Winning the Customer Experience Game:

Leveling Up with Progressive Delivery and Observability within the Mobile Gaming Application Lifecycle

An Intellyx Whitepaper for SmartBear
by Jason English, Principal Analyst
April 2022

A friend once told me life is a business. But I never bought into that definition.

Having played games all my life, I suggest approaching *life as a game*.

Just like life, in a game you try alternate strategies, take chances, and learn how to avoid risks to survive to later levels, where you can continually progress and improve.

[Only, there's not always a 'continue button' in life when things get unstable …
Sorry – I'll leave those comparisons alone from here.]

Seriously though, the simulated world of gaming does provide an excellent window into how any organization should view their customer's real-life experience with a software product. The survival of the digital business behind producing a game depends heavily upon the quality of the user experience and the stability of its software in front of customers.

This paper will explore how progressive software delivery and observability practices can help digital organizations deliver excellent customer experiences at a massive scale, using the high volume and highly competitive mobile gaming arena as an extreme example.

How has the game changed?

Video games surpassed movies, music and television as the highest grossing form of entertainment, valued at US$173.7B in 2020 with an anticipated growth rate near 10% annually.

We're even watching other people play video games instead of other major sports, with eSports viewership in the US now second only to NFL Football. Gaming is now culturally embedded in our consciousness, and we can't get enough of it. We want newer challenges, ease of use, better graphics, smoother gameplay, and we'll keep trying new games to find better experiences.

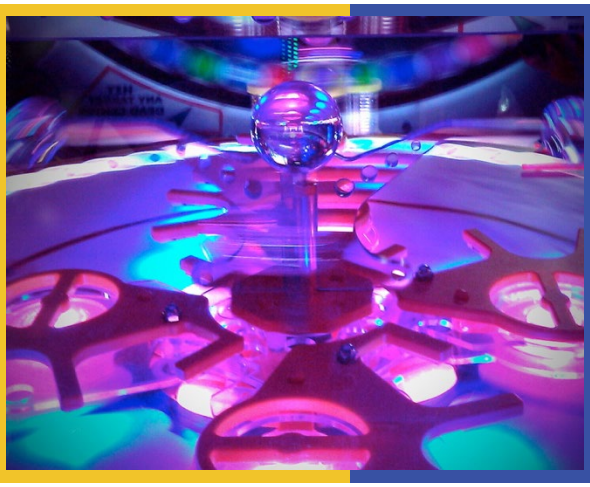
The ubiquity of smartphones today (roughly 6.6 billion devices) puts a powerful gaming console in the hands of almost everyone in the world. The recurring income from in-game purchases and ad traffic generated by mobile games installed on smartphones is rapidly outgrowing that of console and desktop games.

Current trends such as remote work and social distancing only increased the state of play, with a report from mobile carrier Verizon noting a 75% week-over-week increase in

gaming usage at the start of the pandemic, with 858 million gaming apps downloaded in one week.

Mobile games call for a thoroughly modern paradigm of distributed application development. The most popular and successful titles involve dimensions of multi-player gaming, social networking and community elements. This means almost every user action and status displayed in the front-end user interface of the local app actually happens through API calls to the game producer's cloud, or other service-enabled back-ends.

Round trips between the end user's device and cloud-based resources represent the biggest experiential challenge of mobile game development, one shared by teams delivering many other forms of applications.



Any function of a service-based application can only perform as well as the least performant API calls and data handoffs between local front-ends and cloud-based back-ends.

Whether you are delivering mobile games, or virtually any other form of customer-facing application that depends on services, there is only one imperative. Fail to deliver a quality user experience, and customers will leave, faster than they arrived.

Saving progress: Staying alive with progressive delivery

A huge global market for gaming creates fierce competition for players, driving wild successes, tragic failures, and frequent consolidation among mobile gaming firms.

The flow of money in this space depends upon delivering novel and fascinating experiences to users. Software engineers are under pressure to release new games – and put new features in games – faster than ever.

Over the last two decades, development teams have accelerated software delivery by adopting better collaboration and automation practices, such as agile methodologies, test-driven development (TDD), continuous delivery and DevOps. These strategies encouraged teams to work closely together to chop epic projects into bite-sized features that they could test and deliver in much quicker cycles.

Now that such practices are commonplace across teams in most industries, they are just table stakes for the studio entering an app into a marketplace. There's still more timeline to squeeze out of software updates.

