International Journal of Research in Engineering, Science and Management
Volume 5, Issue 7, July 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

Change Management System for Software Production Environments

G. M. Ananya*
Student, Department of Information Science and Engineering, RV College of Engineering, Bengaluru, India

***Abstract*: Changes in the IT production environment occur on a day-to-day basis. These need to be managed from their origination to termination. These modifications need to be recorded and approved at the appropriate level in order to manage these changes well. Change Management aims to help firms recognize and minimize the risks of modifications to the IT production environment. Detection of changes to the production environment, their handling, recording, and approval/refusal of these changes are all part of Change Management. An organization that is see-through about its work would have its change management process avert change-coupled events, minimize the negative impact on the conveyance of services to its users and clients, and conduct a legal, systematic methodology in handling all the changes. This paper discusses the significance of change management and system optimization strategies to manage changes in the production environment and analyses the performance of the same.**

***Keywords*: Change management, batch jobs, production environment, pagination, infinite scrolling.**

1. Introduction

Implementing an application is a repetitive process. Developers amend the application codebase and recurrently move these modified components to the test environment for user feedback and acceptance. The application is moved to production when it is stabilized at a given point in its software development lifecycle. After a software version is released for general use, all the issues must be fixed, and solutions and their implementations need to be tested and integrated into the production environment. This process also referred to as production control, is the spirit of the change management operation.

Companies vary widely in their approaches to change management. Some obligate much of the responsibility to its developers, while others lay out elective cycles to keep a more significant level of control [2]. While developers prefer firm-developed tools to achieve this, change management experts prefer to handle changes using batch-oriented methods to move systems between different environments. Software engineers might be expected to initiate change bundles after the application is moved to production. Larger companies now use a combination of these approaches. A standard change management process involves developers selecting certain resources to deploy to production and creating a change management bundle, and moving this bundle to an acceptance test. A few associations might use a computerized interaction to import the substance, to accomplish better incorporation with their business cycle. At the point when the application is prepared for release, the production control personnel starts deployment of the application to the production environment. When end users begin to use the software, the role in the change management process is changed to that of application maintenance support. Starting now and into the foreseeable future, gradual updates to creation are worked with by production control personnel using the change management facility.

Change management (change enablement, change control) is an equilibrium between efficiency and risk, i.e., deploying changes in an appropriate and timely manner while minimizing the risk of deployment [3]. Alternatively, some organizations attempt their best to avoid different types of risks, disregard standard procedures and push out changes as soon as possible, which is likely to lead to change-related incidents and debilitating outages. Framework and infrastructure modifications materialize persistently and are often propelled by changing customer requirements, the addition of new features and functionality, releases and patches, and solutions to user's incidents or problems. IT managers need to act in accordance with a standard procedure so that they never deploy a patch with faulty changes or additions while the end users are using it live. A methodology should be fostered that will assist with making a consolidated change management practice that adjusts hazard and throughput of transformation. Protecting the live environment and deploying change and new functionalities in a timely manner are the two objectives of change management. These two might appear to be in conflict at times but evaluating the risk at numerous stages of a change's circution can help achieve both and keep a check on the deployment process.

The Change Manager makes sure that all changes are overseen in a controlled way, including standard changes and crisis upkeep connecting with business cycles, applications, and framework. This incorporates change norms and strategies, impact evaluation, prioritization and approval, emergency changes, tracking, detailing, closure, and documentation. A Change Manager must,

a) Establish the process for change management and

*Corresponding author: ananyagm.is18@rvce.edu.in

ensure that each change follows the complete procedure to ensure minimum disruption to IT services (The change management practice should be formalized through a management-approved policy) [6].

b) Assess all solicitations for change to decide the effect on business cycles and IT services, and to evaluate whether the change will unfavorably influence the operational environment and present inadmissible risk.
c) Approve acceptable changes.
d) Guarantee that changes are logged, focused on, evaluated, approved, arranged, and booked, and are presented in a controlled and composed way.
e) Liaise with all essential parties to facilitate change building, testing, and execution, as per plans.
f) Cautiously oversee emergency changes to limit further occurrences and guarantee that the change is controlled and happens safely. Confirms that crisis changes are suitably surveyed and approved after the change.
g) Maintain a tracking and reporting system to document rejected changes, communicate the status of approved and in-process changes, and complete them.

