

Disasters in Software (and How to Avoid Them)

Jason Bock

Personal Info

- <http://jasonbock.net>
- <https://twitter.com/jasonbock>
- <https://github.com/jasonbock>
- <https://youtube.com/jasonbock>
- <https://twitch.tv/jasonbock>
- jason.r.bock@outlook.com

Downloads

<https://github.com/JasonBock/Presentations/>

Overview

- Definitions
- Examples
- Recommendations
- Conclusions

Remember...

<https://github.com/JasonBock/Presentations/>

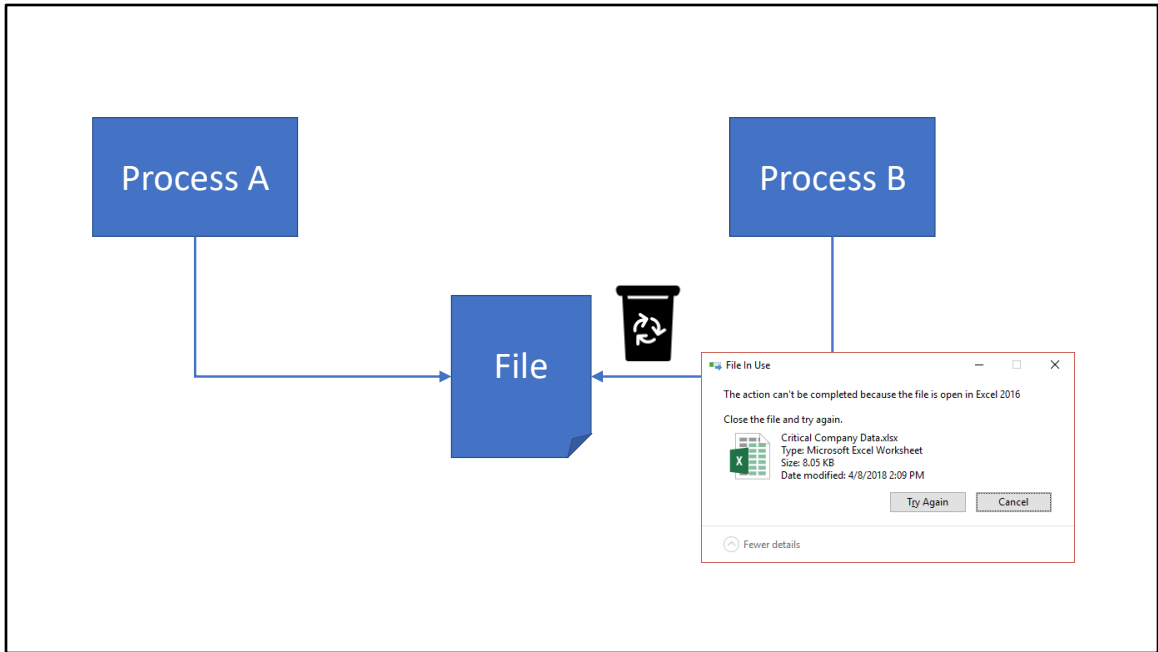
Definitions

Disasters in Software (and How to Avoid Them)



What is a “disaster”? It’s kind of like a “failure”, though typically I end up laughing at failures, and they’re usually not that devastating. Failure == “lack of success.”

https://unsplash.com/photos/52jRtc2S_VE



Typically, failures are teachable and recoverable. It's a failure to not open a file, but we can easily recover from it, and we can plan for it.

```
var source = /* reference to array */  
var destination = new int[0];  
  
foreach(var item in source)  
{  
    var holder = new int[destination.Length + 1];  
    destination.CopyTo(holder, 0);  
    holder[destination.Length] = item;  
    destination = holder;  
}
```



Failures in software can also be attributed to really bad code. This copies an array, but it's awful in terms of performance and memory consumption. This was in production code, and it was "tolerable" in that the developer never saw a big enough problem to change this.




So, what is our operating definition of a disaster?

<https://unsplash.com/photos/1UDjq8s8cy0>

disaster

 [dih-zas-ter, -zah-ster] [SHOW IPA](#)  

See synonyms for: [disaster](#) / [disasters](#) on Thesaurus.com

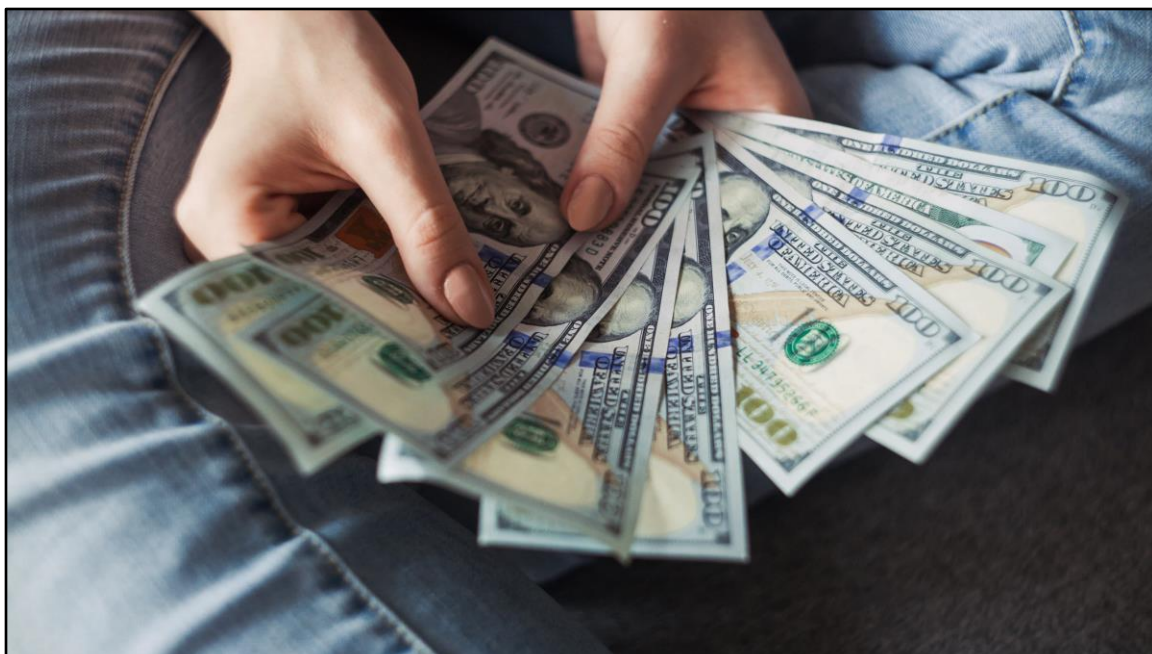
 Elementary Level

noun

- 1 a calamitous event, especially one occurring suddenly and causing great loss of life, damage, or hardship, as a flood, airplane crash, or business failure.
- 2 *Obsolete.* an unfavorable aspect of a star or planet.

