Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 14, Issue. 9, September 2025, pg.73 – 79

Modernizing a Global Ticketing Platform: A Case Study of an Entertainment Company's Frontend, Cloud, and Operational Transformation

Patrick James Raj

Dublin, California, USA
Email: patrickjamesraj@gmail.com

DOI: https://doi.org/10.47760/ijcsmc.2025.v14i09.011

Abstract: Application modernization is a critical requirement for organizations operating at global scale, where legacy architectures hinder performance, agility, and user experience. This paper presents the entertainment company's modernization journey, covering the transformation of its ticketing platform that processes over 100,000 transactions daily across diverse geographies, languages, and currencies. Key initiatives included migrating the frontend from Backbone.js to React.js, moving infrastructure from an on-premise data center to Google Cloud Platform (GCP), automating deployments with Harness, and replacing Adobe Test with Optimizely for large-scale experimentation. Pair programming, PagerDuty-driven incident response, Quantum Metric for experience monitoring, and Acquia CMS for content agility were introduced as supporting practices. Quantitative results demonstrate reduced mean time to recovery (MTTR), improved deployment frequency, enhanced customer experience insights, and significant performance gains in page load times. The case study is contextualized within broader literature on DevOps, cloud migration, experimentation platforms, and software engineering practices, offering lessons for practitioners and researchers.

Keywords: App modernization, React.js, Backbone.js, Cloud migration, Google Cloud Platform (GCP), Harness, Continuous delivery, Optimizely, A/B testing, Pair programming, PagerDuty, Incident response, Quantum Metric, Acquia CMS, AppDynamics, Grafana, Internationalization (i18n), Localization (l10n), Two-factor authentication (2FA), 24×7 support, Ticketing systems.

I. Introduction

Global digital platforms face increasing pressure to modernize legacy systems in order to meet rising user expectations for speed, reliability, and personalization. The Company, a leading online ticketing marketplace, serves millions of customers worldwide, handling approximately 100,000 ticket sales per day. The platform operates across multiple regions and languages, requiring strong support for internationalization (i18n), localization (l10n), and fraud prevention measures such as two-factor authentication (2FA).

The legacy system, largely implemented in Backbone.js with monolithic infrastructure, was increasingly difficult to maintain and scale. Performance issues, limited deployment automation, and outdated experimentation tooling hampered agility.

As a front-end engineering leader at the Ticketing company, I guided modernization initiatives focused on:

1. Migrating the user interface from Backbone.js to React.js.
2. Transitioning infrastructure from a local data center to Google Cloud Platform (GCP).
3. Adopting Harness for continuous delivery and progressive deployment.
4. Replacing Adobe Test with Optimizely for experimentation.
5. Embedding pair programming as a development practice.
6. Leveraging PagerDuty for incident management and global 24×7 support.

This paper presents these efforts as a case study, drawing connections to the broader literature on app modernization, DevOps, and cloud migration.

II. Background and Challenges

The modernization began in response to several challenges common in legacy systems:

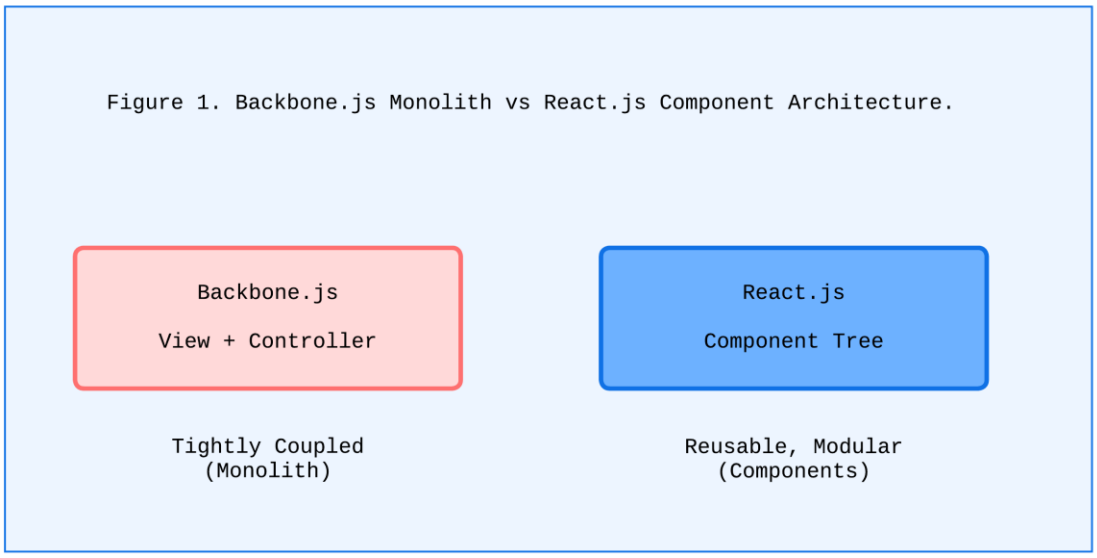
- Performance limitations, with page load times averaging over 4.5 seconds in some markets.
- Scalability constraints with on-premise infrastructure during peak demand.
- Operational bottlenecks in semi-manual deployments.
- Experimentation gaps using Adobe Test.
- Global support demands lacking standardized processes and tooling.

