Tales of Terror

October 2000

Gather 'round the campfire and let your flashlights cast spooky shadows. You're about to encounter hair-raising horror in the form of performance problems, poor planning and skipped specifications …

You've probably heard of spectacular software failures like the Denver Airport baggage system, the attempt to automate welfare in the state of Florida and the Internal Revenue Service's many IT project mishaps, as popularized in Robert L. Glass' *Software Runaways* (Prentice Hall, 1998). This Halloween season, we invited *Software Development's* contributing editors, editorial advisors and writers to send us their scary stories. Whether due to misguided managers, technological risk or flat-out incompetence, everyone has experienced a failure they can share—and most were brave enough to sign their names. Have fun rubbernecking, but do take heed, lest your project become the next casualty!

Metric Madness

I was working on a large network management system when one of our managers was swallowed by our methodology group's software metrics seminar. We feared the worst, because few attendees ever came back unscathed. Consistently, they returned with just enough knowledge to make them dangerous to their teams. Sure enough, our worst nightmares became reality when our manager began repeating the phrase "you can't control what you can't measure." We shuddered, knowing that a good idea was about to be distorted into an instrument of evil.

Sure enough, within a week of falling under the methodology group's spell, our manager wrote a little program to measure our productivity. The metric chosen was the much vilified source lines of code count (SLOC). Each week he would run his program to create a baseline count for our productivity. Then he began to measure our individual performance against the baseline. If your individual line count fell below the baseline, you were called into his den to explain why your productivity had fallen off. If you were above the baseline, you might receive a commendation.

One coder, known for his "cut-and-paste" style, received a pair of hockey tickets for being our most productive programmer. A colleague who was able to consolidate a large volume of duplicate code, on the other hand, was called into the manager's den because the consolidation gave him a negative line count for that week. When he emerged, he wasn't quite the same.

We knew that we had to protect ourselves. Otherwise, we were destined to suffer my colleague's fate and not receive our bonuses. We knew our manager's program calculated source lines of code by counting semicolons. Soon all statements were terminated with double semicolons:

```
Alarm a = Alarm.mkAlarm(cos);;

a.ack();;
```

Our manager was ecstatic—believing that his effective use of metrics had raised the group's productivity. We began to push the limits of good taste, using semicolons for block comments:

```
/*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Your comment goes here

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*/
```

Soon our group was cited as an example of how the effective use of metrics had led to a dramatic improvement in productivity. Our manager was proud of us … until our little subterfuge was discovered in a code audit.

The moral of this story is: Be careful when you choose a metric, especially in how you interpret and act on it. There is no quicker way to turn a metric into an instrument of evil than to base a programmer's compensation on it. Forget this moral and you may be smitten by "metric madness," that short-lived euphoria experienced when you believe the world is wonderfully productive, followed by shock and surprise when you discover that you have chosen the wrong metrics, or grossly misinterpreted them. You have been warned.

—Steve Adolph
Senior Technical Consultant
WSA Consulting Inc.
Britannia Beach, British Columbia

The Perfect Job

In 1990, when DOS dinosaurs still ruled the earth and software jobs were hard to come by, I found a job listing in the classifieds. It sounded like they were doing great stuff: CAD and drafting, three-dimensional solid modeling, walkthroughs and even animation. The R&D director and his right-hand engineer were enthusiastic, brilliant, sarcastic and fun. (Still true and still two of my closest friends.)

And the management? Later, later. They showed me their software. They showed me a beta of Microsoft Windows 3.0. I was goggle-eyed, on fire. None of us noticed the transition from "if you worked here" to "when you start Monday."

Although he stumbled a little over technical details, the president was a good-looking smooth talker who uttered phrases like "growing the company" and "new marketing." The owner lived in Switzerland, and the company had recently been renamed. Something about the conversation rang a little funny, but who cared? They were doing great stuff here!

They offered. I leaped. I roared into their code and met the other engineers. These people were terrific! We'd eat Autodesk's lunch!

Except … the company had been renamed not once, but four or five times. And the owner liked to dodge debt by sucking the company assets into Switzerland, bankrupting on the U.S. side, and then starting a "new" company. Well, who cared? "Just let us build great stuff!" I thought.

However, the owner also believed in the "anti-aircraft process" for new-product development: Wait for ideas to fly over, then shoot them down. It got to be a joke. Hold up a piece of paper: "Is this what you want?" "No." Crumple, toss, scribble and hold up a new one. "Is this it?" "No." Morale floundered. A Windows product never happened.