This is what I'm using.

<https://www.dictionary.com/browse/disaster>



Also, the “loss” typically equates to money, usually LOTS of it.

<https://unsplash.com/photos/ICPhGxs7pww>

An unexpected incident in software that has catastrophic results

To bring it a little closer to our world...

Examples

Disasters in Software (and How to Avoid Them)

Let's go through some examples of software disasters.



Let's accept that we've all made mistakes. Giving examples, isn't to blame or ridicule others.

<https://www.pexels.com/photo/happy-women-hugging-4584462/>



Now, bad software doesn't mean it's a disaster, though arguable MS Bob was. Bob did what it was intended to, it was just very, VERY poorly received.

<https://fossbytes.com/microsoft-bob-infamous-desktop-enhancement-software/>

```
year = ORIGINYEAR; /* = 1980 */
```

```
while (days > 365)
{
    if (IsLeapYear(year))
    {
        if (days > 366)
        {
            days -= 366;
            year += 1;
        }
    }
    else
    {
        days -= 365;
        year += 1;
    }
}
```

?



"A bug in the internal clock driver related to the way the device handles a leap year affected Zune users," said the company in a statement. "That being the case, the issue should be resolved over the next 24 hours as the time change moves to January 1, 2009."

Zune Bug - <https://techcrunch.com/2008/12/31/zune-bug-explained-in-detail/> - For one day, this thing did not work. Why? It didn't handle leap years right, and got stuck in an endless loop

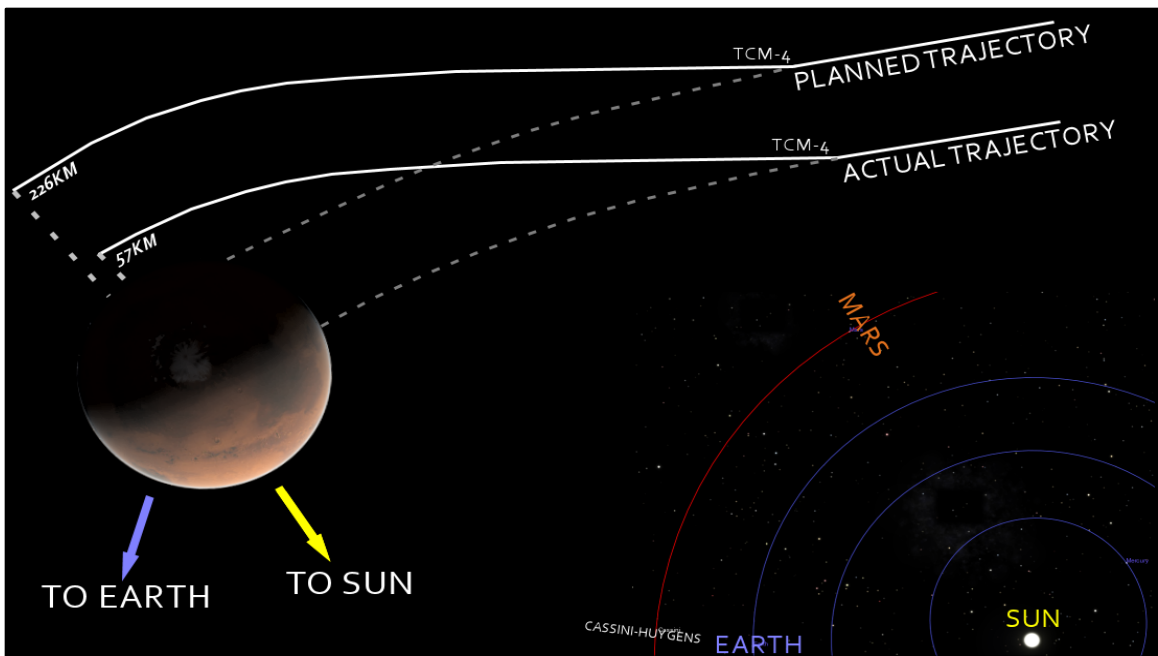
<http://www.hanselman.com/blog/BackToBasicsExploreTheEdgeCasesOrDateMathWillGetYou.aspx>

<https://www.wired.com/2008/12/zune-freeze-res/>



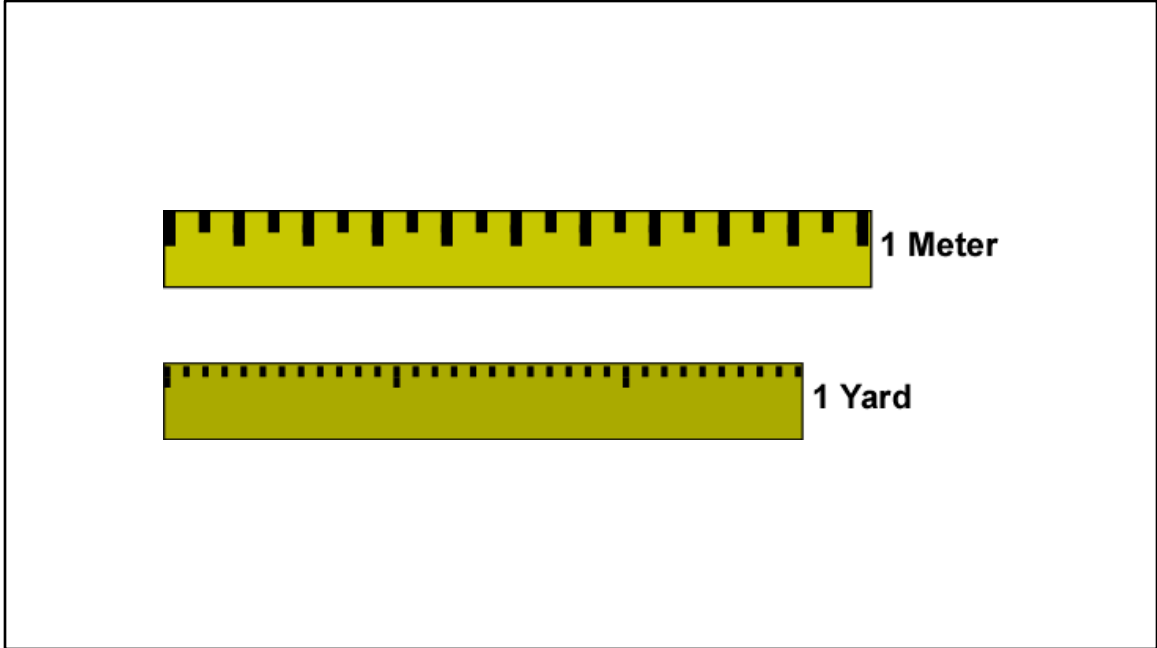
Mars Climate Orbiter – basically, it's about mismeasurements. To be specific, the units of measurement.