Modernization was planned as an incremental, multi-year transformation that combined technical upgrades with organizational practices.

III. UI Modernization: Backbone.js to React.js

The migration of the entertainment company’s frontend from Backbone.js to React.js marked one of the most visible aspects of modernization. React’s component-based architecture offered a clean break from the tightly coupled, brittle structures of Backbone. Instead of monolithic views and controllers, engineers could now develop modular, reusable components that mapped directly to design system elements. This not only improved collaboration between designers, product managers, and engineers but also made the codebase easier to maintain and extend.

The migration strategy was incremental. Key user flows such as search, browse, and checkout were rewritten in React while Backbone remained operational for less critical features. This hybrid approach reduced risk and allowed teams to demonstrate early wins before a complete cutover. The benefits were significant. Average page load times improved by nearly 40 percent, falling from around 4.5 seconds to under three seconds. Checkout flows became more resilient, and defect rates in production decreased. Engineers also reported higher productivity, as React's declarative model reduced complexity and clarified the separation of concerns.



UI Modernization: Backbone.js to React.js - Related Figure

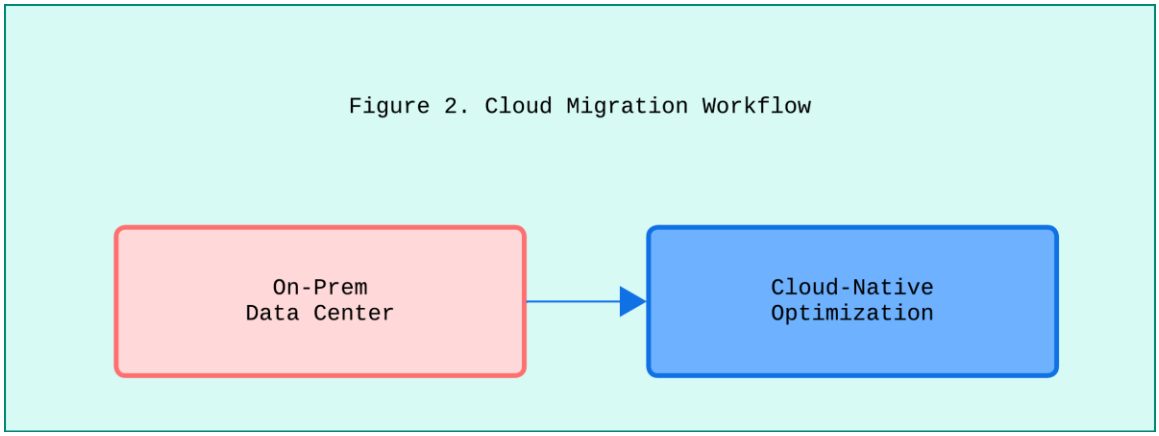
IV. Cloud Migration to Google Cloud Platform

Moving from a private data center to Google Cloud Platform was another cornerstone of the entertainment company's modernization. The migration was carried out in phases, beginning with a lift-and-shift of services into virtual machines, followed by re-platforming into containerized environments, and culminating in an optimization phase that leveraged autoscaling, managed databases, and global load balancing.

