Friends don’t let friends release bad software. Most teams know too well the sting of learning from their own mistakes, but it’s much less painful to learn lessons from the experience of others. With that in mind, this article reviews a number of very public catastrophic software failures in recent years. No, we’re not trying to depress you. Rather, these are good reminders that can incentivize all software professionals to remember the importance of testing.

The Cap Gemini World Quality Report makes it clear that the last few years have been very challenging for software companies as they grapple with new technologies. Also, there have been too many major software failures, each of which has a deleterious impact on public reputation, net profit, and customer satisfaction. The aftermath of each major failure disperses far and wide over the Internet and the evening news, and the black cloud can hover over a business for years.

Before outlining an approach to avoiding such failures, let’s review a sampling of the major software failures that became bad news in recent years.

**GPS satellite failure affects many critical systems**

The catastrophic failure of a 25-year-old GPS satellite this past in January 2017 activated a software bug which that only manifested itself for only a mere 13 microseconds. Though the fault was momentary, the result was an enormous impact to global positioning systems (GPS), the US Air Force, and telecom networks. For every nanosecond of discrepancy in GPS timing, a distance reading may be in error by as much as a foot. So, a
The Equifax mega-breach

This is one of the biggest exposures, mostly due to poor quality assurance culture and practices. The personal information that was stolen in the Equifax database breach potentially compromised the identities of 145 million people. That is more than a third of the US population. Many of these people had given no consented for Equifax to retain their personal information in their databases. Worse yet, Equifax did not apply any patches to its website for many months after major flaws in its software were disclosed publicly.

Blue Cross/Blue Shield system failure

The Blue Cross Blue Shield Association of North Carolina endured a major new-system failure in January 2016, the result of which was that nearly 25,000 participants were enrolled in the wrong health insurance plan. BC/BS of North Carolina was eventually made to pay a fine of $3.6 million. The deluge of complaints made it clear that there was a system problem. The problems became much worse when it became news that an internal source made it know that BCBS management had been aware of the problem yet continued with the implementation of the system. Clearly, that was an abysmally poor decision. Software bugs and shoddy features can be very challenging for any organization. Failure to remediate and fix all serious problems is a sure-fire way to lose customers, incur penalties, and damage the company image.

GameStop fails on the launchpad

GameStop has been struggling to stay viable as digital games continue to increase in popularity. The company launched a rental service known as PowerPass in October 2017. For a fee of $60, a member could check out a used game from a local store one at a time over the course of six months. At the end of that period, the member could keep one of those used game titles. One month after it launched, GameStop brought an end to PowerPass. Kotaku reported that company computers had not been set up to manage the PowerPass program. Surely this is a lesson on the value of good implementation and integration planning.
The touch controls on the Google Home Mini—one competitor of the Echo Dot—activates the Google Assistant software. A major problem with the product back in 2017 was that the touch controls were initially configured such that the software was receiving signals for a continuous depression of those buttons. The result was that Google Assistant was continuously recording every word that was said. This was a major fail because even though Google made an attempt to fix this problem by publishing an update to the firmware, it didn't solve the problem at all. Google wiped the collective egg off its face and disabled this feature entirely. This feature launch gone wrong is even worse when we realize that it was entirely preventable.

**Software malfunction drops thousands of 911 callers**

In 2014, a major 3rd-party nationwide emergency call center hub that directs and assigns 911 calls failed due to a major software malfunction. On April 9, thousands of calls were instantaneously dropped. Apparently, the tracking software contained a fixed-value counter maximum of 40 million calls. When this actual limit was reached on April 9, all additional calls were simply dropped—failing to provide service to 11 million people in more than 7 states.

### Preventing catastrophe at your company

It's might be easy to assume that failures such as these are merely a cost of shipping software in an interconnected world. That can't be true since many companies continue to operate responsibly and steadily provide high-quality products and services. Indeed, a primary reason that many software companies continue to embrace test automation is that it equips them to readily find any defects before the software build enters production.

Now that CI/CD is becoming common practice, there is a much higher frequency of upgrades, updates, and patches occurring than ever before. But, the high frequency of product changes should never put your company at risk. It's quite feasible for teams to deploy a continuous stream of changes and simultaneously ensure that each and every build functions properly.

Comprehensive testing combines conventional testing excellence with effective automation technology to achieve bug feature validation and broader test coverage—while updates and changes continue to aggregate in the pre-production build. Technology failures can spawn interminable public relations problems—as we have seen in this article. Though it can sometimes be difficult to precisely quantify cost savings that directly result from preventing defects, there is enormous value in making the effort to build bug-free applications and protecting product/brand reputation.

Ensuring that all features will be thoroughly tested while new updates continue through the product pipeline simply cannot be done effectively with manual testing. This is why automation frameworks such as Functionize are critically important for testing non-trivial applications. With the Functionize AI toolset, teams can reliably and extensively automate testing—at virtually any frequency or scale.

One way to think about improvements to your test automation toolset is to view it as insurance. Not only will your team build better quality software, many have found that implementing test automation is a hedge against customer disappointment, negative publicity, and employee frustration. This is quite important if you are an Agile DevOps development team. The rate of change in large, complex application development pipelines makes it impracticable to ensure high quality can be managed using conventional, non-automated approaches.

### Protect your brand, and delight your customers

As the demand for cutting-edge technology and maximum convenience intensifies, there is a pressing need for skillful software testers that will guard their brand and their users from the never-ending potential for software failures. At Functionize, our mission is to create maintain scale.
customers with best-in-class tools that will help ensure that the inevitable bugs in your software products will be found by your testers—and never be seen by your customers.
The Functionize platform is powered by our Adaptive Event Analysis™ technology which incorporates self-learning algorithms and machine learning in a cloud-based solution.

First name*

Last name*

Email*

Phone number*

Company name*

I'm interested in learning more about*
- AI Cloud Testing
- ALP test creation
- Autonomous testing
- Cross Browser Testing
- Functional Testing
- Load and Performance Testing
- Model-based testing
- Self-healing maintenance
- Service Virtualization
- Test cloud orchestration

Submit

© 2020 Functionize, Inc. All rights reserved.
Privacy Policy | Terms of Service | Support Policies

We use cookies to ensure that we give you the best experience on our website. If you continue to use this site we will assume that you are happy with it.

It's fine