<https://news.cornell.edu/stories/1998/12/two-nasa-spacecraft-launches-one-mars-and-other-research-birth-stars-involve>



“Twenty-four hours prior to orbital insertion, calculations placed the orbiter at an altitude of 110 kilometers; 80 kilometers is the minimum altitude that Mars Climate Orbiter was thought to be capable of surviving during this maneuver. Post-failure calculations showed that the spacecraft was on a trajectory that would have taken the orbiter within 57 kilometers of the surface, where the spacecraft likely disintegrated because of atmospheric stresses.”

https://en.wikipedia.org/wiki/Mars_Climate_Orbiter#/media/File:Mars_Climate_Orbiter_-_mishap_diagram.png



Why?

“The primary cause of this discrepancy was that one piece of ground software supplied by Lockheed Martin produced results in a United States customary unit, contrary to its Software Interface Specification (SIS), while a second system, supplied by NASA, expected those results to be in SI units, in accordance with the SIS.”

And they knew this!

“The discrepancy between calculated and measured position, resulting in the discrepancy between desired and actual orbit insertion altitude, had been noticed earlier by at least two navigators, whose concerns were dismissed. A meeting of trajectory software engineers, trajectory software operators (navigators), propulsion engineers and managers, was convened to consider the possibility of executing Trajectory Correction Maneuver-5, which was in the schedule. **Attendees of the meeting recall an agreement to conduct TCM-5, but it was ultimately not done.**” – does that sound familiar? 😊

<http://www.physics.arizona.edu/~hoffman/ua200/mechanics/1a1035.gif>

"The cost of the mission
was \$327.6 million total
for the orbiter and
lander"



Denver International Airport Baggage Handling System

“What was to be the world’s largest automated airport baggage handling system, became a classic story in how technology projects can go wrong.”

<https://unsplash.com/photos/f71sZHAbU-A>



What happened?

“The embarrassing missteps along the way included an impromptu demonstration of the system to the media which illustrated how the system crushed bags, disgorged content and how two carts moving at high speed reacted when they crashed into each other [4]. When opening day finally arrived, the system was just a shadow of the original plan. Rather than automating all 3 concourses into one integrated system, the system was used in a single concourse, by a single airline and only for outbound flights [5]. All other baggage handling was performed using simple conveyor belts plus a manual tug and trolley system that was hurriedly built when it became clear that the automated system would never achieve its goal”

<https://unsplash.com/photos/kj4e59Sf7Q0>

“Expenditure to maintain the empty airport and interest charges on construction loans cost the city of Denver \$1.1M per day throughout the delay”

<http://calleam.com/WTPF/wp-content/uploads/articles/DIABaggage.pdf>



Knight Capitol Group

“On August 1, 2012, Knight Capital caused a major stock market disruption leading to a large trading loss for the company. The incident happened due to a technician forgetting to copy the new Retail Liquidity Program (RLP) code to one of the eight SMARS computer servers, which was Knight's automated routing system for equity orders. RLP code repurposed a flag that was formerly used to activate the old function known as 'Power Peg'. Power Peg was designed to move stock prices higher and lower in order to verify the behavior of trading algorithms in a controlled environment.[12] Therefore, orders sent with the repurposed flag to the eighth server triggered the defective Power Peg code still present on that server.[13] When released into production, Knight's trading activities caused a major disruption in the prices of 148 companies listed at the New York Stock Exchange, thus, for example, shares of Wizzard Software Corporation went from \$3.50 to \$14.76. For the 212 incoming parent orders that were processed by the defective Power Peg code, Knight Capital sent millions of child orders, resulting in 4 million executions in 154 stocks for more than 397 million shares in approximately 45 minutes.[13] Knight Capital took a pre-tax loss of \$440 million. This caused Knight Capital's stock price to collapse, sending shares lower by over 70% from before the announcement. The nature of the

Knight Capital's unusual trading activity was described as a "technology breakdown".^{[14][15]}

“The incident happened due to a technician forgetting to copy the new Retail Liquidity Program (RLP) code to one of the eight SMARS computer servers, which was Knight's automated routing system for equity orders”

https://en.wikipedia.org/wiki/Knight_Capital_Group#2012_stock_trading_disruption

“In 45-minutes Knight went from being the largest trader in US equities and a major market maker in the NYSE and NASDAQ to bankrupt”

https://en.wikipedia.org/wiki/Knight_Capital_Group#2012_stock_trading_disruption



Ariane 5 Rocket Explosion

<https://www.nasaspaceflight.com/2019/11/ariane-5-tiba-1-inmarsat-gx5/>



"All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space."

https://www.youtube.com/watch?v=PK_yguLapgA

https://en.wikipedia.org/wiki/Ariane_5#VA241_anomaly



"All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space."