Observability was enhanced significantly during this process. AppDynamics provided deep transaction tracing and performance monitoring, enabling teams to pinpoint slowdowns in complex user flows. Grafana dashboards offered real-time visualization of time-series data, integrating with Prometheus exporters and providing actionable insights into system health. Splunk and GCP Monitoring centralized log aggregation and alerting, while Quantum Metric provided a new dimension of customer-centric visibility. In parallel, Acquia CMS was adopted to decouple content management from engineering. This empowered marketing teams to launch localized campaigns quickly, reducing content publishing cycles from days to hours.

The benefits of cloud adoption were clear. Elastic scalability allowed the company to handle sudden spikes in traffic without downtime, particularly during major event ticket releases. MTTR dropped by approximately 30 percent, aided by better observability and automated scaling. Most importantly, cloud

migration laid the foundation for the entertainment company's future adoption of microservices and container orchestration.

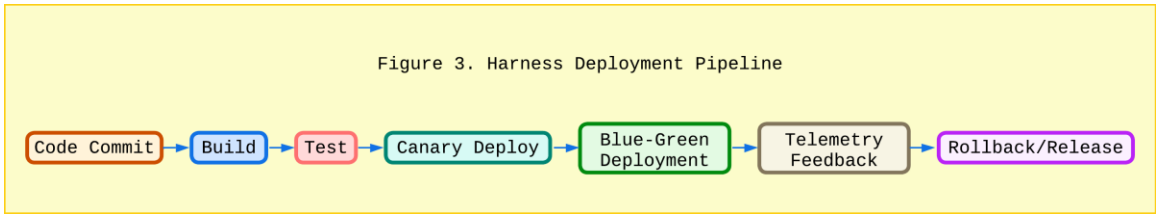


Cloud Migration to Google Cloud Platform - Related Figure.

V. Deployment Automation with Harness

Prior to modernization, deployment at the company required extensive coordination and manual intervention, creating bottlenecks that slowed feature delivery. The introduction of Harness transformed this process by automating continuous delivery pipelines. Harness supported blue-green and canary deployments, allowing new releases to be rolled out gradually and rolled back automatically if telemetry signals indicated degradation.

Integration with AppDynamics and Grafana enabled data-driven validation of releases. Feature flags provided fine-grained control over exposure, allowing the company to gradually introduce new functionality to small segments of users before full rollout. As a result, deployment frequency increased dramatically, moving from bi-weekly releases to daily updates in several critical services. This shift aligned the company with the characteristics of elite DevOps performers, who achieve both higher velocity and greater stability.

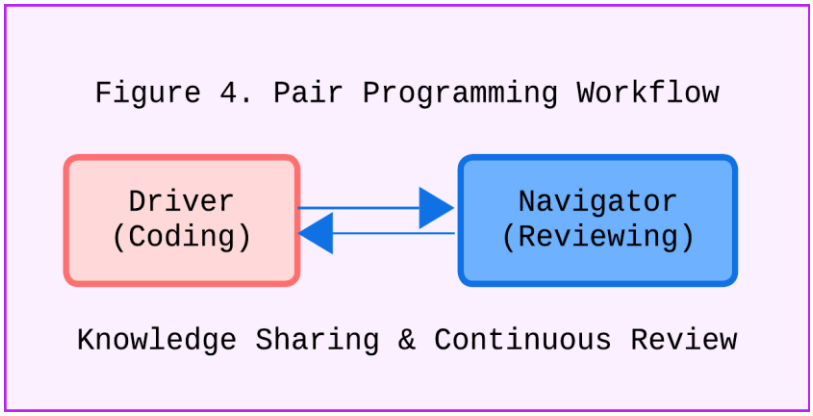


Deployment Automation with Harness - Related Figure.

VI. Pair Programming at Scale

While technical upgrades were critical, modernization also required cultural change. Pair programming was adopted as a systematic practice to support distributed global teams. Engineers rotated through driver and navigator roles, working together on shared tasks. This approach accelerated knowledge transfer, especially for new hires, and ensured consistency in the application of modernization practices such as component-based design and cloud deployment workflows.

