

How Complex Systems Fail

John Allspaw

WHEN I FIRST READ DR. RICHARD COOK'S ESSAY, "HOW COMPLEX SYSTEMS FAIL," I WAS CONVINCED THAT HE HAD WRITTEN THE PIECE SPECIFICALLY FOR THE FIELD OF WEB OPERATIONS. It is a concise and insightful article on the observations of failure and degradation that could have been inspired by years of experience running the distributed architectures for high-volume web companies. As I read through the observations in the essay, I kept remembering events in my own experience troubleshooting degradations and failures that happen when you operate a complex and fast-growing web application.

As it turns out, Dr. Cook is a physician, educator, and researcher at the University of Chicago. He really nailed this topic that spans across many different disciplines and fields. Whether it's structural engineering, space exploration, or patient safety in the medical field, complex systems all have commonalities when it comes to failure. Web operations is one such field.

How Complex Systems Fail

(Being a Short Treatise on the Nature of Failure; How Failure Is Evaluated; How Failure Is Attributed to Proximate Cause; and the Resulting New Understanding of Patient Safety)

By Richard I. Cook, MD, Cognitive Technologies Laboratory, University of Chicago

1. Complex systems are intrinsically hazardous systems

All of the interesting systems (e.g., transportation, healthcare, power generation) are inherently and unavoidably hazardous by their own nature. The frequency of hazard exposure can sometimes be changed, but the processes involved in the system are themselves intrinsically and irreducibly hazardous. It is the presence of these hazards that drives the creation of defenses against hazards that characterize these systems.

2. Complex systems are heavily and successfully defended against failure

The high consequences of failure lead, over time, to the construction of multiple layers of defense against failure. These defenses include obvious technical components (e.g., backup systems, “safety” features of equipment) and human components (e.g., training, knowledge), but also a variety of organizational, institutional, and regulatory defenses (e.g., policies and procedures, certification, work rules, team training). The effect of these measures is to provide a series of shields that normally divert operations away from accidents.

3. Catastrophe requires multiple failures—single-point failures are not enough

The array of defenses works. System operations are generally successful. Overt catastrophic failure occurs when small, apparently innocuous failures join to create opportunity for a systemic accident. Each of these small failures is necessary to cause catastrophe, but only the combination is sufficient to permit failure. Put another way, there are many more failure opportunities than overt system accidents. Most initial failure trajectories are blocked by designed system safety components. Trajectories that reach the operational level are mostly blocked, usually by practitioners.

4. Complex systems contain changing mixtures of failures latent within them

The complexity of these systems makes it impossible for them to run without multiple flaws being present. Because these are individually insufficient to cause failure they are regarded as minor factors during operations. Eradication of all latent failures is limited primarily by economic cost, but also because it is difficult before the fact to see how such failures might contribute to an accident. The failures change constantly because of changing technology, work organization, and efforts to eradicate failures.

5. Complex systems run in degraded mode

A corollary to the preceding point is that complex systems run as broken systems. The system continues to function because it contains so many redundancies and because people can make it function, despite the presence of many flaws. After-accident reviews nearly always note that the system has a history of prior “proto-accidents” that nearly generated catastrophe. Arguments that these degraded conditions should have

been recognized before the overt accident are usually predicated on naïve notions of system performance. System operations are dynamic, with components (organizational, human, technical) failing and being replaced continuously.

6. Catastrophe is always just around the corner

Complex systems possess potential for catastrophic failure. Human practitioners are nearly always in close physical and temporal proximity to these potential failures—disaster can occur at any time and in nearly any place. The potential for catastrophic outcome is a hallmark of complex systems. It is impossible to eliminate the potential for such catastrophic failure; the potential for such failure is always present by the system's own nature.