Teams can shift-left functional and regression testing to code check-in, and even run earlier performance tests, by having continuous integration pipelines provision staging environments for automated testing to run against. While shift-left testing can generate an additional 20-30% cycle time boost upon its introduction, there's still an important speed limit: users.

Good old manual user acceptance testing (UAT) never goes away, no matter what practices and automated technology are brought to bear. Whenever a new application feature is introduced at scale, we still need to understand the real end user's experience. When gamers encounter errors, slowdowns and service interruptions, we better learn from them, or the competition is just a swipe away.

"Game over, man. Game over! Hey, maybe you haven't been keeping up on current events, but we just got our asses kicked, pal!"

- *Bill Paxton as Private Hudson, in Aliens [1986]*

Progressive delivery is the next chapter in the evolution of agile delivery processes. This practice involves producing highly granular, iterative features at each release time, and gathering feedback about the impact of changes ***near or after deployment***. Progressive delivery could circumvent much of the time needed for UAT.

But wait, to anyone who cares about quality, this sounds scary and dangerous. How is this even possible? Isn't that just trial-and-error testing in production? Wouldn't instability in front of customers make them go away?

Gamers want a stable user experience, but they are also very familiar with the concept of iteration. Incremental experimentation is a basic component of developing a player strategy.

Compared to the risk-averse users of say, a financial or medical app, most gaming users would rather see new features and new levels delivered faster, so long as any resulting glitches aren't too inconvenient or long lasting.

By reducing the scope of changes to tighter increments, risk is limited, and by focusing on the tradeoffs between application delivery speed and stability, progressive practices can prioritize a middle way forward on this journey.

Playing a role in a massively multiplayer environment

Multiplayer games provide ideal edge-case goals for the rest of us in spades, even if our own team goals are simply producing mundane business apps.

- **You want scale?** Some role-playing and puzzle games have well over 100 million downloads and several million active gamer profiles.
- **You want performance?** Millions of eSports players measure their own game performance by the APS (actions per second) metric, meaning smooth frame rates and sub-second feedback between user and server is essential.
- **You want stability?** Every click of every player in a massively multiplayer game creates a digital record, and even in a non-action game, gamers expect to maintain their state of play with freedom from disruption.

Replacing requirements with objectives

In the world of enterprise applications, we are all too familiar with the concept of service level agreements (or, SLAS). SLAs represent obligations between the software vendor and its business customer, for targets like 99.995% availability, or less than 2 second response times to a query.

Fail to meet the SLA, and the vendor might get notified. Fail to remedy the situation in the agreed upon timeframe, and fines or breach of contract terms could be triggered. This particular approach to assuring performance is rather outmoded – not just because it isn't motivational for development teams, but because it is focused on past failures instead of continuous improvement.

In a mobile game scenario, there's no SLA between the game producer and consumer. Players are free to leave if ads are hanging up the game, or if they aren't having a good time. Instead, game producers started focusing on SLOs, or service level objective targets for continuously improving customer experience (CX).

Much of CX is still about 'speeds and feeds' associated with application performance monitoring (APM). Network latency, compute time and data query/response roundtrips are still golden signals worth measuring.

Observability tools picked up where APM left off, correlating these metrics alongside deeper system-level logs and traces to provide transparency into the inner workings of applications. As applications moved to cloud infrastructure and microservice-based architectures, observability allowed engineers to investigate the root causes of production problems.

The gaming industry still needed the forward-looking means to improve CX, as the introduction of new game features could inevitably expose unknown problems. Perhaps if there were a way to lay breadcrumbs like Hansel and Gretel, so we could retrace the steps of our characters, when they are lost in the forest?

The golden signals of game app performance

The "performance" of an app release in the mobile gaming world also measures several results most software engineers don't normally consider to be part of performance testing.

- **Active user sessions** – Fewer live players than normal could signify an access problem, whereas too many players could signal an upcoming scaling crisis.
- **Crashes** – There will always be local app crashes and hangups happening somewhere in the world, especially within a heavily used, global multiplayer game ecosystem. Are there more error events than normal, and if so, can the team proactively do anything about them?
- **Lag time** – This metric measures the time it takes requests to make the round trip between the app and the data source. Latency is part of lag, but mobile games also measure load times and on-screen frame rates that gamers experience as a lagging game.
- **Economics** – Are users spending money on in-app purchases, and using the virtual currency within the game regularly? Are ad networks functioning properly and delivering the expected bonuses to players? Prioritizing the release and performance of in-demand features makes a direct impact on revenue.