https://www.youtube.com/watch?v=PK_yguLapgA

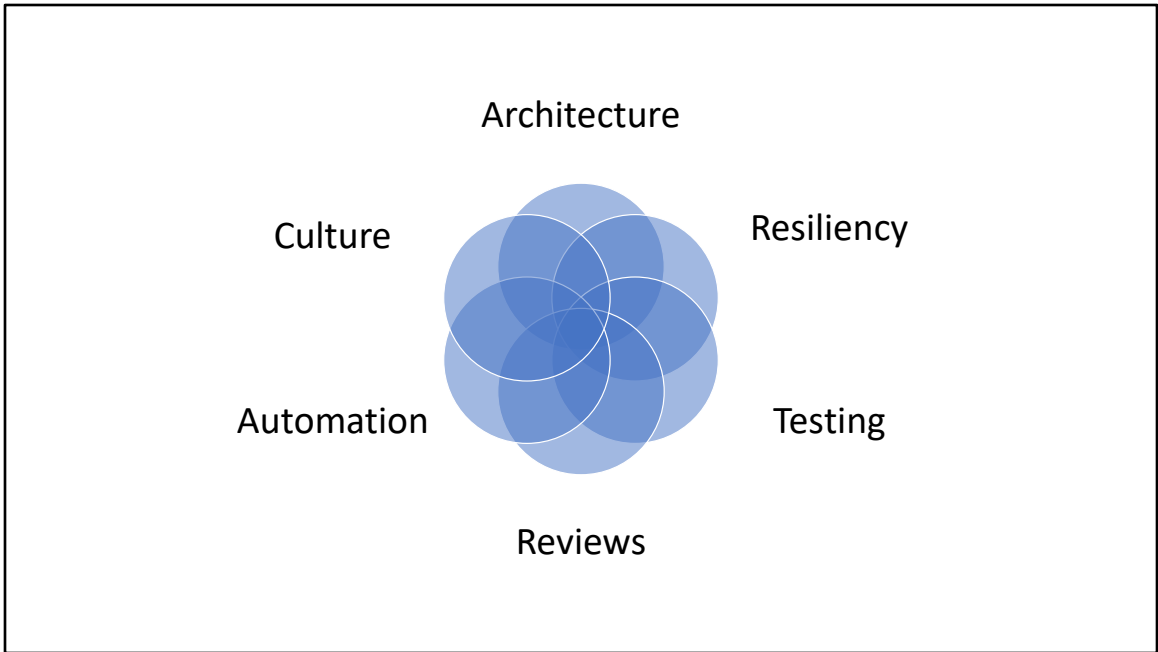
https://en.wikipedia.org/wiki/Ariane_5#VA241_anomaly

Cost per launch
\$165–220M

Recommendations

Disasters in Software (and How to Avoid Them)

Let's go through some recommendation of things you can do to mediate and reduce software disasters.





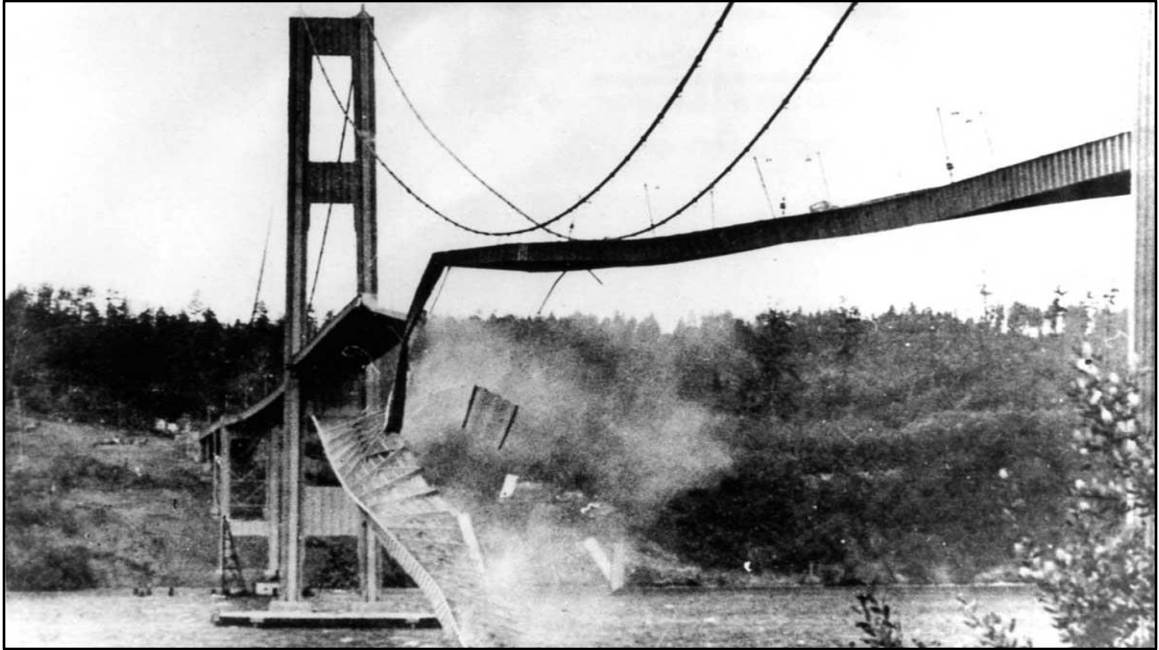
If you have a big ball of mud with duct tape and staples all over it and spaghetti code everywhere, you're going to fall off a cliff. Having a good architecture doesn't mean you throw names of well-known patterns everywhere and demand 100+ page documents that have fancy pictures but little substance. Having a strong understanding of the needs of the users is paramount. A well-architected application supports these needs and requirements. It's not about pushing your favorite technology or language, though using something that the devs know well is a bonus.

<https://www.pexels.com/photo/overhead-photo-of-an-architect-s-deisgn-5582599/>



We have the best of intentions, and we are smart, intelligent folks. But we're fallible...