The framework of the Change Management System proposed in this paper is implemented using Angular for the client side, Spring Boot for the server side, and a database. The results obtained, the efficiency of the framework, scalability, and other aspects will be discussed in context with this implementation.

2. Design

The Change Manager System is a full-stack application which is a detective control system that monitors changes in the production) and attempts to correlate these changes with Change Requests. The architecture of the Change Management System implemented is shown in figure 1. Any exceptions (i.e., changes that could not be correlated against a valid change approval) need to be approved by the owner of the changed component using the application's web interface.

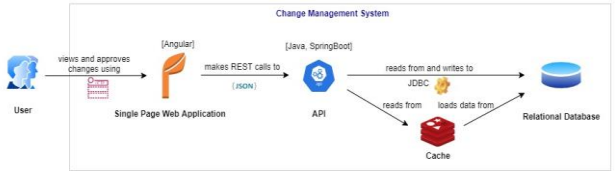

Fig. 1. Architecture of CMS

A Change Request is required when there is an addition, modification, or removal of any IT Service, system, or component(s) that are part of a production environment as well as all services with agreements that specifically state service levels and environment up-time, are subject to the Change Management process and policies.

The applications containing changes requiring approval could be loaded using two approaches, one by cross-validating the user's permissions through policy entitlements, or by using a batch job, which caches the entitlements into a database. A batch job is a predefined group of processes submitted to the framework to be performed with practically no interaction between the client and the system. These positions that don't need client communication to run can be handled as batch jobs. Several batch jobs can be active at the same time. The former approach's performance was highly inconsistent with response times of the API call varying from 3 seconds to over 80 seconds depending on the number of changes pending approval for an application. However, the latter process of using a batch job is faster in response time as the data is fetched by querying a database whose data is refreshed every day from entitlements API.

A. Types of Changes

1) Standard changes

Standard changes are generally safe, usually rehashed, and pre-supported. They're performed regularly and follow a reported, supported process. For instance, making another example of an information base is a standard change. For some organizations, standard changes are a superb chance for automation. A few organizations report that many as 70% of the standard changes can be handled using automation - opening up their groups to zero in on typical and emergency changes.

2) Normal changes

Normal changes are non-crisis changes that don't have a characterized, pre-supported process. For instance, upgrading to a new system, relocating to another data center, and execution enhancements are ordinary changes. They're not standard and repeatable. There are chances included. But at the same time, they're not emergencies. This implies they can go into the typical change management line for risk appraisal and endorsement.

3) Emergency changes

These changes arise from an unexpected error or threat and need to be addressed immediately - usually to restore service for customers or employees or secure systems against a threat. For example, implementing a security patch, and dealing with a server outage are emergency changes. The urgency of emergency changes means they need to be handled on a much tighter timeline because the risk of a lengthy review process is higher than the risks involved with resolving the issue quickly.

B. Debottlenecking

1) Lazy loading

Lazy loading is a technique that defers the loading of non-critical resources at page load time. Instead, these non-critical resources are loaded as and when needed.

2) Pagination

Data split across numerous pages utilizing pagination. However, this is the old way of showing information, this is pertinent in certain circumstances when the client is looking for something specific inside the rundown of results as it gives better control of the data.

3) Infinite scrolling

Infinite scrolling uses lazy-loading to load data from the server. It records the scroll position of the grid, based on which the data is loaded onto the client-side. Infinite scrolling allows

the grid to lazy-load rows from the server depending on the scroll position of the grid. In its simplest form, the more the user scrolls down, the more rows get loaded. The grids have an 'auto extending' vertical scroll. For data with thousands of records, the records seem to be loading infinitely. That means as the scroll reaches the bottom position, the grid will extend the height to allow scrolling even further down, almost making it impossible for the user to reach the bottom. However, once the last record is reached, the grid stops loading data and ceases infinite scrolling. The grid will ask the application, via a data source, for the rows in blocks. Each block contains a subset of rows of the entire dataset. Figure 2 is a high-level overview of infinite scrolling.