The president's "new marketing" idea was to sell our complex CAD software by cold-call telemarketing. After a few thousand people said "Huh?" the owner canned him. He instituted "management by fax." We soon came to dread that little warbling sound.

The R&D director bailed. The right-hand guy took his place. "The office manager does the accounting," he said. "And she is as straight as they come. Since she's still here, it can't be too crooked." She quit. We came to work one day, and there was a grim-looking padlock on the door, next to a grim police officer who was next to an extremely grim landlord. Apparently someone had told him about the rename-the-company dodge.

We got (surprise!) a new company name. The new R&D director left. Our new offices had space for twenty, but the exodus accelerated, and soon there were just four of us rattling around. I stayed long enough to help hire some replacements, then left. It was clear that great stuff was never going to happen there again.

—Rick Wayne
Software Engineer

University of Wisconsin at Madison

Attack of the Niche Product

Here's a horror story about great stuff not making it—and it's happened in so many variations that I am certain everyone will have an immediate, yet different guess about YYY.

A small group of software developers, all with productivity levels way above average, often discussed the inadequacy of the various development tools they used daily. So, naturally, they went ahead and created something better. In flight, they also created a small start-up company to finish and commercialize these new tools. They had created something quite marvelous and even delivered it roughly on time.

Years later, the tool YYY is still around and still undergoing periodic revisions. However, its market share is too small to be computable and it hardly returned the investment made. What went wrong? Marketing had been weak. Some hype technologies had been included only at a slow pace. However, none of this was the problem. In hindsight, the biggest problem was that the star developers had created a sharp tool for only a small target market: themselves.

—Clemens Szyperski
Beyond Objects Columnist
Redmond, Wa.

I Don't Know, Do You?

Some time ago, I took on an engagement as a program manager, the person who pulls together the engineering, documentation, testing, sales, marketing, training and deliverables for a project release. I asked the development project manager what the state of the development was. He said he didn't know, so I asked to see his project plan or the project schedule. He didn't have one. I asked how he knew what was happening, and he told me that he had a one-on-one meeting with everyone every day to find out what was going on. I asked how long the meetings lasted, and he said, "At least one hour." I asked how much the project had slipped so far, and he asked, "Do we have an end date in mind?"

He wasn't a competent project manager. I first stopped his daily meetings with the developers, and we improved the engineers' productivity dramatically (from nothing to something).

Next, I visited the test manager. He was primarily concerned with duplicating the builds the developers created, so his group hadn't done much testing yet. We were supposed to ship a beta version in six weeks, so I asked him to take the risk that we might not be able

to duplicate the builds, but we needed to test the product, to see if it worked at all. I checked back with him in a couple of days, and he was still concerned about the build duplication. I asked him to show me the test plans. There were none. I asked him to show me the test cases, test procedures or test scripts. There were none.

I asked him if he could write a plan for how we were going to test the product, and to list the major areas to test. He looked at me sheepishly, and said, "I don't know how the product works."

I was stunned. I'd never been on a project before where the project manager didn't know how to manage projects and the test manager didn't know how to investigate the product to determine what to test.

We didn't make the original beta release or ship dates, but we did eventually deploy a product the customers liked and paid for. And, no, the project manager and test manager were not part of the final team who produced it.

—Johanna Rothman
Editorial Advisory Board Member
Arlington, Mass.

Before There Were Doctors

How is this for a horror story? In the early 1980s, I reconstructed a CP/M diskette directory by hand because the only copy of a major project's source files were on it and it was corrupted. I mean really by hand, because Norton Disk Doctor didn't exist in those days.

I had to use the low-level debugger (DDT) and map specific memory addresses to the values for track and sector, then code a call into the system to read. I dumped the memory to a printer, figured out which sectors needed to be linked together, used the skew table to determine their group numbers and then modified memory (using debugger commands) to create the directory entries. Next, I modified the specific system memory locations to point to track and sector for the directory and coded a call into the system to write my data to disk. Finally, I exited the debugger, did a DIR command and my files miraculously reappeared!

—Andy Barnhart
Contributing Editor
Raleigh, N.C.

Dividing by Zero

We had a project to develop a high performance algorithm. There was a prototype algorithm and a simulation, but no good analysis of whether the algorithm would work in real life. We spent a lot of time building a real system to test it, only to find that the algorithm performed poorly. We tried to fix it but ran out of research funds. We then spent the next six months using the funds of another project to rescue it; however, at the end we had conclusive results showing that the repaired version still didn't perform well. Moreover, by then a commercial product using specialized hardware did the job. The other project eventually was also abandoned without completion.