<http://www.tacomannarrowsbridge.org/z1940-100.jpg>



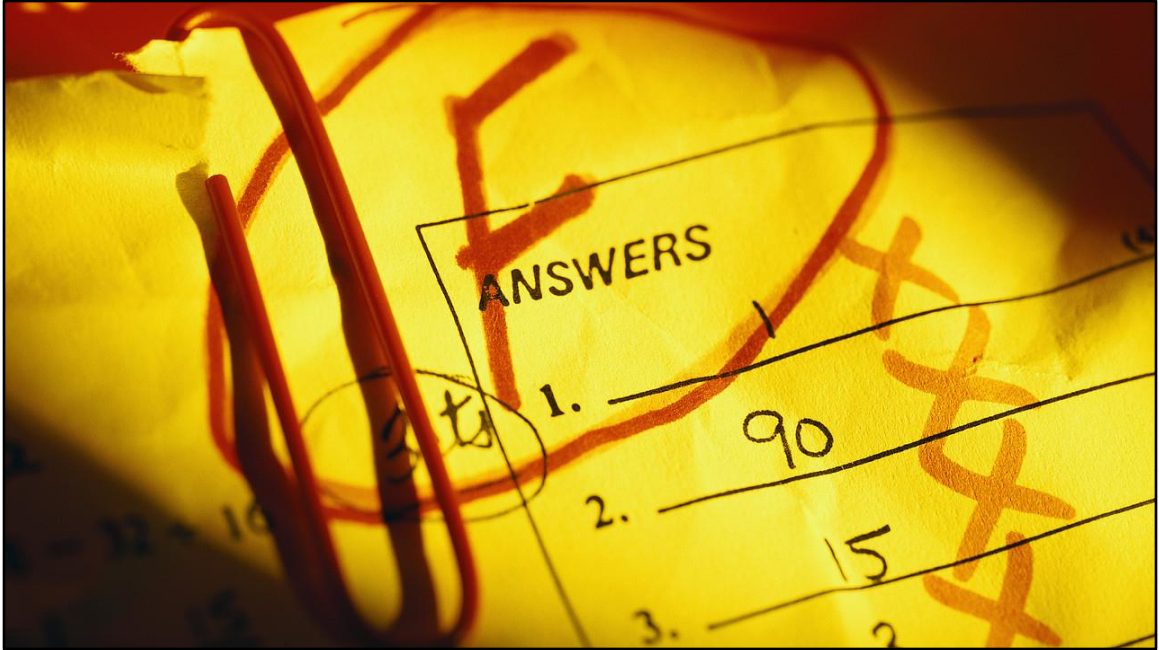
If there's a truism in code, it's that we'll write code that fails. And sometimes that happens spectacularly.

<https://exit133.com/uploads/Narrows-Bridge-Collapse.jpg>



What kind of plans do you have in place if things go wrong? e.g .you can't call a service, or a database is unreachable, or a disk fills up with space, or the network is extremely laggy, or....a whole assortment of problems. Dependencies are not in your control, and you have to accept that things will fail. Building to withstand failures reduces the chances for disaster.

<https://www.shutterstock.com/video/clip-3445493-stock-footage-sunset-behind-pearl-bridge-akashi-ohashi-spanning-the-seto-inland-sea-from-kobe-japan.html>



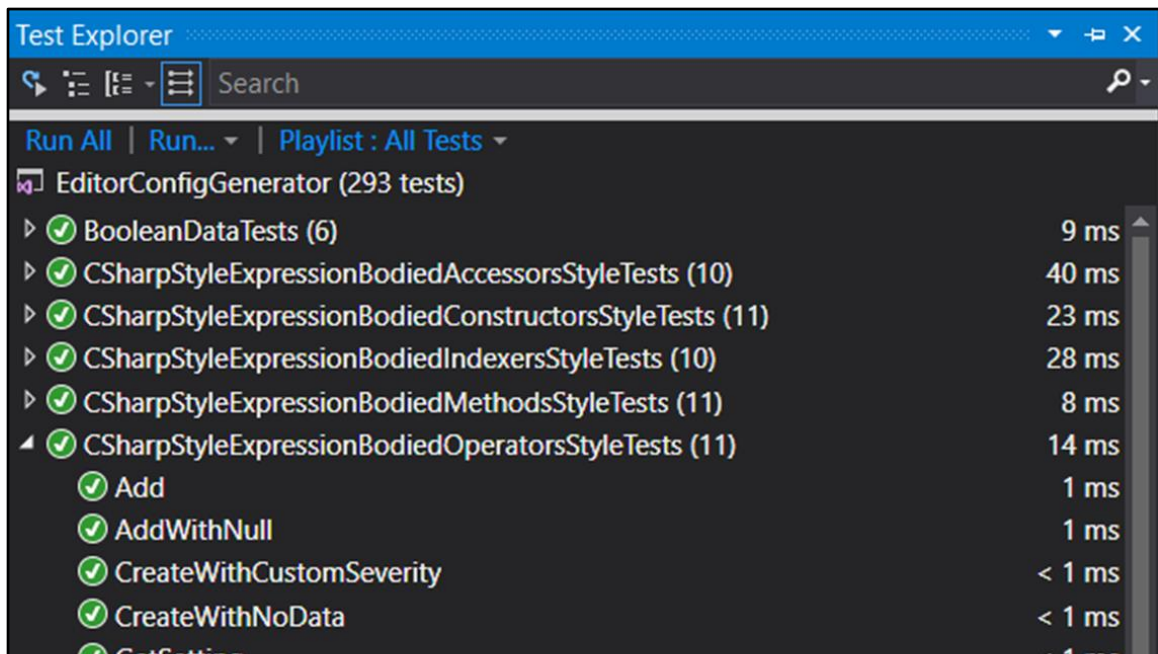
I cannot stress this enough. If you do not test your software, you have no hope. Zero. Nada. Zip. Ziltch. I don't care if you come up later and tell me how awesome your software is and you have no tests of any kind.

