Note: Link available here (requires VLE access).

Initial Post: Gennaro Coppola

I want to report a story that happened to my team at the beginning of my career. My team was maintaining an e-commerce for a company, developing new integrations and features. The working model was very simplistic: developers used to work in branches and merge their solutions in develop and then at intervals of some months develop branch would merge in master and being deployed by another team.

No code reviews, no testing except some for the new features. What happened is that one developer during work changed some code to "test" something, and then committed this "test code" because he forgot about it and didn't even check the changelist of the commit. When the code has been deployed caused runtime error during payment. Users complained to the back office and when operations understood that the site was not working correctly deployed the previous working version.

Causing a total stop of sales for 8 hours with potential damage near the one million dollars. Fault can be attributed to the single developer because he made a mistake, or to the management who considered implementing a proper software development lifecycle "too expensive". Because a code review process could have highlighted if there was code out of the scope of a feature, or possible harassing code and an automated test suite even a simple one who would focus on the main features of the portal would have found the problem before to deploy in production.

Peer Response: Shan Swanlow

Hi Gennaro,

This is a tricky situation. Even though computers are a core part of information systems, humans are just as important and this does introduce risk that is extremely difficult to anticipate. As Suresh mentioned, CI/CD is really useful for being able to prevent this type of situation from happening: I've previously used Bitbucket Pipelines to achieve the same goal.

In my experience, however, one issue with using CI/CD for automated testing is that it works well only if you have near perfect test coverage, this is typically easy if tests are built from the beginning of the project. Based on your responses (forgive me if I'm wrong), it seems like there are not enough resources or permissions for your team to create a full CI/CD pipeline or get full test coverage. One other approach that can be used is static code analysis, which is essentially some tool that reads source code and tries to detect errors in the code without executing or compiling it. For example, if a live URL was changed to a test URL, the code analysis tool would see this in the source file and tell you to change it. It could be a useful solution in this situation since it requires a very small time investment and can detect mistakes of this nature. The downside is that it's highly inaccurate and produces a lot of false positives depending on the tool you use. Do you think this type of approach would be a good temporary solution if you're not able to get permission to add proper tests?

Note: within the same thread, I provided a second peer response to Andrey, regarding a different topic. I have attached his original post, my response to it, and his subsequent response.

Reply: Andrey Smirnov

Hi Gennaro,

Thanks for providing more details surrounding the incident. What you have described is symptomatic of extremely poor engineering management, from the complete lack of transparent and collaborative process down to how the blame was assigned. Leadership should be about setting direction and enabling people to (hopefully) do the best job of their lives, not about micromanaging tasks and individuals. Good leaders take complete responsibility for failure, but give all credit to their teams when reaching success. In that sense, being a leader is quite a thankless job that requires a lot of discipline and a healthy ego.

All that said, it is rather easy to criticise the above situation and the people responsible for it. However, from what I have seen in my career, the presence of poor software engineering practices cannot always be fully attributed to the inexperience of the IT manager, but is often conditioned by a number of serious, often financial, constraints and a very real pressure received from the business side. Perhaps in the case that you have described it was about all of these factors combined.

Kind regards,

Andrey

Peer Response: Shan Swanlow

Hi Andrey,

This is an interesting perspective because it highlights the importance of understanding how the business side can harm IT departments, despite having good intentions. I've never really considered this before, so I wanted to ask: if you were the IT manager in this case, how would you convince the business managers to properly invest into a good SDLC? There is a clear case for investment: losing nearly 1 million dollars is a significant problem and improving the development process will probably cost less. However, it takes a lot of time to improve development processes as well- there's a lot of research and learning involved, it takes time for tangible results to appear, and this makes business teams less likely to approve such a request (at least in my experience). From your experience, is this a common obstacle, and if so, how would you navigate it?

Reply: Andrey Smirnov

Hi Shan,

That is an excellent question, however it is the one to which I cannot offer a clear-cut answer. Advocating for an investment into a better SDLC from the perspective of avoiding a potential failure might not be a good strategy when taking to the business. First of all, you will have to explain why the current setup is deficient, which will quickly steer the discussion into a technical territory that the business representatives might not be able to navigate. This approach is almost guaranteed to fail.

I would start with something many people in engineering tend to underestimate, and that is building relationships. Relationships matter a lot on this level and as an IT manager, you need to have that buy-in with the business where it is oftentimes less about logic, and more about communication, heart and conviction.

