Eva Dukerschein

DevOps

7/14/2024

Assignment 8.2

The Dangers of the Change Approval Process

The change approval process is a mainstay in many industries. It is a very old framework for assessing and managing risk that originates in the 1940’s. The purpose behind a change approval process is that any submitted changes to a business are audited and reviewed before the change may be implemented. In this way it is supposed to avoid creating errors and issues that the proposer of the change may not have foreseen. For a very long time and for many industries still, this process is not only smart, but is incredibly necessary. However, change approvals can take time depending on the framework and in the world of technology, time is a valuable commodity. As such, when it comes to change approval processes, it is important to find a framework that is efficient and actually works. If the framework is slow, inefficient, and ineffective, the company stands much to lose especially against competitors with a leaner and more suitable change approval process.

Change management has an extremely broad definition. It is defined by the International Professional Practices Framework (IPPF) as “technology changes that affect an organization’s systems, programs, or applications.” There are a number of different frameworks that can be used to manage changes in a company’s systems. The goal isn’t necessarily to eliminate all of these, but rather to streamline and perfect a framework that acts in the organization’s interests. Processes that might be subject to change approval processes application code revisions, system upgrades, infrastructure changes, security updates and patches, code revisions, regular updates and patches, and more. Typically however, the end goal of all change approval processes are to reduce risk and avoid outages or incidents that could be introduced by bad code or configurations. In addition, an effective change approval process allows future changes to be repeated, measured, and audited.

To better understand the change approval process, it is important to understand what risks are being mitigated. In addition to avoiding outages, which have cascading effects such as impacting business objectives negatively and upsetting customers, other risks include attrition of staff, quality control issues, security vulnerabilities, general future instability that may lead to outages with future changes, and even compliance failures. There are also different tiers of changes that will carry these many risks. Standard changes are low-risk and follow procedures that make them very safe to implement. Normal changes vary in risk level. They are typically new changes or experimental changes that will traditionally need to see some sort of review before being implemented. Finally, emergency changes are changes that have to happen as soon as possible. For these changes it is not possible to wait for a formal approval process. Typically these occur when a previous failure was not caught and there is now an emergency to rectify.

Traditional change approval processes were typically clunky affairs. There would be multiple levels of trade-off across multiple teams. Ticketing systems might be used to attempt to standardize and categorize the proposed change. As the change proposal went up the ladder it was reviewed by individuals with less and less familiarity with the code. That is assuming that the approver even had time to review the code at all. If they were busy or did not prioritize the approval above other duties, the change could be left in limbo for long stretches of time. Not only was this demotivating to the team who submitted the change request, it also put the business behind their competitors. Using old change approval methods could create a ninefold increase in inefficiency. In the tech industry, every second matters and this kind of slow-down could make or break a company. And even after passing through all of these approval gates, there may still be an error that went uncaught.

Modern DevOps teaches us that continuous integration and continuous deployment is a more efficient and safe way to produce high quality software in a timely manner. These older methods were not in line with the way that highly agile interconnected teams worked. As such, new frameworks needed to be made with regard to safely introducing changes. It is important to standardize deployment when creating a new change approval process. Having too many tools, systems, and technologies present in the pipeline can slow down approvals and reviews massively. Having standardized patterns for building the software or environments is essential. This makes approving changes easier for people who are less familiar with the code. Having a standardized platform allows for changes to be uniform, autonomous, replicable, and fast. Speed must be taken extremely seriously. Small changes should be the priority. As changes grow in size, the time it takes to review and approve them increases geometrically. Instead of proposing and submitting several changes in one package, it is better to create smaller batches more frequently. Multiple manual reviews lead to long wait times. A better option is to automate what changes are possible and feasible to automate. Many approvals can even be made at the floor level for standard or emergency changes. Having a second pair of eyes on the code as it is being developed – a practice known as pair programming – can go a long way in detecting errors. Peer review is also extremely effective as the code is being reviewed by people most familiar with the product, as opposed to a CIO who has a higher-level understanding of the technology.

In conclusion, change approval processes, in one form or another, are integral to the process of creating stable code. However, the methods vary both in scale and in effectiveness. When working with a DevOps framework, the approval process must also conform to a DevOps framework and prioritize speed, automation, and cross-team inter-connectivity. Formal, multilevel approval chains work against a company’s best interest when it comes to delivering a high-quality product in a reasonable time-frame. As such it is necessary to understand where the risks in such a process lie and how to overcome them.

Works Cited

*Change Management Is Broken: Here’s How To Fix It - DZone DevOps*. (n.d.). Dzone.com. https://dzone.com/articles/change-management-is-broken-heres-how-to-fix-it

*How Does DevOps Handle Change Management?* (2020, October 17). Cprime. https://www.cprime.com/resources/blog/how-does-devops-handle-change-management/

*IT Change Management 3rd Edition Supplemental Guidance | Practice Guide*. (n.d.). Retrieved July 14, 2024, from https://www.theiia.org/globalassets/documents/content/articles/guidance/gtag/gtag-it-change-management/gtag\_it\_change\_management\_3rd-edition.pdf

‌Lachhman, R. (n.d.). *Change Management in the World of Continuous Delivery | Harness*. Harness.io. https://www.harness.io/blog/change-management-continuous-delivery

‌