

AIR JAM: The U.S. Federal Aviation Administration spent \$2.6 billion trying to upgrade its air-traffic-control system, only to cancel the project in 1994. Gridlocked skies are still with us today.



Why Software

FAILS

We waste billions of dollars
each year on entirely
preventable mistakes

By Robert N. Charette

Have you heard the one about the disappearing warehouse? One day, it vanished—not from physical view, but from the watchful eyes of a well-known retailer's automated distribution system. A software glitch had somehow erased the warehouse's existence, so that goods destined for the warehouse were rerouted elsewhere, while goods at the warehouse languished. Because the company was in financial trouble and had been shuttering other warehouses to save money, the employees at the "missing" warehouse kept quiet. For three years, nothing arrived or left. Employees were still getting their paychecks, however, because a different computer system handled the payroll. When the software glitch finally came to light, the merchandise in the warehouse was sold off, and upper management told employees to say nothing about the episode.



MARKET CRASH: After its new automated supply-chain management system failed last October, leaving merchandise stuck in company warehouses, British food retailer Sainsbury's had to hire 3000 additional clerks to stock its shelves.



This story has been floating around the information technology industry for 20-some years. It's probably apocryphal, but for those of us in the business, it's entirely plausible. Why? Because episodes like this happen all the time. Last October, for instance, the giant British food retailer J Sainsbury PLC had to write off its US \$526 million investment in an automated supply-chain management system. It seems that merchandise was stuck in the company's depots and warehouses and was not getting through to many of its stores. Sainsbury was forced to hire about 3000 additional clerks to stock its shelves manually [see photo, "Market Crash"].

This is only one of the latest in a long, dismal history of IT projects gone awry [see table, "Software Hall of Shame" for other notable fiascoes]. Most IT experts agree that such failures occur far more often than they should. What's more, the failures are universally unprejudiced: they happen in every country; to large companies and small; in commercial, nonprofit, and governmental organizations; and without regard to status or reputation. The business and societal costs of these failures—in terms of wasted taxpayer and shareholder dollars as

well as investments that can't be made—are now well into the billions of dollars a year.

The problem only gets worse as IT grows ubiquitous. This year, organizations and governments will spend an estimated \$1 trillion on IT hardware, software, and services worldwide. Of the IT projects that are initiated, from 5 to 15 percent will be abandoned before or shortly after delivery as hopelessly inadequate. Many others will arrive late and over budget or require massive reworking. Few IT projects, in other words, truly succeed.

The biggest tragedy is that software failure is for the most part predictable and avoidable. Unfortunately, most organizations don't see preventing failure as an urgent matter, even though that view risks harming the organization and maybe even destroying it. Understanding why this attitude persists is not just an academic exercise; it has tremendous implications for business and society.

SOFTWARE IS EVERYWHERE. It's what lets us get cash from an ATM, make a phone call, and drive our cars. A typical cellphone now contains 2 million lines of software code; by 2010 it will likely have 10 times as

Software Hall of Shame

YEAR	COMPANY	OUTCOME (COSTS IN US \$)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to \$33.3 million* loss.
2004-05	UK Inland Revenue	Software errors contribute to \$3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after \$54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately \$400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing \$527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to \$160 million loss.
2003-04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of \$100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after \$170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after \$33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to \$445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to \$100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after \$130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing \$25 million.
1999	United Way	Administrative processing system canceled after \$12 million is spent.
1999	State of Mississippi	Tax system canceled after \$11.2 million is spent; state receives \$185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to \$151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of \$50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after \$4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after \$40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to \$3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause \$350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	\$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after \$25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after \$2.6 billion is spent.
1994	State of California	DMV system canceled after \$44 million is spent.
1994	Chemical Bank	Software error causes a total of \$15 million to be deducted from 100 000 customer accounts.
1993	London Stock Exchange [UK]	Taurus stock settlement system canceled after \$600 million** is spent.
1993	Allstate Insurance Co.	Office automation system abandoned after deployment, costing \$130 million.
1993	London Ambulance Service [UK]	Dispatch system canceled in 1990 at \$11.25 million**; second attempt abandoned after deployment, costing \$15 million.**
1993	Greyhound Lines Inc.	Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of \$61 million.
1992	Budget Rent-A-Car, Hilton Hotels, Marriott International, and AMR [American Airlines]	Travel reservation system canceled after \$165 million is spent.

Sources: *Business Week*, *CEO Magazine*, *Computerworld*, *InfoWeek*, *Fortune*, *The New York Times*, *Time*, and *The Wall Street Journal*

* Converted to U.S. dollars using current exchange rates as of press time.

† Converted to U.S. dollars using exchange rates for the year cited, according to the International Trade Administration, U.S. Department of Commerce.

** Converted to U.S. dollars using exchange rates for the year cited, according to the *Statistical Abstract of the United States*, 1996.

many. General Motors Corp. estimates that by then its cars will each have 100 million lines of code.