If having good relationships based on mutual trust and respect still does not help, which is very much possible given the sensitive (financial) nature of the discussion, then the best thing you can do is to be mindful about the specific constraints that you and your team(s) are operating under. Sometimes you simply have to "bite the bullet" and do the best with what you have. That might mean, as an example, clearly establishing the team's maturity level in their surrounding context, and tailoring the way you support the team accordingly. Larson (2019), in his insightful book "An Elegant Puzzle: Systems of Engineering Management", proposes four stages of the team evolution, namely falling behind, treading water, repaying debt and innovating. Depending on where the team is located in this "continuum", there are different strategies that can be employed in order to help progress the team to the next maturity level, as well as to mitigate the risk of technical failures. All that said, sometimes having an incident in production akin to what Gennaro described above will be the only thing that enables positive change, which is of course rather unfortunate.

References

Larson, W. (2019) An Elegant Puzzle: Systems of Engineering Management. 1st ed. Stripe Press.

Note: Another discussion thread was posted, which I replied to. The link is available here (requires VLE access)

Initial Post: Sergio Rafael Zavarce Caldera

Between 6 and 7 December 2018, a software failure brought the O2 network down, affecting data services for more than 30 million users in the UK. Other companies that relied on O2 network such as Lycamobile, Giffgaff, Sky Mobile and Tesco Mobile were also affected. London's bus stop screens and systems aboard more than 8500 London buses also failed due to this incident (Goodley, 2018). O2 offered several types of compensation for its customers depending on their plan, including discounts or credit of days' worth charges.

The failure was caused by an out-of-date licence in a software used by several hardware devices distributed by Ericsson. In total, 11 countries were affected by this failure, causing considerable reputation and monetary damage. O2 was asking for a compensation of as much as 100 million pounds from Ericsson (Goodley, 2018).

Ericsson (2018: 1) declared:

"During December 6, 2018, Ericsson has identified an issue in certain nodes in the core network resulting in network disturbances for a limited number of customers in multiple countries using two specific software versions of the SGSN–MME".

In another press release, SoftBank Corp., a Japanese telecommunications company apologized for the incident that affected their mobile customers nationwide (SoftBank Corp., 2018).

References:

SoftBank Corp. (2018). Apology for Mobile Communication Service Troubles [press release]. 6 December. Available at: https://www.softbank.jp/en/corp /news/press/sbkk/2018 /20181206_02/ (Accessed: 10 May 2021).

Ericsson (2018). Update on software issue impacting certain customers [Press release]. 6 December. Available at: https://www.ericsson.com/en/press-releases/2018/12/update-on-software-issue-impacting-certain-customers (Accessed: 10 May 2021).

Goodley, S. (2018). O2 poised to receive millions from Ericsson over software failure. Available at: https://www.theguardian.com/business/2018/dec/09/o2-poised-to-receive-millions-from-ericsson-over-software-failure (Accessed: 10 May 2021).

Reply: Shan Swanlow

Hi Sergio,

This was an interesting example because it highlights the challenges in implementing controls when there are multiple contributors to the processes of an information system. As Solomon mentioned, it was found that the root cause of the issue was an expired security certificate. In 2019, a investigation report was published by the UK's governing body for communication (Ofcom). The report stated that the expired certificate was hardcoded into Ericsson's product (Ofcom, 2019). Hardcoding can be reasonable in certain contexts, but in this context, it would be considered a bad practice (CISQ, 2021).

Ofcom (2019) also stated that O2 responded in a satisfactory manner, along with having acceptable controls and testing procedures in place. They further argue that due to the nature of this problem, it would not be realistic to anticipate, or check for this type of bug in advance. I agree with this, however, when something happens that can cause an outage at a global scale, it may be worthwhile to find ways of minimizing the impact. This issue was caused by human error, and I argue that human error is unpredictable, so it would be more logical to come up with a solution that can apply to any unpredictable cause, as opposed to creating a direct solution to some specific, yet unpredictable cause. Backup infrastructure/redundancy could be one way of achieving this, but could significantly increase costs and therefore might not give a return on investment if it isn't used enough. What do you think- is there merit to this kind of approach?

References

CISQ. (2021) CWE-1051: Initialization with Hard-Coded Network Resource Configuration Data. Available from: https://cwe.mitre.org/data/definitions/1051.html [Accessed 19 May 2021].

Ofcom. (2019) O2 network outage Decision to conclude investigation into Telefónica UK Limited's compliance with section 105A(4) of the Communications Act 2003. Available from: https://www.ofcom.org.uk/__data/assets/pdf_file/0014/175010/o2-network-outage-cceb.pdf [Accessed 19 May 2021].