant no. CCR-0085899 and infrastructure grant no. EIA-9802069, and the California State MICRO Program. The information presented here does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

We claim that the answer to this question lies at least partly in the fact that most dependability research has focused on the computer system itself—its hardware and software—and has ignored a crucial determinant of system dependability: the human operator. Human operators play a vital role in system dependability, detecting problems as they arise, diagnosing them, and repairing them to maintain dependability. Unfortunately, as we will illustrate in this paper, human operators rarely perform these tasks perfectly, and often end up being the primary source of system failures and unavailability. Thus, until we can completely eliminate the human by building fully self-maintaining, self-administering systems (a goal which remains far away on the horizon), the performance of human operators will remain a critical component of system dependability.

We believe that it is time for the system dependability community to take the behavior of human operators into account, both in the design of high-dependability systems and in benchmarks for dependability. Dependable systems must be built to tolerate the inevitable errors made by their human operators; dependability benchmarks must measure a system's propensity for inducing human errors and must quantify its resilience to them.

In the remainder of this paper, we will present evidence underscoring the importance of the human operator in system dependability, then will briefly consider how the human might be taken into consideration both in dependability benchmarking and in the design of dependable systems.

2. The importance of human error

Humans make errors; this is a fact of life. Major league baseball players—who are highly trained, carefully selected, paid millions of dollars, and watched by

RAID System	Total Trials	Trials with Human Errors		Human Error Rate	
		fatal errors	any errors	fatal	overall
Windows	35	1	3	2.9%	8.6%
Solaris	33	0	6	0.0%	17.1%
Linux	31	3	7	9.7%	22.6%

Table 1. Human error rates for simple software RAID maintenance task. On each trial, the human operator was asked to identify and repair a single failed disk in a software RAID volume. Five people participated in the experiments, each carrying out between 5 and 9 trials on each of the three RAID systems. Fatal errors represent situations in which data on the RAID volume was lost, whereas the overall error rate includes trials in which the operator made errors but was able to recover without data loss.

thousands of fans—make errors in 1% to 2% of their fielding chances. The probability of human error on even the most straightforward tasks (like reading numerical digits off a screen) is nonzero [4], and during stressful situations, human error rates rise to between 10% and 100% [8]. In a simulator study of the behavior of bridge watchkeepers, who are supposed to prevent maritime accidents (which could result in the loss of life as well as ships), there was a serious error rate of 3.5% [9]. Since computer maintenance scenarios often involve multiple stresses—deadline pressure, system alarms, unexpected repairs at off-hours, poor environmental conditions, and so on—it should come as no surprise that human operator errors during maintenance are a significant source of system outages.

Some simple experiments that we have carried out hint at the seriousness of human error during maintenance [1]. We asked five people to carry out a basic repair task: replacing a failed disk in a software RAID system. All five people were trained on how to perform the repair, and were given step-by-step printed instructions to follow; the repair process was repeated several times for each participant. We computed two measures of human error: the fraction of the trials in which human mistakes resulted in data loss, and the fraction of trials in which the human made a mistake but was able to recover from it. We ran the experiment across three different software RAID systems to get a feeling for the variance in these metrics. Our results are summarized in Table 1. Notice that even on this simple maintenance task with full instructions and in a low-stress setting, our human operators made fatal errors at a rate of up to almost 10%. When we take into account non-fatal errors, the percentages jump to a range of between 8.6% and 22.6%.

This data illustrates two important points about human operator error during maintenance. First, it is inevitable even with good training and on simple tasks. Second, there can be a significant variance in error rates between different systems. This fact is an important motivator for considering humans in any dependability benchmark, as we will discuss in Section 3.

Our software RAID data speaks directly to the propensity of human operators to make errors during maintenance, but only hints at how important such errors are in causing system outages. To understand that connection we can turn to a large collection of existing field data supporting the theory that operator errors play a key role in system failures and outages. In 1999, Oracle reported that half of the product failures that they analyze are due to human error [6]. A study by Murphy and Gent of VAX system failures in 1993 indicated that system management tasks involving human operators were responsible for more than 50% of failures, and that the error rate was rising as hardware and software failures become less important [7]. Gray reports a similar statistic for fault-tolerant Tandem systems studied in the mid-1980s: 42% of failures in these systems were due to system administration errors—again human error [3]. Turning back the clock further, data from the late 1970s reveals that operator error accounted for 50-70% of failures in electronic systems, 20-53% of missile system failures, and 60-70% of aircraft failures [2].

Even if we look at arguably one of the most fault-tolerant systems in existence today—the public switched telephone network (PSTN)—we find significant evidence of human-error-induced failure. A study by Kuhn of switch failure data collected between 1992 and 1994 reveals that, excluding overload outages due to intentional underprovisioning of the system, human errors were responsible for 52% of reported outages and 50% of outage minutes [5]. Of those human-induced outages and outage minutes, roughly half were the direct result of errors made by telephone company personnel during maintenance operations.