7. Post-accident attribution to a “root cause” is fundamentally wrong

Because overt failure requires multiple faults, there is no isolated “cause” of an accident. There are multiple contributors to accidents. Each of these is necessarily insufficient in itself to create an accident. Only jointly are these causes sufficient to create an accident. Indeed, it is the linking of these causes together that creates the circumstances required for the accident. Thus, no isolation of the “root cause” of an accident is possible. The evaluations based on such reasoning as “root cause” do not reflect a technical understanding of the nature of failure, but rather the social, cultural need to blame specific, localized forces or events for outcomes.¹

8. Hindsight biases post-accident assessments of human performance

Knowledge of the outcome makes it seem that events leading to the outcome should have appeared more salient to practitioners at the time than was actually the case. This means that *ex post facto* accident analysis of human performance is inaccurate. The outcome knowledge poisons the ability of after-accident observers to re-create the view of practitioners before the accident of those same factors. It seems that practitioners “should have known” that the factors would “inevitably” lead to an accident.² *Hindsight bias remains the primary obstacle to accident investigation, especially when expert human performance is involved.*

¹ Anthropological field research provides the clearest demonstration of the social construction of the notion of “cause” (cf. Goldman, L. [1993], *The Culture of Coincidence: accident and absolute liability in Huli*, New York: Clarendon Press; and also Tasca, L. [1990], *The Social Construction of Human Error*, unpublished doctoral dissertation, Department of Sociology, State University of New York at Stonybrook).

² This is not a feature of medical judgments or technical ones, but rather of all human cognition about past events and their causes.

9. Human operators have dual roles: as producers and as defenders against failure

The system practitioners operate the system in order to produce its desired product and also work to forestall accidents. This dynamic quality of system operation, the balancing of demands for production against the possibility of incipient failure, is unavoidable. Outsiders rarely acknowledge the duality of this role. In non-accident-filled times, the production role is emphasized. After accidents, the defense against failure role is emphasized. At either time, the outsider's view misapprehends the operator's constant, simultaneous engagement with both roles.

10. All practitioner actions are gambles

After accidents, the overt failure often appears to have been inevitable and the practitioner's actions as blunders or deliberate willful disregard of certain impending failure. But all practitioner actions are actually gambles, that is, acts that take place in the face of uncertain outcomes. The degree of uncertainty may change from moment to moment. That practitioner actions are gambles appears clear after accidents; in general, analysis regards these gambles as poor ones. But the converse—that successful outcomes are also the result of gambles—is not widely appreciated.

11. Actions at the sharp end resolve all ambiguity

Organizations are ambiguous, often intentionally, about the relationship between production targets, efficient use of resources, economy and costs of operations, and acceptable risks of low- and high-consequence accidents. All ambiguity is resolved by actions of practitioners at the sharp end of the system. After an accident, practitioner actions may be regarded as “errors” or “violations,” but these evaluations are heavily biased by hindsight and ignore the other driving forces, especially production pressure.

12. Human practitioners are the adaptable element of complex systems

Practitioners and first-line management actively adapt the system to maximize production and minimize accidents. These adaptations often occur on a moment-by-moment basis. Some of these adaptations include (1) restructuring the system in order to reduce exposure of vulnerable parts to failure; (2) concentrating critical resources in areas of expected high demand; (3) providing pathways for retreat or recovery from expected and unexpected faults; and (4) establishing means for early detection of changed system performance in order to allow graceful cutbacks in production or other means of increasing resiliency.

13. Human expertise in complex systems is constantly changing

Complex systems require substantial human expertise in their operation and management. This expertise changes in character as technology changes, but it also changes because of the need to replace experts who leave. In every case, training and refinement of skill and expertise is one part of the functions of the system itself. At any moment, therefore, a given complex system will contain practitioners and trainees with varying degrees of expertise. Critical issues related to expertise arise from (1) the need to use scarce expertise as a resource for the most difficult or demanding production needs and (2) the need to develop expertise for future use.