<https://social.eli.ubc.ca/files/2011/08/fail.jpg>

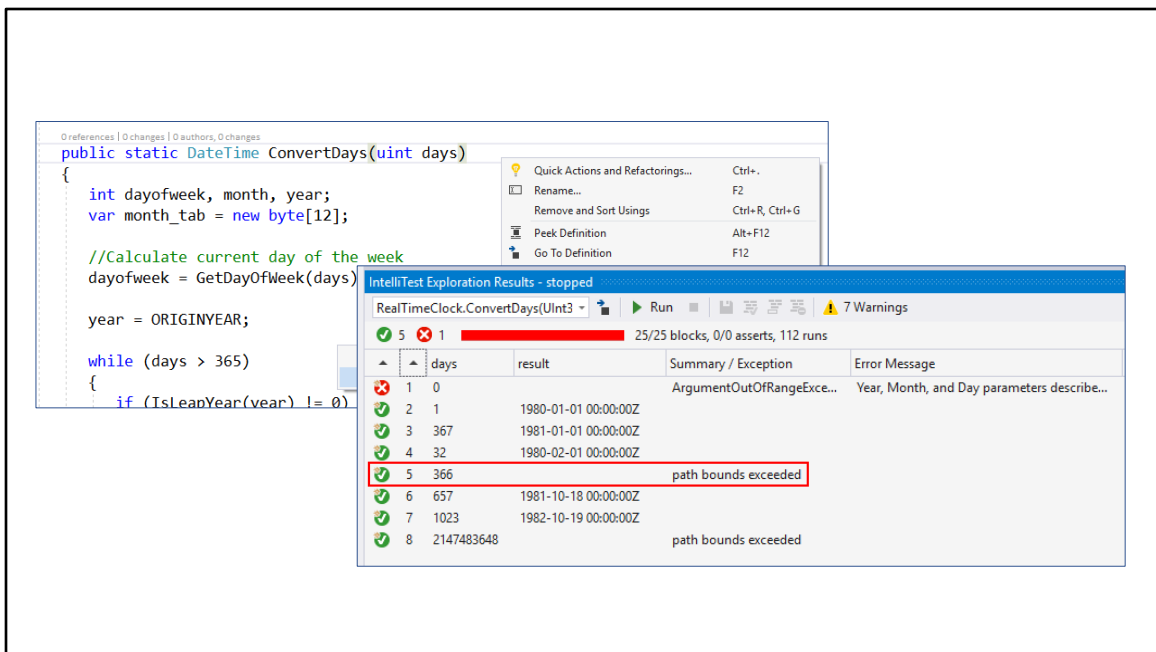


You skating on thin ice and you don't know where or when you'll fall in.

<https://unsplash.com/photos/mNIE4uaD3do>



Having tests gives you confidence that your code is working as expected, and that you can change it over time.



We can also use Intellitest to generate tests for us and exercise our code's boundary conditions (show what happens with the Zune bug). Note that this requires Enterprise SKU, so that's probably not doable for everyone, but it is an option.

<https://docs.microsoft.com/en-us/visualstudio/test/intellitest-manual>



Side note: doing analysis for formatting issues is a good thing. But don't argue over them for hours at a time, just pick something to be consistent.

<http://www.stellman-greene.com/blog/wp-content/uploads/2008/09/sally-code-review.png>



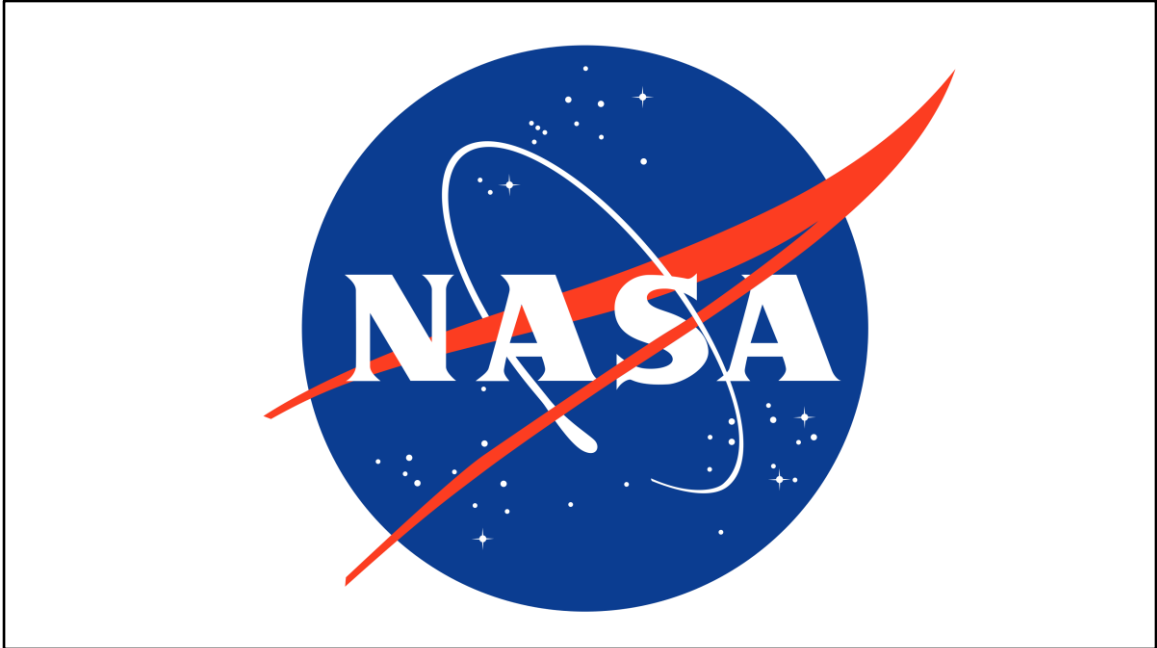
Having people do all the steps manually can lead to breakdowns and failures (see Amazon issue above). Having a strong, healthy DevOps CI/CD pipeline can uncover issues well before they ever make it into production.

<https://unsplash.com/photos/jHZ70nRk7Ns>



If people just don't give a ****, it doesn't matter how great your strategy is. "Culture eats strategy for breakfast". Well, you are what you eat, so give that toxic culture some good processed food to eat.

<https://www.pexels.com/photo/pancake-with-sliced-strawberry-376464/>



Let's talk about a very famous case of culture: NASA.

<https://www.nasa.gov/>

https://en.wikipedia.org/wiki/NASA?msclkid=828940ccaeb811ecb126ef0d092c99e3#/media/File:NASA_logo.svg



Specifically, Challenger. Unfortunately, people, when they hear of Challenger, they think of the accident.

<https://vistapointe.net/space-shuttle-challenger.html>



Specifically, Challenger. Unfortunately, people, when they hear of Challenger, they think of the accident.