The average company spends about 4 to 5 percent of revenue on information technology, with those that are highly IT dependent—such as financial and telecommunications companies—spending more than 10 percent on it. In other words, IT is now one of the largest corporate expenses outside employee costs. Much of that money goes into hardware and software upgrades, software license fees, and so forth, but a big chunk is for new software projects meant to create a better future for the organization and its customers.

Governments, too, are big consumers of software. In 2003, the United Kingdom had more than 100 major government IT projects under way that totaled \$20.3 billion. In 2004, the U.S. government cataloged 1200 civilian IT projects costing more than \$60 billion, plus another \$16 billion for military software.

Any one of these projects can cost over \$1 billion. To take two current examples, the computer modernization effort at the U.S. Department of Veterans Affairs is projected to run \$3.5 billion, while automating the health records of the UK's National Health Service is likely to cost more than \$14.3 billion for development and another \$50.8 billion for deployment.

Such megasoftware projects, once rare, are now much more common, as smaller IT operations are joined into "systems of systems." Air traffic control is a prime example, because it relies on connections among dozens of networks that provide communications, weather, navigation, and other data. But the trick of integration has stymied many an IT developer, to the point where academic researchers increasingly believe that computer science itself may need to be rethought in light of these massively complex systems.

WHEN A SOFTWARE PROJECT FAILS, it jeopardizes an organization's prospects. If the failure is large enough, it can steal the company's entire future. In one stellar meltdown, a poorly implemented resource planning system led FoxMeyer Drug Co., a \$5 billion wholesale drug distribution company in Carrollton, Texas, to plummet into bankruptcy in 1996.

IT failure in government can imperil national security, as the FBI's Virtual Case File debacle has shown. The \$170 million VCF system, a searchable database intended to allow agents to "connect the dots" and follow up on disparate pieces of intelligence, instead ended five months ago without any system's being deployed [see "Who Killed the Virtual Case File?" in this issue].

IT failures can also stunt economic growth and quality of life. Back in 1981, the U.S. Federal Aviation Administration began looking into upgrading its antiquated air-traffic-control system, but the effort to build a replacement soon became riddled with problems [see photo, "Air Jam"]. By 1994, when the agency finally gave up on the project, the predicted cost had tripled, more than \$2.6 billion had been spent, and the expected delivery date had slipped by several years. Every airplane passenger who is delayed because of gridlocked skyways still feels this cancellation; the cumulative economic impact of all those delays on just the U.S. airlines (never mind the passengers) approaches \$50 billion.

Worldwide, it's hard to say how many software projects fail or how much money is wasted as a result. If you define failure as

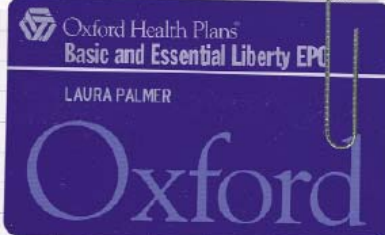
Case Study #1

COMPANY:

Oxford Health Plans

FAILURE:

New billing system cannot keep up with expanding business, resulting in uncollected payments of \$400 million from patients and \$650 million owed to caregivers.



LOSS:

October 1997 announcement of quarterly loss triggers stock price to drop from \$68 to \$26 in one day, wiping out \$3.4 billion in corporate value. Company later pays investors \$225 million to settle lawsuits.

the total abandonment of a project before or shortly after it is delivered, and if you accept a conservative failure rate of 5 percent, then billions of dollars are wasted each year on bad software.

For example, in 2004, the U.S. government spent \$60 billion on software (not counting the embedded software in weapons systems); a 5 percent failure rate means \$3 billion was probably wasted. However, after several decades as an IT consultant, I am convinced that the failure rate is 15 to 20 percent for projects that have budgets of \$10 million or more. Looking at the total investment in new software projects—both government and corporate—over the last five years, I estimate that project failures have likely cost the U.S. economy at least \$25 billion and maybe as much as \$75 billion.

Of course, that \$75 billion doesn't reflect projects that exceed their budgets—which most projects do. Nor does it reflect projects delivered late—which the majority are. It also fails to account for the opportunity costs of having to start over once a project is abandoned or the costs of bug-ridden systems that have to be repeatedly reworked.

Then, too, there's the cost of litigation from irate customers suing suppliers for poorly implemented systems. When you add up all these extra costs, the yearly tab for failed and troubled software conservatively runs somewhere from \$60 billion to \$70 billion in the United States alone. For that money, you could launch the space shuttle 100 times, build and deploy the entire 24-satellite Global Positioning System, and develop the Boeing 777 from scratch—and still have a few billion left over.

WHY DO SOFTWARE PROJECTS FAIL SO OFTEN?

Among the most common factors:

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures