DevOps for Customer First Strategy

Scale CX (Customer Experience) using the power of DevOps tenets

GlobalLogic®

Contents

Introduction

Both Customer Experience and Devops are no new industry buzzwords and even as they are being mentioned here, they are being deeply discussed in almost all new and existing business transformations. Customer interactions with organizations and their line of business products are becoming progressively digital and expectations are only upwards. This only calls out aloud that an exceptional Customer Experience is paramount.
The competition these days is not only against digital giants of the same product line but also the ones providing customers with unparalleled online experiences. Uber Airbnb, TaskRabbit, and similar on-demand and sharing economy business models have set examples of experiences one can not ignore. The global knowledge of the customers is growing and they are getting accustomed to well-designed, thoughtful digital experiences, hence they do not settle for anything less in any of their experiences or any of their product or service purchases.

Organizations have a need to fulfill an unprecedented demand for digital services, and delivering services in silos is not optimum. Just like behind good software a functional, performant, secure, reliable, and highly available software design is required, a successful customer experience has to be thought of exactly the same way in terms of its experience design. In order to deliver highly differentiated experiences, it's crucial that the design be elevated, prioritized, and scalable.
An organization needs to take steps to elevate its experience design practice to provide exceptional customer experiences. There are two important elements to provide a joined-up experience for the customer. First is developing great software to bring in the digital transformation and the other is to operationalize the digitized transformation which is called DevOps.

In this document, we will see the latter element of how an organization can take steps to elevate its DevOps practice to provide exceptional customer experiences making a vital shift in the digital age.

Make DevOps part of Organizational identity

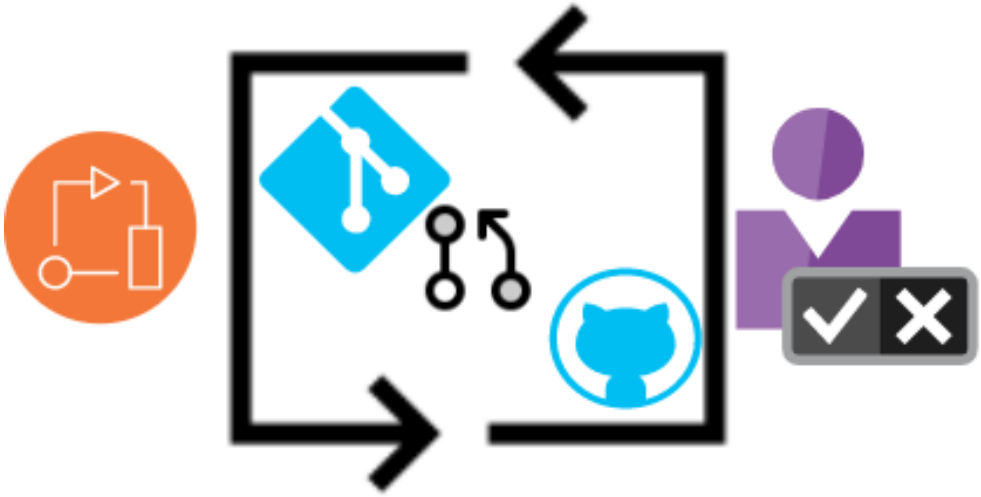
DevOps is an organizational culture shift that emphasizes continuous delivery - a focus on everybody working together to improve development performance measures such as **throughput** while at the same time increasing *stability* and reducing *mean time* to restore service.

DevOps is meant to engage our customers' time to value. A good DevOps means an accelerated customer experience. A better Devops means an assured customer experience. To bring in this assurance, new or improved concepts, technologies and frameworks under DevOps need to be adopted.

- GitOps
- DORA
- MultiVaried Testing
- Break AntiPatterns

GitOps

Gitops, to begin with, is DevOps and Gitops has been introduced to simply make DevOps better. GitOps enhances DevOps by incorporating Git throughout the software delivery process, making it easier to orchestrate projects and keep them in sync.
Gitops is a good step towards standardization as well. It is an evolution of DevOps, especially the IaC workflow where the traditional ***push*** approach of detecting changes is recommended to be replaced by the ***pull*** approach. The GitOps workflow hence improves the development pipeline's productivity and velocity and in turn the system's reliability index.



Gitops directly impacts the customer experience by :

- Decreasing operational overheads by providing a self-service large cluster management platform for the developers to innovate and deploy software all the way to production without a full need from the platform team.
- Providing Consistent and Reproducible clusters reconcile the current state of any of the clusters from multiple Git repositories.
- Providing Hybrid and Multi-Cloud Strategies, especially Kubernetes. Like any technology, Kubernetes gets more complex when it has to be to run across multiple clouds. GitOps is designed to centralize and automate much of the operational work that goes into supporting clusters across clouds. It enables holding everything – not just application code but also ops, policy, and config information in version control.
- Working with service mesh technology where a single Git commit can control customized workflows by taking advantage of the service mesh's ability to *measure* and *manipulate* traffic anywhere in the application's call graph. Together they deliver far more than the sum of their parts.
- Providing Robust Network Security and Observability when combined with Cilium for example as a networking, observability, and security solution.

The end goal achieved is a smoother, faster, and more reliable software development and delivery thus enhancing the customer experience.

DORA

DORA stands for **DevOps Research and Assessment.**

With digital disruption for most industries software has become key to market differentiation. To react more quickly to the continually changing needs of customers, companies have had to become software companies. As a result, the importance of the internal software development organization has only increased. Creating a level of flexibility that enables the team to react, even on short notice, to the changing needs of customers is a central component of DevOps culture. Flexibility is being built-in into the way they work but yet far from 100%. Just as many indicate that they have CI/CD pipelines in place, a basic DevOps requirement, with which development teams can automate and test changes to their code.
Gartner notes that many DevOps initiatives fail because expectations for these initiatives have not been clearly defined within the enterprise. In order to clarify and agree on these expectations, experts recommend that IT and the business reach an understanding of common goals and metrics. The below questions arise -

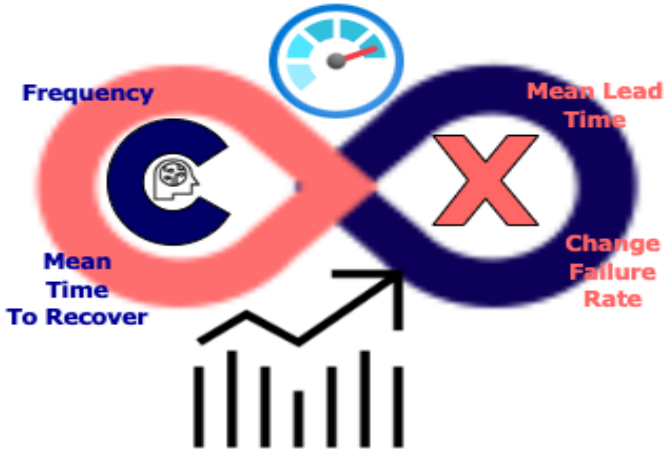
- ➢ What metrics do software development teams actually track and use?
- ➢ What metrics are being used to evaluate software development?
- ➢ What metrics measure the value of software for the customer?

Possible answers to capture the above metrics are

- ☑ Open support tickets. However, while this metric is easily accessible, it does not always produce a lot of useful insight.
- ☑ Feature Adoption, Churn, Return on Investment, or Net Promoter Score
- ☑ Monthly number of active users. However, this number might not denote any real relationship between the software delivered and its value to customers.

Even when operating at different levels of **DevOps maturity**, development teams in general have hardly any insight into how customers benefit from their work, and few are able to discuss these benefits with the business. But having such insights ready at hand would improve collaboration between IT and the business. The more customer value metrics a development team tracks, the more positive that team views their working relationship with the business. This would be a good spot to recall the words of the Agile Manifesto, which itself serves as the foundation for all DevOps initiatives. The first Agile principle reads as: **“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”** Without knowing whether the intended value for the customer is being achieved or not, development teams are effective in driving with blind spots.

DORA metrics come to great rescue providing objective data to measure the performance of software delivery teams and drive product improvement.



Deployment Frequency: The frequency of deployments in a production environment is relatively easy to track, but teams need to define, for example, what they mean by “daily deployments” based on the number of days in the workweek. Actual deployment frequency will vary according to the requirements of different teams and services.

Lead Time for Changes: This measures the amount of time it takes for a committed change request to go into production. Ideally, each step in the development and deployment process will be time-stamped in order to identify areas for improvement.

Mean Time to Recovery (MTTR): This measures the amount of time it takes to restore service after a failure.

Change Failure Rate: This metric captures the percentage of deployments that result in different types of failure (downtime, rollbacks, poor performance, etc.).

Based on our many large-scale digital transformation customer-oriented experiences, what do we, at GlobalLogic, think is a good strategy to implement the DORA?

- ☑ Track DORA metrics improvement on at least a *monthly* frequency if not lesser.
- ☑ Contextualize the metrics to give them sufficient meaning for all stakeholders.
- ☑ Automate the feedback cycle on improvement and enhance the developer experience.
- ☑ Benchmark the cycle time of Coding, Review Feedback, Feedback work, and Deployment.

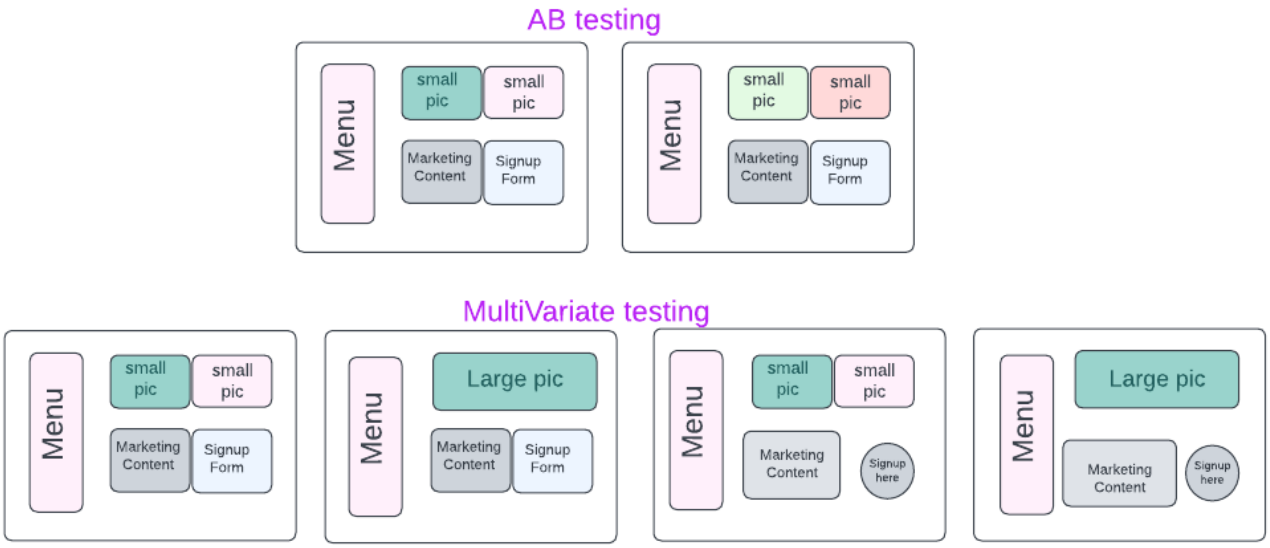
- ☑ Correlate data across different DevOps tools to identify bottlenecks and automate developer workflow optimization.

MultiVariate Testing

MultiVariate is about making data-driven decisions. But what happens if one wants to focus on something a little less tangible, like customer satisfaction? What is a satisfied customer? The satisfaction can possibly be linked to some concrete behaviors like purchases or subscriptions. But as anyone who has shopped online knows that one can have a great experience without converting. Maybe the visitor to the site is researching before buying, or maybe she enjoys browsing until she is ready to subscribe. These customer behaviors are still positive and worth fostering. But without direct metrics, are you left to just hope that your customers are satisfied? The answer is a resounding no. There are strategies you can employ and metrics you can use that will help you gauge and improve customer satisfaction as part of your multivariate testing.

Implementing multivariate tests to improve customer satisfaction by focusing on metrics such as return visits and calls to customer service, and by introducing feedback points throughout the customer journey.

Multivariate testing uses the same core mechanism as A/B testing which we have all heard of and most have also practiced. However, the multivariate test compares a higher number of variables than the A/B test and reveals more insights about how these variables interact with one another. The purpose of a multivariate test, then, is to measure the effectiveness of each design combination at a granular level.



AB test is usually comparing two versions with mostly the same business elements. For example, it can be to test a more appealing color or font change. Whereas multivariate tests provide a strategy to test customer-centric ideas.

Recently one of our e-commerce customers conducted a test designed to measure how their checkout process would impact the subscription enrollment experience. This retail brand specializes in pet care, so its product requires a level of engagement, payment assurance, and product volume. They used to have relatively high declined card errors in their checkout process during payment capture, but after revamping the authorization of cards while capturing card details they saw post-purchase customer service calls decrease by 30%. Without any user experience change they ended up improving the customer experience significantly, in a quantifiable way.

Similarly one may want to measure purchases, return visits, and order value or revenue per visit among users in each variation. If return visits go up while purchase-related metrics remain stable or also increase, then it can be inferred that your customers are satisfied with their experience. If a drop is seen in return visits or a drop in purchases or revenue then there is a high possibility that your customers are looking for something that is unavailable. There is an opportunity for more business here.

Break Antipatterns

DevOps organizational structure, Deployment architecture, Automation, and Testing responsibilities influence the effectiveness of DevOps execution. DevOps requires a full analysis of the value chain connecting the needs of the customers and business stakeholders to the needs of the operations stakeholders. However, there are practices that are anti-pattern to the above-mentioned fundamental ways of DevOps working and should be discouraged to follow.

Silo Devops Team

Under the tag of Devops, enterprises create a specific team for creating pipelines, advising on source code branching strategies, infrastructure, security, release deployment, and similar operations. This team is disconnected from the team of developers resulting in communication gaps, mismatched expectations, and build or deployment failures disturbing the agile methodology.

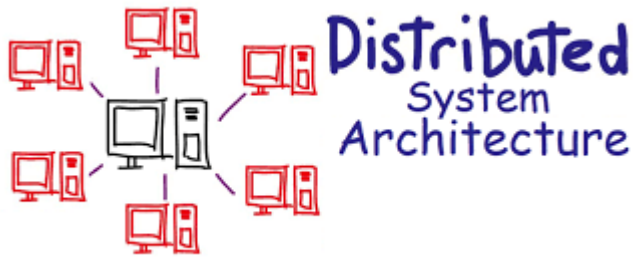
Recommended Solution:

A coherent organizational structure for software development and delivery should be created. While the responsibility is not shared across the teams, the accountability needs to be shared for smooth functioning and making agile delivery a success. Developers, testers, and operations should sufficiently overlap.

Deploying Undistributed Architecture

DevOps principle calls out that a new deployment to a live environment should be such that it does not break the existing business functionality. If not followed, it can hamper the customer experience.

Recommended Solution:

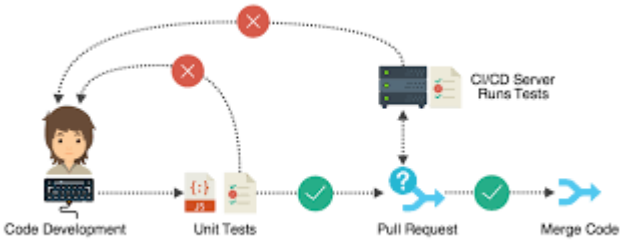


The solution to this is to break down the software architecture preferably in a domain-driven approach or a loosely coupled approach. The business functionality should be abstracted in such a way that only the interfaces for the components to communicate with each other are visible. This makes deployment components granular and the business impact is at a much lower risk.

Testing Afterward

Even when DevOps and Agile methodologies are claimed to be practiced by any project team, good testing is still followed as an afterward step of development by a dedicated team. Unit tests and effective code coverage are seldom mandated by organizations and are left to the will of the project manager. Not implementing unit tests correctly or completely and leaving for the system testing to report bugs eventually messes up the integration and the release timelines.

Recommended Solution:



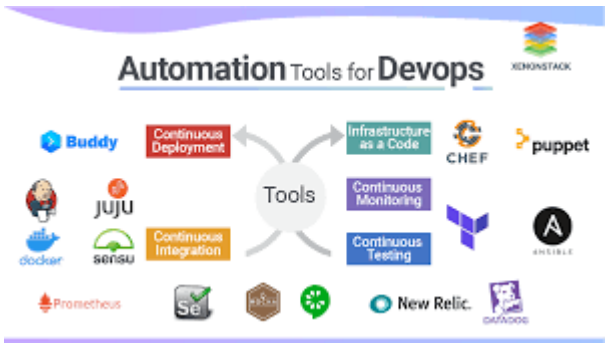
A test-first approach and a continuous pattern is the answer. Firstly, A shift-left approach should be diligently introduced by enforcing the unit tests in the development lifecycle.

Secondly, The testing team should not work in silos just like the earlier stated antipattern, and should go onboard test engineers onto the development team.

Delayed Automation

Automation is often left to the priority of the project owner. It occurs that functions are excluded from automation because in order to meet business deadlines or for inexplicable reasons of not having enough skills. This can impair the process and only creates more backlog and always remain in the noncritical bucket because of the notion that automation does not bring a business ROI. However, this same lack of automation hinders the generation of long-term organization ROI.

Recommended Solution:



To get the most value out of automation specific criteria should be defined and then the priority of the automation tasks should be listed. These criteria could be based on what is most repetitive, the riskiest part of the process, where automation can create the highest ROI, or what takes the most time. Once the criteria are set it should be planned in the sprints without fail.

Conclusion

Customer experience is a blend of many aspects for a successful digital transformation. It involves customer behavior and response during the pre-purchase, consumption, and post-purchase stages. By intertwining the disciplines of software delivery and DevOps tenets, Customer experience only stands to move north. The quality and effectiveness of their products, services, and processes are a constant measure of upkeep. DevOps can serve the

needs of digital CX by enabling enterprises to deliver innovations at high velocity. At the same time, by bringing in CX as the prime or additional “quality” metric into DevOps, teams would help to create Customer first culture in the entire IT organization that is also just as imperative.

Author

Surbhi Nijhara, Principal Architect, GlobalLogic

surbhi.nijhara@globallogic.com