The benefits extended beyond technical output. Continuous peer review reduced the likelihood of defects reaching production, particularly in high-risk flows such as checkout. Pair programming also helped bridge the gap between teams in different time zones, fostering stronger collaboration and accountability. Engineers reported improved confidence in both their code and their colleagues, creating a positive feedback loop for organizational resilience.



Pair Programming at Scale - Related Figure.

VII. A/B Testing Modernization: Adobe Test to Optimizely

The shift from Adobe Test to Optimizely fundamentally changed how the company approached experimentation. While Adobe Test had limited the company to basic A/B experiments, Optimizely enabled multivariate testing, full-stack experiments, and feature flagging capabilities. This expanded experimentation beyond the frontend into deeper aspects of the platform, such as pricing algorithms and recommendation engines.

As a result, experimentation became a central part of product development rather than an afterthought. Teams could now make data-driven decisions with statistical rigor, using Optimizely's advanced stats engine to ensure results were valid and actionable. This allowed the company to iterate more quickly on features, test hypotheses in production, and continuously improve the customer experience.

Figure 5. Comparison of Experimentation Platforms

Capability	Adobe Test	Optimizely
UI Experiments	Yes	Yes
Full-Stack Testing	No	Yes
Feature Flags	No	Yes
Multivariate	Limited	Yes
Stats Engine	Basic	Advanced

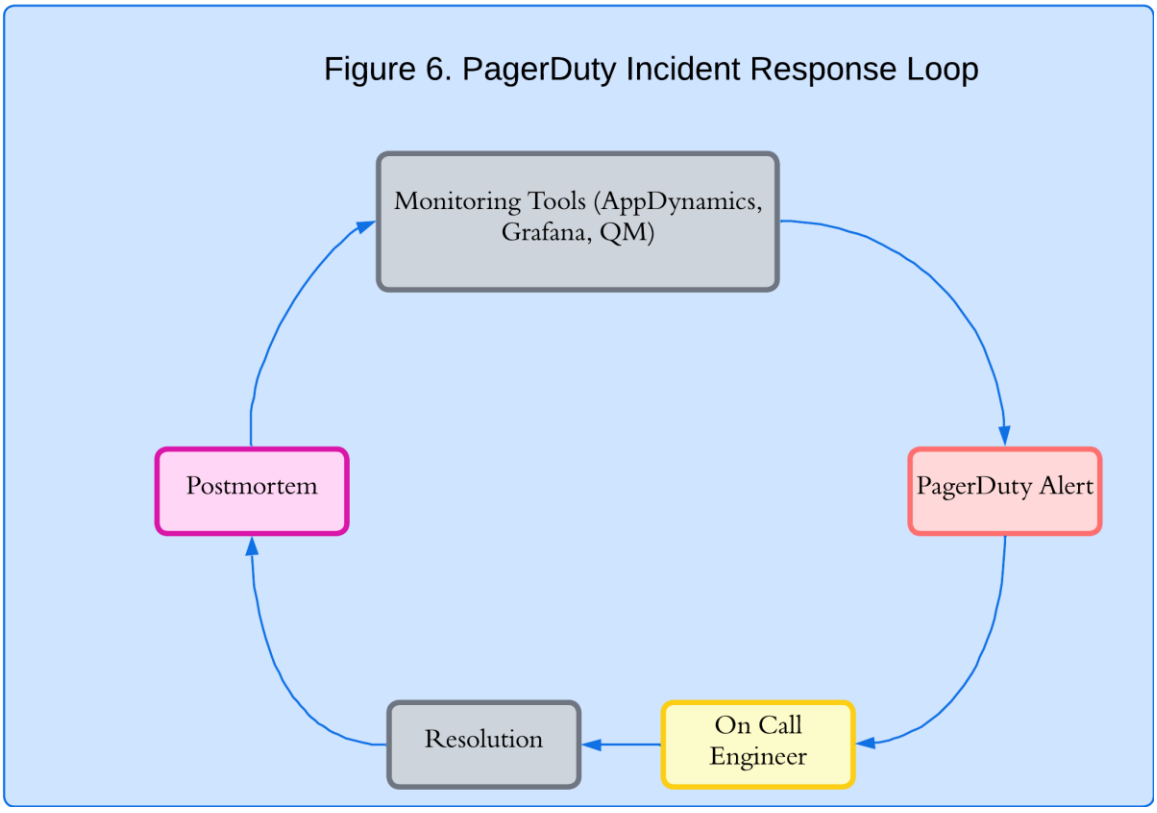
A/B Testing Modernization: Adobe Test to Optimizely - Related Figure.