—Anonymous

The Project to End All Projects

The organization had 1,200 IT people and an upper management team enthralled with the idea of running information technology like a business. "We're going to be pros," they said.

"After we implement this project, things will be different. Do you want to get funding for a project? Prove it's worthy. Want to know the status of every IT project in the pipeline and its return on investment? Show me the money. Need to decide which projects to fund or kill? Let's check on their financial valuation."

Upper management's solution was to launch a meta-project meant to integrate the 30-something processes currently involved, add some new ones and, of course, build a new software tool to tie it all together. ("Surely, the software tool can save us!" is an oft-heard sentiment in stories like this one.)

They launched a project to manage the management of projects, creating a bureaucratic project to eliminate a bureaucratic process. And along the way, they demonstrated some classic anti-project management patterns, among them:

• Start big, but say you're starting small.

• Don't model the behavior you expect in the future by going through the effort to produce return on investment for the project.

• Have two senior managers, not one, as sponsor so that if they disagree, decision-making can be dragged out twice as long.

• Don't communicate the purpose clearly to the organization.

• Require every IT project leader to follow a bunch of confusing procedures, but don't

involve them in defining the procedures.

• Design and create a new software tool using contractors who get to learn the latest object-oriented, Web-enabled technology.

• Don’t use your own in-house staff for the analysis and development of the tool.

• Expect the tool to change behavior.

• Don’t involve the actual end users—in this case, project leaders—in prototyping or testing the system.

• Expect all the high-level managers to buy into the tool and tell them to tell everyone else that its use is mandatory.

After a little over a year and about $200,000, the effort slowly withered away.

—Ellen Gottesdiener
Editorial Advisory Board Member
Carmel, Ind.

Moribund Management

Forget the single-project horror stories. Sooner or later those projects will come to an end. Let me tell a more insidious tale, one that continues year after year after year: "The Attack of the Pointy-Haired Managers." Here are only a few of my close calls with these horrible monsters:

• A procurement manager at a large government contractor once asked my development team to follow his two-month evaluation and procurement process to get the best deal possible on software that we had already downloaded for free from the Internet. Luckily, we convinced him that his staff was better suited to perform the evaluation, which they did, enabling us to continue using the software we had already obtained.

• A program manager at a financial organization insisted that the existing workstations owned by his organization were sufficient for our new development environment. After reading the side of the box, I quickly realized that the workstations didn’t meet the minimum configuration, let alone the recommended one. It wasn’t until a consultant from the vendor came in to train the staff that the manager finally admitted that we needed to upgrade.

• A network support manager at a telecommunications firm once insisted that it would take six weeks to provide me with a workstation, completely ignoring the fact that I was only there for an eight-week period. At least the company believed that it was saving

money by outsourcing workstation support.

• A large group of logical data modelers, albeit not managers, working at a large contractor to the U.S. government, produced artifacts that were only used by another large group of logical data modelers working for the government. Had they been process modelers, they would have quickly discovered that nothing they produced was used by anyone outside of the two groups.

• The head of a system department once allowed a salesman from his primary vendor to become the de facto system architect on the most important project within his company. Anything the salesman wanted to sell to the client became part of the project, including some products that worked and others that didn't. To make matters worse, the team couldn't find any combination of the products that could work together within their technical environment. This project failed miserably.

Can these wretched managers and their ilk be stopped? We've tried silver bullets to no avail, but perhaps with education, experience and a lot of hard work we can rid ourselves of this terrible plague. Until then, try cloves of garlic to ward them off.

—Scott Ambler
Contributing Editor
Newmarket, Ontario

A Stitch in Time

A few years ago, when I was working as a technical and project leader, a manager tried to give me some advice. She explained that she had observed that the testing and debugging phases of my projects took up significantly less time than those of other projects, while my requirements analysis and design phases were much longer in comparison to projects led by others. Her argument was that, although my projects were always completed on time, I should try to cut the requirements analysis and design time by half, because this would result in completion long before the deadline—a great recipe! She even promised my team a big bonus if we implemented her suggestion. It was beyond her capacity to understand that the significant reduction of time spent in the testing and debugging phases might actually be a result of our good requirements analysis and design.

—Gilda Pour
Editorial Advisory Board Member
Software Engineering Professor,
San Jose State University
San Jose, Calif.

Data Demons

Early in my consulting career, I was heading up the design and C++ implementation group on a transportation system. A third-party database group was managing the project, and the client had assembled five data modelers to do the object analysis on the system.

I was worried. I arranged to review their object models a week before their formal review. Their "war room" was papered in hand-drawn class diagrams on large sheets of easel paper. My eyes immediately fell on the Customer class: it had an attribute named Customer_type. I almost went ballistic. I asked (knowing the answer), "How many types of Customer do we have?"

"Twelve," answered the team leader.

"Do they all have the same business rules and behavior?"

"Oh, no," he continued, "they're all quite different. That's why we use the type attribute."

He was bewildered when I looked him in the eye and said, "You don't have a clue to what you're doing!" He didn't understand that he was forcing me into a maintenance nightmare, drowning in switch statements and duplicated business rules. Every one of the remaining models was also a thinly disguised data model. I felt sick.

The formal review never took place. The client fired the integration group (their second in five months), hired a third group (three's the charm, right?) and switched to Smalltalk! I had one week left on my contract—I did not renew.

A year later, they delivered a prototype that would not scale, then canceled the whole project, fired everyone (except the database group) and switched to Clipper! It was only a miserly $10 million dollars down the drain. Thank goodness it wasn't "real" money like the amount that the government spends.

—Gary K. Evans
Contributing Editor
Architect, Evanetics Inc.
Columbia, S.C.

Bigfoot Sighting

My first job at a large communications company involved a massive project targeted at call center agents—the people on the other end of technical support lines. At first, it seemed like heaven. No code could be written without first producing detailed specifications, test plans and design documents. The project involved a dozen developers, several deliverables and was forecast to take a year or so. And the product was replete

with features for both the agents and their supervisors. At about the time everything came together, the company decided that the resulting product's memory footprint was too large. The product was scrapped.

This story has a happy ending, however: A complete redesign resulted in a smaller application targeted only for the agent, which was built in one month by three developers. The product was a success in the marketplace.

—Dana Cline
Contributing Editor
Thornton, Colo.

The Horror, the Horror

During a three-year period in the early- to mid-1990s, I worked in the insurance industry for a regional affiliate of a national provider of automotive, travel, insurance and financial services. The first year was a busy one. I worked with a small group of analysts and an enthusiastic and hands-on business unit manager in a successful effort to replace the aging software and hardware that controlled the company's remittance check processing.

Based on the positive outcome of that project, I was promoted to an analyst position and was responsible for providing technical expertise to three different departments within the company. One of the perks of the new job was an office with a commanding view of San Francisco's ornate City Hall and the nearby performing arts centers. Directly below my second-story window, in a vacant lot that adjoined the Arts Commission gallery, a group of local artisans and Buddhist monks were creating, with painstaking care, a large, elaborate and colorful sand painting known as a mandala. A mandala is destroyed upon completion to emphasize the impermanence of all earthly striving, which, as I later found out, was a fitting metaphor for a particular project the company's bumbling IS department had been working on for the past seven years.

The XXX Release 2.0 project was an attempt to modernize an aging batch COBOL application running on an IBM System/36 minicomputer. The original application tracked incoming customer calls, routed them to different service providers and generated various reports. If—and this was a big if—everything worked correctly, the customer's information was logged and service records were updated. The system required a roomful of hard-working call recorders and service dispatchers to manually fill in oodles of customer information. A decision had been made to modernize the batch application and make it more interactive by migrating it to a client/server OS/2 PC network, but the majority of IS managers only understood batch COBOL applications and were constantly befuddled by even the most rudimentary software engineering principles related to interactive applications. (I had quickly earned their enmity by rapidly developing—with the assistance of another developer—a series of interactive applications using tools like

Visual Basic and Visual C++.) After the company introduced a no-smoking rule in its buildings, I noticed, too, that three or four of the IS managers would gather regularly outside the main entrance, chain smoking and commiserating.

I was angry about the inefficiency and waste of the multi-year, multimillion dollar XXX 2.0 project that was siphoning money away from, in my mind, other, more useful projects. My anger was somewhat mollified one winter morning when I greeted a particularly troublesome IS manager puffing away in the doorway with a bandaged hand. His wounded hand had been the result of a home workshop accident, and I realized then and there that (under the skin, so to speak) he was a decent enough person who should be treated with respect (no matter how boneheaded his office politics were).

I left the insurance company shortly afterwards to take a position with a large telecomm company, where I ran into projects similar to XXX 2.0, which left their own bloody footprints in the snow. I heard later that the XXX 2.0 project dragged along for a few more years, before dying a lingering and costly death. Shortly before I left the insurance company, the sand mandala outside my window, with its vivid colors and intricate symmetries, was also swept away.

—Roger Smith
Technical Editor