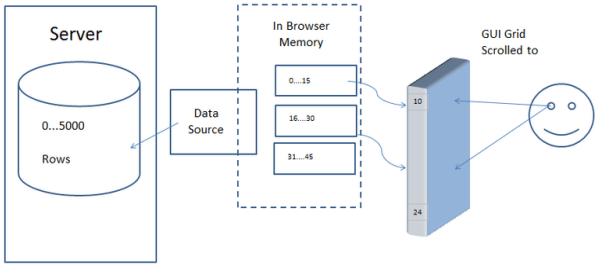

Fig. 2. Infinite scrolling

When the records from one block are already loaded, and the user requests for the next set of records, the corresponding records of the succeeding block are loaded. In figure 2, the remote server provides the rows from a database to the data source. The grid stores the blocks in a cache. During the implementation of this concept, an option to either set a limit on the number of blocks stored or to never expire them can be added. If the limit is set, then as the user scrolls down, the previous blocks will be discarded and will be loaded again if the user scrolls back up. Sorting or filtering must be done on the server side.

4) Query optimization

An inefficient query will drain the production database's resources, and cause slow performance or loss of service for other users if the query contains errors. It's essential to optimize the queries for minimum impact on database performance [5]. The performance of the production database is too critical to have unclear or ambiguous requirements. Thus, the

requirements need to be as specific as possible and should be confirmed with all stakeholders before running the query. Some of the ways to optimize a SQL query for faster results are,

1. SELECT 'fields' instead of SELECT * will point the database to querying only the data you need to meet the business requirements.
2. Create Joins with INNER JOIN and not WHERE.
3. To filter data, use WHERE instead of HAVING. The database in Change Management System is queried only to fetch records based on certain conditions. HAVING clause is evaluated after WHERE. Hence, to pull data of a user, WHERE proves to be more efficient.

3. Advantages

The objective of the Change Management process is to direct a formal, normalized procedure in the treatment of all changes to be straightforward about our work, forestall change-related occurrences, and limit the adverse consequence on the conveyance of administrations to our clients and clients [4]. The following are the advantages of a Change Management System,

1. Works on the nature of changes by guaranteeing that a standard procedure is utilized to deal with all changes and control the effect on the everyday tasks of the association.
2. Decreases the potential for change-incited occurrences by giving a norm, formal, way to deal with taking care of a change.
3. Ensures that all changes are logged (or captured) in a centralized repository where the information can be shared by other processes.
4. Evaluates every one of the progressions in light of their effect, advantage, and hazard to the business and endorse or decline the solicitation for change appropriately.
5. Plans all changes in view of the necessities of the business, the accessibility of various resources, and consideration of other changes being pushed out to production.
6. Ensures that all changes are appropriately tested and certified and that the appropriate implementation and remediation plans are available.
7. Gives convenient correspondence of progress plans and timetables as well as the situation with all changes to fitting partners and impacted clients.
8. Gives general announcing capacities against the vault of changes, including moving data and explicit measurements applicable to the cycle.

4. Conclusion

Change Management Software allows companies to manage, monitor, and optimize the change management process in their organizations. Hence, this software is extremely important to reduce manual management and approval of changes. The software implemented is capable of collecting, recording, and approving the changes in the production environment on consent from the product owner. The system shows a performance that is on par with Change Management Systems of today in the IT sector.

Acknowledgment

I would like to thank the Department of Information Science and Engineering, RV College of Engineering, Bengaluru, for the constant support in the field of Research and Development. I'm indebted to my guide, Prof. Priya D., who has made a significant contribution to the improvement of this research paper.

References

[1] Pandey Amit and Mishra Sushma, "Understanding IT Change Management Challenges at a Financial Firm," 2014.

[2] https://www.atlassian.com/itsm/change-management

[3] https://www.infotech.com/research/ss/optimize-it-change-management

[4] https://www.userlane.com/change-management-software/

[5] https://uit.stanford.edu/service/changemgt

[6] https://www.sisense.com/blog/8-ways-fine-tune-sql-queries-production-databases/

[7] Łopatowska, Jolanta, "Change Management in Production Planning and Control," 2007.

[8] https://www.techtarget.com/searchitoperations/tip/How-change-management-and-configuration-management-differ-in-IT

[9] https://freshservice.com/change-management-software/itil-change-management

[10] https://linfordco.com/blog/change-control-management/

[11] Natalia Iakymenko, Anita Romsdal, Erlend Alfnes, Marco Semini & Jan Ola Strandhagen, "Status of engineering change management in the engineer-to-order production environment: insights from a multiple case study," International Journal of Production Research, 58:15, 4506-4528, 2020.