What is particularly surprising about this collection of data is that the fraction of human-induced outages has hovered, virtually unchanged, around 50% for the past three decades. This statistic hints at an endemic problem: even as system architectures and implementations have progressed dramatically over the past 30 years, human errors and their consequences have remained the primary cause of failures and outages. Why might this be so? Perhaps the problem of addressing human error is entirely intractable. But this is unlikely; work in the user-interface and HCI communities shows that it is possible to compensate for human error (as do successes in the area of safety-critical systems for heavy industry and air travel). A more plausible explanation is that the problem of human error has simply been ignored by the community of dependable-system designers and researchers: a survey of the recent literature in the area of general-purpose dependable computer system design reveals virtually no research interest in addressing human error.

We claim that human error is not a problem to be left solely to the user interface community, and that it must be addressed in system design as well. For this to happen, we need tools—benchmarks—that can reflect the impact of human error on system dependability, and we need to start considering human error tolerance in our system designs. The next two sections of this paper address these issues of benchmarks and human-error-tolerant design techniques.

3. Incorporating humans into dependability benchmarks

The data in the previous section makes it very clear that humans play an integral role in determining a system's dependability. Humans can increase dependability by diagnosing and repairing system problems, but they can also reduce it by making errors either during repair or during preventative maintenance. As such, any meaningful dependability benchmark must capture this kind of human behavior.

Rather than trying to directly calculate human error rates and estimate their effects on dependability, we believe that the right way to capture human behavior is to treat human operators as integral parts of the tested system and to carry out the dependability benchmark with the involvement of the operators. In this approach, operators are allowed to participate in system repairs during the benchmark; the benchmark should also be set up so that the operators carry out a realistic set of typical maintenance tasks on the system during the course of the benchmark. If the operators increase system dependability by efficiently and correctly performing repairs, then that will be reflected in the resulting dependability score; if the operators decrease dependability by making errors, that dependability impact will also be reflected in the benchmark result.

This approach does have several practical challenges that must be addressed. One is selection of the operators. If the system being benchmarked is an existing, deployed system, this task is trivial: simply use the system's normal day-to-day operations staff. But if the system is new, or under development, this problem is more difficult. The simplest solution is probably the best: use the people best-trained in the system's operation as the human operators in the benchmark. Although this approach raises questions about reproducibility and ease of result comparison across systems, it may be the only practical one, and there is sufficient precedent in the benchmarking community for similar solutions. For example, the well-respected TPC database performance benchmarks face a similar problem in that human administrators are needed to configure and tune the tested database systems; TPC's solution is that the benchmarks are typically carried out with the database vendor supplying its best database administrators to configure and tune the tested systems. We are simply advocating a similar approach to obtaining operators for dependability benchmarks; just as the best database administrators should be able to tune a DBMS to provide its optimal upper-bound performance, the behavior of the best operators during a dependability benchmark should provide a lower bound on the potential human dependability impact.

There is also the issue of using humans at all during the benchmark. Most traditional performance benchmarks are fully automated and run without human involvement. It would be ideal to have dependability benchmarks that are equally automated and easy to use, but since the human operator plays such a crucial role in affecting system dependability, at least for now there seems to be no alternative to including the human in the benchmark loop.

Although human participation in benchmarking is new to the systems community, it is well-established in other communities of computer science, particular human-computer interaction. We believe that existing techniques for experimental design can be migrated across these community boundaries, and that systems researchers can learn how to carry out experiments with human participation. Indeed, our initial study of human error in software RAID system maintenance (described above in Section 2) demonstrates that even simplistic experiments conducted by non-experts can provide useful results in quantifying human error behavior and suggesting system improvements [1]. Moreover, until we have sophisticated modeling technology that can mimic human maintenance behavior and realistically predict mistakes, or until systems become more self-maintaining and self-administering, the need for human operators in both real life and dependability benchmarking will remain.

Finally, note that it might be possible to partially address the automation of dependability benchmarks by splitting them into two phases, an initial human-dependent phase in which human errors are measured, and a second automated phase in which those errors are simulated and injected into systems along with the standard benchmark workload. In his classic book on human error, Reason states that errors can be categorized as skill slips, memory lapses, or planning mistakes [9]. By extending this categorization to the examples of operator error collected during the first phase of the benchmark, we can use the results as guidelines for the types of errors to simulate in the second phase. If we can also create systems that allow recovery from at least some types of human error (see the next section), then the second phase of the benchmark becomes feasible, and evaluating operator success in recovering from typical human-induced faults should become as viable as evaluating recovery from hardware and software faults.

4. Building dependable human-operated systems

Benchmarks will help us evaluate our progress toward dependable human-operated systems. But in addition to a measurement technology, we also need to think about ways to design human-operated systems so that they can be robust to human errors during operation and maintenance. As in most issues of dependable systems, there are two basic approaches. The first approach is one of *avoidance*: we can try to reduce the rate of human errors by improving system interfaces, providing better task guidance during maintenance, and providing better operator training. This is a traditional approach, and unfortunately one that has not seen dramatic success in practice as evidenced by the rather stable rate of human-induced outages described in Section 2.

There are, perhaps, novel techniques that could be used to improve avoidance mechanisms. For example, systems could provide on-line operator training in the form of unannounced “fire-drill” tests in which the system simulates a realistic failure and requires the operator to repair it inside of an isolated sandbox that protects against errors. Such on-line training would improve an operator’s hands-on familiarity with the failure modes, interfaces, and recovery procedures of the system, and would allow operators to learn from their mistakes in a safe environment where the impact of those mistakes is minimized.

But even the most sophisticated avoidance mechanisms can only go so far; as we saw in Section 2, humans still make mistakes despite training, and stress can drive up error rates on even the simplest tasks with the best user interfaces. These observations motivate the second approach to human error, *tolerance*: we can design systems so that they are tolerant of human errors, and work to minimize the impact of human error on overall system dependability. This approach, while not unusual in hardware fault-tolerance, is rather radical when applied to human error-tolerance. Few if any systems today provide maintenance interfaces that are tolerant of error, that provide effective and rapid recovery paths from arbitrary human mistakes. The concept of *undo*—ubiquitous in the productivity applications that we use every day—is rarely found in system maintenance, yet it is clear that undo matches well with human error-proneness and our natural desire to experiment with possible actions before committing to them. We claim that the approach of tolerance—perhaps implemented as an undo for system maintenance actions—may be the only one that will provide measurable dependability gains in practice, as human errors (even with the best user interfaces and training) are a fundamental inevitability of life. We are currently pursuing research into the appropriate semantics and implementation of maintenance undo in the context of Internet server systems.

Of course, the approaches of avoidance and tolerance are complimentary; even a system tolerant of human errors will benefit from human-error-avoidance mechanisms like improved user interfaces. An avoidance system is essentially an optimization; it reduces the day-to-day pressure on the tolerance mechanisms, allowing them to be implemented with simpler, more reliable, but potentially more resource-intensive techniques. But tolerance is still fundamental, required to handle the errors that inevitably slip through.

5. Conclusion

The traditional focus on hardware and software as the only important aspects of dependable systems has led to a state in which human mistakes have become the largest single source of system failures and outages. We have argued for a reevaluation of these traditional priorities and for a new focus on human error. We are at the infancy of dependability benchmarking, offering us a unique opportunity to incorporate consideration of human behavior from the start; we believe that it is essential to seize this opportunity lest we miss our chance to develop realistic dependability benchmarks that reflect the reality of today's human-error-filled environment. Furthermore, we believe that the time is long overdue to consider human error in the design of dependable systems. By building systems that acknowledge human error and tolerate it, we can break free of the traditional focus on incremental hardware and software improvements, and once again begin making significant advances in system dependability.

Acknowledgements

The authors wish to express our gratitude to Armando Fox, James Hamilton, Jim Gray, John Chuang, and the Berkeley ISTORE research group for the stimulating discussions that inspired this work. We also wish to thank Eric Anderson for his considerable contributions in collecting the software RAID human error data, and finally our anonymous reviewers for their insightful commentary and suggestions.

References

[1] A. Brown. Towards Availability and Maintainability Benchmarks: A Case Study of Software RAID Systems. *UC Berkeley Computer Science Division Technical Report UCB//CSD-01-1132*, Berkeley, CA, January 2001.

[2] J. M. Christensen and J. M. Howard. Field Experience in Maintenance. *Human Detection and Diagnosis of System Failures: Proceedings of the NATO Symposium on Human Detection and Diagnosis of System Failures*, J. Rasmussen and W. Rouse (Eds.). New York: Plenum Press, 1981, 111–133.

[3] J. Gray. Why Do Computers Stop and What Can Be Done About It? *Symposium on Reliability in Distributed Software and Database Systems*, 3–12, 1986.

[4] B. H. Kantowitz and R. D. Sorkin. *Human Factors: Understanding People-System Relationships*. New York: Wiley, 1983.

[5] D. R. Kuhn. Sources of Failure in the Public Switched Telephone Network. *IEEE Computer* 30(4), April 1997.

[6] J. Menn. Prevention of Online Crashes is No Easy Fix. *Los Angeles Times,* 2 December 1999, C-1.

[7] B. Murphy and T. Gent. Measuring System and Software Reliability using an Automated Data Collection Process. *Quality and Reliability Engineering International*, 11:341–353, 1995.

[8] R. H. Pope. Human Performance: What Improvement from Human Reliability Assessment. *Reliability Data Collection and Use in Risk and Availability Assessment: Proceedings of the 5th EureData Conference*, H.-J. Wingender (Ed.). Berlin: Springer-Verlag, April 1986, 455–465.

[9] J. Reason. *Human Error*. Cambridge University Press, 1990.

[10] T. Sweeney. No Time for DOWNTIME—IT Managers feel the heat to prevent outages that can cost millions of dollars. *InternetWeek*, n. 807, 3 April 2000.