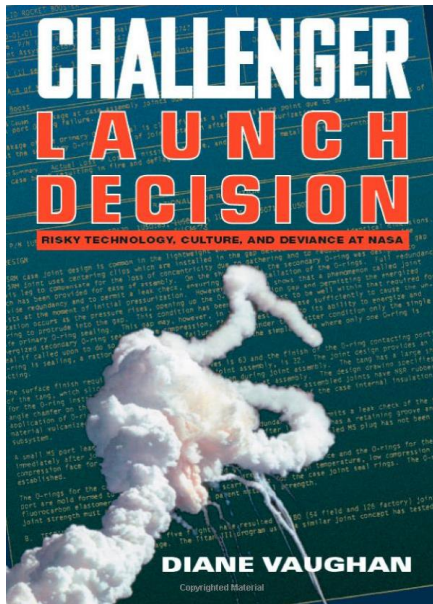
<https://www.ien.com/safety/news/20492193/engineer-who-predicted-challenger-disaster-dies>



But....how did this really happen? The mechanics are complex, but to summarize in one slide....

- 1) It was a cold launch day, and that made the O-rings brittle
- 2) At launch, there were black puffs of smoke that indicated the O-ring being vaporized. But surprisingly, a “seal” must have been made at this point.
- 3) As the shuttle rose in the sky, it experienced strong wind shears (though under tolerance). This must have jarred the seal enough, because a flame started showing up on that SRB.
- 4) This lead to rapidly increasing problems, and the eventual destruction of the shuttle.

<https://www.youtube.com/watch?v=6JISfB32sJo>



Production of Culture Normalization of Deviance

“NASA management became accustomed to ... phenomena when no serious consequences resulted from these earlier episodes”

One of the best books I've ever read was this one. It's dry, and arguably boring, but ridiculously thorough. The book stresses the point about “production of culture” and “normalization of deviance”.

<https://www.amazon.com/Challenger-Launch-Decision-Technology-Deviance/dp/0226851753/>
https://en.wikipedia.org/wiki/Space_Shuttle_Columbia_disaster



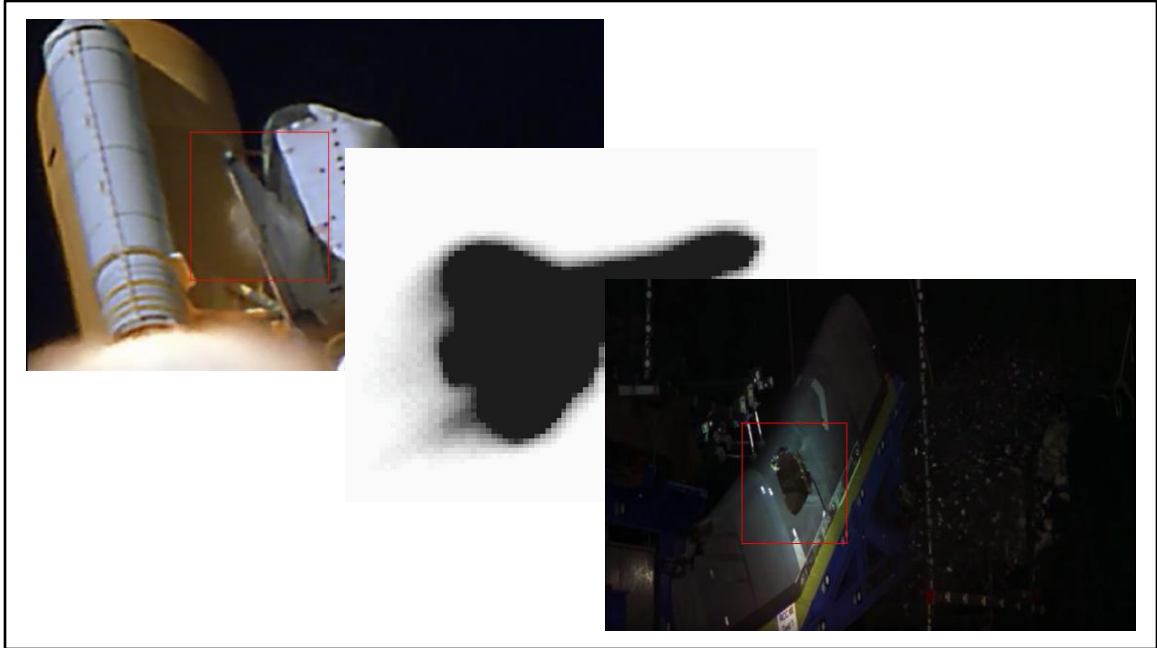
The sad thing is, even after changes within NASA, another shuttle, Columbia, was lost.

<https://nara.getarchive.net/media/a-left-side-view-of-the-space-shuttle-columbia-preparing-to-land-after-completing-9460e1>



The sad thing is, even after changes within NASA, another shuttle, Columbia, was lost.

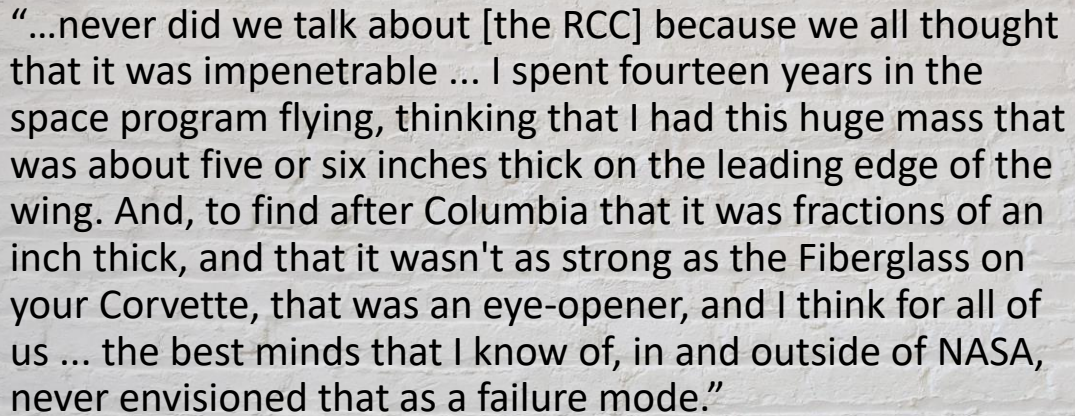
<https://www.pinterest.com/pin/iconic-images-and-major-moments-that-became-part-of-history--187814246936550484/>



Foam strikes, which were known on other missions, hit the left wing. During re-entry, hot gases entered the craft, eventually causing its destruction. Testing later showed that a piece of foam the size estimated from the original strike at the estimated speed could cause a hole in the wing.