VIII. Incident Response with PagerDuty

Modernization would not have been complete without improving operational resilience. PagerDuty was integrated into the entertainment company's observability stack, working alongside AppDynamics, Grafana, Splunk, and Quantum Metric to provide automated incident response.

PagerDuty routed alerts to the appropriate engineers based on predefined escalation policies, reducing mean time to acknowledgment. Engineers relied on Grafana dashboards for system health insights and AppDynamics traces for application-level diagnostics, while Quantum Metric provided visibility into customer behavior during incidents. For example, if user drop-offs spiked during checkout, Quantum Metric triggered alerts that fed into PagerDuty, even before infrastructure-level metrics showed degradation.

This combination of tools enabled experience-centric incident detection, which shifted the entertainment company's reliability practices closer to the principles of site reliability engineering. MTTR improved significantly, particularly during high-demand events, protecting both revenue and customer trust.



Incident Response with PagerDuty - Related Figure.

IX. Lessons Learned

- Incremental modernization reduces risk.
- Tooling multiplies impact: Harness and Optimizely improved both productivity and agility.
- Culture is key: Pair programming and PagerDuty enabled modernization success beyond technical improvements.
- Customer Experience Monitoring Is Essential: Tools like Quantum Metric extend observability beyond infrastructure, ensuring modernization aligns with real-world business outcomes.
- Decoupling Content Empowers Non-Engineering Teams: Acquia CMS reduced reliance on developers for content updates, freeing engineering resources to focus on modernization.

X. Conclusion

The modernization of the entertainment company's ticketing platform demonstrates how legacy systems at global scale can evolve into cloud-native, experimentation-driven, and resilient architectures. By migrating to React.js, adopting GCP, implementing Harness, enhancing observability with AppDynamics, Grafana, and Quantum Metric, adopting Optimizely, and leveraging Acquia CMS, the company delivered measurable improvements. Page load times fell by nearly 40 percent, deployment frequency accelerated from bi-weekly to daily, and mean time to recovery decreased by 30 percent.

Perhaps most importantly, modernization was not simply a technical upgrade. It represented a cultural transformation that combined new tools, new processes, and new ways of working. The entertainment company's experience reinforces the principle that modernization succeeds when organizations align technology with culture, ensuring that improvements in velocity and reliability are matched by improvements in collaboration and customer experience.

References

[1]. NIST, "Cloud Computing Standards Roadmap," Special Publication 500-291, National Institute of Standards and Technology, 2013.
[2]. N. Forsgren, J. Humble, G. Kim, Accelerate: The Science of Lean Software and DevOps. IT Revolution, 2018.
[3]. D. Spinellis, Code Quality: The Open Source Perspective. Addison-Wesley, 2006.
[4]. B. Fitzgerald, K.-J. Stol, "Continuous software engineering: A roadmap and agenda," Journal of Systems and Software, vol. 123, pp. 176–189, 2017.
[5]. M. Fowler, Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2018.
[6]. R. Kohavi, A. Deng, B. Frasca, et al., "Online controlled experiments at large scale," Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2013, pp. 1168–1176.
[7]. K. Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 2004.
[8]. B. Beyer, C. Jones, J. Petoff, N. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O'Reilly, 2016.
[9]. Optimizely, "Experimentation Platform Overview." [Online]. Available: https://www.optimizely.com/
[10]. Harness.io, "Continuous Delivery at Scale." [Online]. Available: https://harness.io/
[11]. AppDynamics, "Application Performance Monitoring." [Online]. Available: https://www.appdynamics.com/
[12]. Grafana Labs, "Observability Platform." [Online]. Available: https://grafana.com/
[13]. Quantum Metric, "Continuous Product Design Platform." [Online]. Available: https://www.quantummetric.com/
[14]. Acquia, "Acquia CMS: Open Source Content Management System." [Online]. Available: https://www.acquia.com/