14. Change introduces new forms of failure

The low rate of overt accidents in reliable systems may encourage changes, especially the use of new technology, to decrease the number of low-consequence but high-frequency failures. These changes maybe actually create opportunities for new, low-frequency but high-consequence failures. When new technologies are used to eliminate well-understood system failures or to gain high-precision performance they often introduce new pathways to large-scale, catastrophic failures. Not uncommonly, these new, rare catastrophes have even greater impact than those eliminated by the new technology. These new forms of failure are difficult to see before the fact; attention is paid mostly to the putative beneficial characteristics of the changes. Because these new, high-consequence accidents occur at a low rate, multiple system changes may occur before an accident, making it hard to see the contribution of technology to the failure.

15. Views of “cause” limit the effectiveness of defenses against future events

Post-accident remedies for “human error” are usually predicated on obstructing activities that can “cause” accidents. These end-of-the-chain measures do little to reduce the likelihood of further accidents. In fact, that likelihood of an identical accident is already extraordinarily low because the pattern of latent failures changes constantly. Instead of increasing safety, post-accident remedies usually increase the coupling and complexity of the system. This increases the potential number of latent failures and also makes the detection and blocking of accident trajectories more difficult.

16. Safety is a characteristic of systems and not of their components

Safety is an emergent property of systems; it does not reside in a person, device, or department of an organization or system. Safety cannot be purchased or manufactured; it is not a feature that is separate from the other components of the system. This means that safety cannot be manipulated like a feedstock or raw material. The state of safety in any system is always dynamic; continuous systemic change ensures that hazard and its management are constantly changing.

17. People continuously create safety

Failure-free operations are the result of activities of people who work to keep the system within the boundaries of tolerable performance. These activities are, for the most part, part of normal operations and superficially straightforward. But because system operations are never trouble free, human practitioner adaptations to changing conditions actually create safety from moment to moment. These adaptations often amount to just the selection of a well-rehearsed routine from a store of available responses; sometimes, however, the adaptations are novel combinations or *de novo* creations of new approaches.

18. Failure-free operations require experience with failure

Recognizing hazard and successfully manipulating system operations to remain inside the tolerable performance boundaries requires intimate contact with failure. More robust system performance is likely to arise in systems where operators can discern the “edge of the envelope.” This is where system performance begins to deteriorate, becomes difficult to predict, or cannot be readily recovered. In intrinsically hazardous systems, operators are expected to encounter and appreciate hazards in ways that lead to overall performance that is desirable. Improved safety depends on providing operators with calibrated views of the hazards. It also depends on providing calibration about how their actions move system performance towards or away from the edge of the envelope.

As It Pertains Specifically to Web Operations

1. It will be difficult to tell that the system has failed

Overt failures such as server fires or a backhoe digging up the fiber optic cable are dramatic, easily detected, and—because they are well defended against—quite rare. Much more common are failures that degrade performance a little (or a lot) or cause portions of a system to stop working. Especially for distributed systems, central indicators may be nominal even though service is effectively down for some or even many users. For systems that involve server-side applications and lots of user input, failures may seem to be cases of “user error” and early reports of failure may be discounted because user-specific problems are so common.

2. It will be difficult to tell what has failed

Because complex systems involve tangled and shifting dependencies, the symptoms of failure are often nonspecific. This is a version of the notion of “action at a distance”; the consequences of failure are likely to be separated in time and space from the causes. When such a failure occurs, diagnosis may require very high levels of expertise

to untangle the skein of components. This is problematic in the operational context because the people who are operating the system when it fails usually know little about the structure itself.

3. Meaningful response will be delayed

The combination of #1 and #2 will produce delays in responding to failures and these delays will be amplified by difficulties in communications related to diagnosis and repair.

4. Communications will be strained and tempers will flare

The people experiencing the failure (system users directly, system operators indirectly) will describe the problem in their own terms while the experts searching for the source of the problem employ an entirely different language. The result is that diagnosis typically involves repeated attempts to describe the problem and its context, often in a situation of escalating emotion. Especially for large systems, a deadly loss of information can occur as early reporters become frustrated with the process and leave. The problem reporters are often customers and their dissatisfaction with the service that has failed for them is magnified by the inability of the problem handlers to do more than ask for more information. Nonoperational management will eventually hear about the operational problem and ask questions. Operations people will find themselves dealing with on one side the customer and point-of-contact representatives and on the other side nonoperational management, each demanding information and action. Pressure will escalate with time and some individuals will become angry.

5. Maintenance will be a major source of new failures

Hardware and software maintenance including routine maintenance and upgrades will cause many failures, directly or indirectly. “Simple” failures with catastrophic consequences induced by maintenance procedures are a recurring theme in IT. There are at least two reasons for this. First, because they fiddle with live systems, maintenance procedures themselves are inherently dangerous. Maintenance almost always involves some sort of suspension of operations with a subsequent restart, and starting complex systems is a recipe for the discovery of new dependencies. Second, although patches and upgrades may receive extensive testing before installation, it is virtually impossible to re-create the operating environment in enough detail for the testing to cover all the modes of failure available to the real, much more complex, system. Every maintenance activity is, ultimately, an experiment. Experienced owners of large, complex systems are reluctant to perform maintenance or upgrades and this may lead to a kind of managerial paralysis: a working system, even one with known flaws, is a precious entity and so maintenance and, especially, upgrades are deliberately delayed because of fear of the failures that come from work on the system.

6. Recovery from backup is itself difficult and potentially dangerous

Although backing up and its varieties (e.g., system checkpoints) is widely recognized as a good practice, actually using backups to recover from a complex system outage is relatively rare. The delays associated with #1–#4 above often contrive to make the most recent backup old enough that using it will lead to important data loss. Restoring from backup is a maintenance-like activity that is itself inherently hazardous, especially because it is an unfamiliar procedure, often one involving steps that the operators have never performed before. Not all managers recognize the complexity and uncertainty that accompanies a system restoration from backups. This can lead to pressure to use backups to restore system operations even before the cause of the outage has been determined.

Given all this, what should operations people do? Good system design is the best defense, of course, but operational people have to play the hand that the designers dealt them. There are a few things that can make operations easier.

7. Create test procedures that front-line people can use to verify system status

Make it easy for point-of-contact and operations personnel to test the system directly. For order entry systems that include credit card processing, for example, a useful test procedure might be a script that uses the system to create and place a real order, including credit card processing with a real credit card and e-mail notification. Remote monitors of system performance may not show that the system is down. Fast, successful transaction processing, including all the ancillary processes, is good evidence that it is up. Conversely, slow processing or failure is strong evidence that something is wrong.

8. Manage operations on a daily basis

It's useful to practice communicating about system functional problems with the experts you will call on when an outage occurs. People—including experts—change jobs, go on vacation, call in sick, and go to lunch. Know who is available today and how to contact them. Formal printed lists or web-based call schedules are wonderful but not always current. An index card with the names, phone numbers, and e-mail addresses of the people you might need to call if the system goes down fits in the pocket and can be handed to a deputy when you go to the dentist. Don't forget to include the name of the maintenance guy with the keys and the director of marketing who is always carping about how important uptime is. It's also quite interesting to call these people every once in a while as a test and as reinforcement regarding their potential roles. The reverse side of the index card is useful for writing down short notes about the time and nature of issues that arise during the day. If you save each day's card in a box you will soon have a nice operational log for the system. (Although

there are many computer-based approaches to this sort of record, the manual method described is a good method for managing the difficult-to-categorize stuff of day-to-day operations that is hard to track with such apps.)