<https://www.youtube.com/watch?v=IgQ3ekcvyRA>

https://upload.wikimedia.org/wikipedia/commons/7/7f/STS-107_Columbia_entry_imaged_from_ground.jpg

A quote is displayed within a rectangular frame that has a textured, light-colored background resembling a wall or paper. The text is in a black, sans-serif font and is centered within the frame.

“...never did we talk about [the RCC] because we all thought that it was impenetrable ... I spent fourteen years in the space program flying, thinking that I had this huge mass that was about five or six inches thick on the leading edge of the wing. And, to find after Columbia that it was fractions of an inch thick, and that it wasn't as strong as the Fiberglass on your Corvette, that was an eye-opener, and I think for all of us ... the best minds that I know of, in and outside of NASA, never envisioned that as a failure mode.”

Not only were foam strikes a known problem, on tiles, reinforced carbon carbon, or RCC, on the wings was known to be an issue

https://en.wikipedia.org/wiki/Space_Shuttle_Columbia_disaster

Conclusion

Disasters in Software (and How to Avoid Them)



There is no easy answer. Software isn't easy. Keep in mind this reality: we can't stop failures from occurring. We can harden systems, we can perform rigorous testing, and yet....something may still fall through the cracks. Nobody ever said this was easy. I'm sure that all of the advice I gave, you could come up with scenarios and situations that I may not have explicitly covered. Some of the best advice I got was in a project management 2-day course. Lots of good advice, but once the teacher said, "I never said PM-ing was easy." These are guiding principles. They don't magically make everything better. It takes time for change to happen. And some companies, you may want to consider running away from in horror (that's easier said than done).

<https://unsplash.com/photos/geNNFqfw48>



But my hope is that we can start to diminish the fear around software, that we can continually raise the bar and make our bits better.

https://unsplash.com/photos/VT8l5wC_pTA

Disasters in Software (and How to Avoid Them)

Jason Bock

Remember...

- <https://github.com/JasonBock/Presentations>
- References in the notes on this slide

- * To keep a Boeing Dreamliner flying, reboot once every 248 days -
<https://www.engadget.com/amp/2015/05/01/boeing-787-dreamliner-software-bug/>
- * Inside the Ford/Firestone Fight -
<http://content.time.com/time/business/article/0,8599,128198,00.html>
- * Failed landings of SpaceX rockets -
<https://www.youtube.com/watch?v=bvim4rsNHkQ>
- * Production postmortem: data corruption, a view from INSIDE the sausage -
<https://ayende.com/blog/180481/production-postmortem-data-corruption-a-view-from-inside-the-sausage>
- * Killed by Code: Software Transparency in Implantable Medical Devices -
<https://www.softwarefreedom.org/resources/2010/transparent-medical-devices.html>
- * A Collection of Well-Known Software Failures -
<http://www.cse.psu.edu/~gxt29/bug/softwarebug.html>
- * This week's failed Russian rocket had a pretty bad programming error -
<https://arstechnica.com/science/2017/11/this-weeks-failed-russian-rocket-had-a-pretty-bad-programming-error/>
 - * Soyuz fails to deliver 19 satellites from Vostochny -
<http://www.russianspaceweb.com/meteor-m2-1.html>

- * Apparent video of crash (not a great vid, but worth an image?) -
https://www.youtube.com/watch?v=N7w63jE_FpY
- * USS McCain collision ultimately caused by UI confusion -
<https://arstechnica.com/information-technology/2017/11/uss-mccain-collision-ultimately-caused-by-ui-confusion/>
- * Denver Airport Baggage System Case Study -
http://calleam.com/WTPF/?page_id=2086 and <http://calleam.com/WTPF/wp-content/uploads/articles/DIABaggage.pdf>
- * Ten of the world's most disastrous IT mistakes -
<https://www.pcauthority.com.au/feature/ten-of-the-worlds-most-disastrous-it-mistakes-264645>
- * Knight Capital Group -
https://en.wikipedia.org/wiki/Knight_Capital_Group#2012_stock_trading_disruption
- * Mars Climate Orbiter -
https://en.wikipedia.org/wiki/Mars_Climate_Orbiter#Cause_of_failure
- * GitLab suffers major backup failure after data deletion incident -
<https://techcrunch.com/2017/02/01/gitlab-suffers-major-backup-failure-after-data-deletion-incident/>
- * Amazon Says One Engineer's Simple Mistake Brought the Internet Down -
<http://gizmodo.com/amazon-says-one-engineers-simple-mistake-brought-the-in-1792907038>
- * Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region - <https://aws.amazon.com/message/41926/?tag=gizmodoamzn-20&ascsubtag=d4705d0fcbaa3d7dea98dccc155390f1065cbe&rawdata=%5Bt%7Clink%5Bp%7C1792907038%5Bau%7C452356546%5Bb%7Cgizmodo>
- * Everything You Need to Know About Cloudbleed, the Latest Internet Security Disaster - <http://gizmodo.com/everything-you-need-to-know-about-cloudbleed-the-lates-1792710616>
- * Incident report on memory leak caused by Cloudflare parser bug -
<https://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/>
- Windows Azure storage issue: Expired HTTPS certificate possibly at fault -
<http://www.zdnet.com/article/windows-azure-storage-issue-expired-https-certificate-possibly-at-fault/>
- * Windows Azure Service Disruption from Expired Certificate -
<https://azure.microsoft.com/en-us/blog/windows-azure-service-disruption-from-expired-certificate/>
- Back to Basics: Explore the Edge Cases or Date Math will Get You -
<http://www.hanselman.com/blog/BackToBasicsExploreTheEdgeCasesOrDateMathWillGetYou.aspx>
- * Zune bug explained in detail - <https://techcrunch.com/2008/12/31/zune-bug-explained-in-detail/>