9. Control maintenance

Operational people don't usually perform software or hardware maintenance beyond routine backups, but they inherit all the problems that maintenance generates. Operations should control maintenance activities and regulate access to the running system. When maintenance is needed, access to the running system should be provided by operations and all maintenance activities should be examined by operations before access is granted. Systems people, especially programming types, are sometimes overconfident about the maintenance code and procedures they have generated. Maintenance activities need planning and careful observation from operations for two reasons. First, this sort of oversight makes explicit the changes and justification for them and generates records of the things that can be used when tracing back problems that appear in the system. Second, the first people to recognize that a new problem has appeared are the operations folks and they benefit from knowing when and how the system has been altered. There is sometimes resistance from the software side to this kind of oversight, in part because of the belief that operations personnel will blame maintenance for any new problem that appears. This attitude can even lead to attempts to do maintenance "on the sly," that is, without letting the operations side know about it. There is sometimes a temptation to just "slip in this one little change" rather than go through a procedure that involves scheduling and notification and what some may consider bureaucratic process. Operations and not systems programming should control access to the system and oversee all the maintenance.

10. Assess performance at regular intervals

Operations can be hectic so that routine stuff gets put off indefinitely. Hints of future problems can often be found in current performance and regular reviews of that performance is the best way to detect these warnings. Especially with distributed systems it can be hard to get a handle on performance, but this only means that it is even more important to look at it closely. A good deal of attention has been given to performance *measurement*—generally this means the automated tools that examine system logs and network hardware. But it is performance *assessment*—the examination and analysis of larger systems—that really matters. The performance measurement tools are only one source of information about system performance. The assessment is the detailed analysis of what these tools reveal in the context of operations. Such analysis is not done in a few minutes and often requires a good deal of data not gathered by the measurement tools, (e.g., transactions processed, the volume of transactions at different times of the day or week, relationships between measured values and recent maintenance). Pay particular attention to things that might become bottlenecks under slightly different circumstances. Would a 10% increase in transaction rates lead to a full scratch

volume? Is the pattern of user login durations suggestive of thoughtful users or poor system responsiveness? The goal of operations is to have every day be just another boring day. Achieving this boredom depends on foreseeing the future performance of the system and making adjustments accordingly.

11. Be a (unique) customer

The easiest way to determine that your system works is to use it as a customer would. If you can get information and perform transactions successfully it is likely that others can as well. You should be a regular user of the system and exercise its functions on a daily basis. Something taking too long? A page not displaying correctly? Some link is 404? It's far better for an operations person to identify these sorts of problems than to have a customer do it. Remember, not all browsers cooperate well with web-based services. Keep that old laptop and put Linux and a couple of different browsers on it for test purposes. Sure, your system works with the latest version of Explorer, but how about an older version of Opera? Stop thinking of Apple users as super-geeks and start thinking of them as customers—if it won't work with Safari the people who use only that won't be using your site. Again, although others in your organization are supposed to take care of these issues, it is operations that will handle the calls from the field when they don't.

Further Reading

Cook, Richard I., Marta Render, and David D. Woods (2000). "Gaps in the continuity of care and progress on patient safety." *British Medical Journal* 320: pp. 791–4.

Cook, Richard I. (1999). "A Brief Look at the New Look in error, safety, and failure of complex systems" (Chicago: CtL).

Woods, David D., and Richard I. Cook (1999). "Perspectives on Human Error: Hindsight Biases and Local Rationality." In Durso, Nickerson, Dumais, et al., eds., *Handbook of Applied Cognition* (New York: Wiley); pp. 141–171.

Woods, David D., and Richard I. Cook (1998). "Characteristics of Patient Safety: Five Principles that Underlie Productive Work" (Chicago: CtL).

Cook, Richard I., and David D. Woods (1994). "Operating at the Sharp End: The Complexity of Human Error." In M.S. Bogner, ed., *Human Error in Medicine* (Hillsdale, NJ); pp. 255–310.

Woods, David D., Leila J. Johannesen, Richard I. Cook, and Nadine B. Sarter (1994). *Behind Human Error: Cognition, Computers and Hindsight* (Wright Patterson AFB: CSERIAC).

Cook, Richard I., David D. Woods, and Charlotte Miller (1998). "A Tale of Two Stories: Contrasting Views of Patient Safety" (Chicago: NPSF); available as a PDF file on the NPSF website at <http://www.npsf.org>.