Enterprise use cases could take a hint from these golden metrics of game stability in their own performance engineering practices. In an ultra-high traffic cloud application with near-constant releases, it is critical to reduce all the data coming back to focus on only the most relevant logs and metrics that impact customer experience.

The endgame: progressively improving customer experience

"It's good for a lot of graphs to go up and to the right, but you don't want to see your crash rate going up and to the right … looking at just 2000 users so far, there's a good chance there's an issue with this particular release."

– Leo Schnee, Software Engineering Manager, Zynga, in a Bugsnag webinar, 2021

Zynga brought many of the world's most successful multi-player mobile game titles to market, such as *Words with Friends* and *Monsters & Puzzles*. Their *Hit It Rich!* slot game has more than 500,000 reviews on Google Play and the Apple Store and maintains an astounding 4.5-star rating.

The *Hit It Rich!* engineering team is continuously developing new features and updates for the game, and progressively rolling them out, usually starting with less than 5 percent of their active user base.

Agents installed when configuring Zynga's pre-production and production deployments stream metric data, including the golden gaming UX signals within the user sessions. When a crash or error is detected, SmartBear's Bugsnag stability monitoring platform will report an event for further examination within observability and testing platforms, or an issue in Jira, or an email or Slack message to the feature owner.

The goal of combining production monitoring with progressive delivery techniques is to proactively identify and resolve problematic user journeys, by actively observing an increasing subset of real users. It can also guide product design, allowing high value features to be offered and optimized faster for VIP customers such as premium members or high-status players.

There are several forms of testing and test enablement that are often included in the progressive delivery arsenal:

- **A/B testing.** By deploying two versions of code, engineers can use a platform like Bugsnag to compare performance metrics between alternative functional releases in the game, or between a version 1 and v1.1 update.

- **Canary** or **blue/green testing** refers to the process of incrementally increasing app traffic flow to an increasing subset of deployment infrastructure and users. This gradual rollout limits the blast radius of failures, allowing quick rollback to reduce further impact on users if an error occurs.
- **Feature flagging** functionality within many DevOps platforms and pipelines assumes that new code will be delivered alongside existing code, modularly packaged as features. Features can be switched on for validation in production, and quickly switched off if monitoring tools are reporting errors.
- **Synthetic monitoring** can be used to generate or reproduce real user traffic patterns against test and staging infrastructure, so progressive testing can 'shift-left' to pre-production infrastructure and observe it under production-like conditions.
- **Production monitoring** gathers live interactions, errors and analytics data streaming from actual end users in production. Developers gain deeper insights into crash conditions and problematic performance trends, so they can spot and act upon them faster.

Fortunately, an involved community of vendors and open source projects are involved in this journey, helping to advance the art of progressive delivery and progressive testing that takes into account the real-life experience of customers.

The Intellyx Take

Whether or not life is a game, the game is the same – whether your team is delivering a mobile game, or any app that will be put in front of customers.

We want to deliver more high-value features within apps, faster, while ensuring higher stability – with fewer severe incidents and faster remediation of defects.

Ensuring a stable app experience means much more than simply capturing bugs. Continuous monitoring of apps can inform progressive delivery practices, allowing developers to experiment with new features and release them incrementally for faster real-world feedback with less risk.

In today's hyper-competitive market, you can't cheat on quality to win the game – but on the level where application stability meets progressive delivery, smart teams will save progress and respawn to safely release features that will delight customers for another day.

About the Author

Jason "JE" English (@bluefug) is Principal Analyst and CMO at Intellyx, a boutique analyst firm covering digital transformation. His writing is focused on how agile collaboration between customers, partners and employees can accelerate innovation.

In addition to several leadership roles in gaming, supply chain, interactive and cloud computing companies, Jason led marketing efforts for the development, testing and virtualization software company ITKO, from its bootstrap startup days, through a successful acquisition by CA in 2011. JE co-authored the book *Service Virtualization: Reality is Overrated* to capture the then-novel practice of test environment simulation for Agile development.

About SmartBear

At **SmartBear**, we focus on your one priority that never changes: quality. We know delivering quality software over and over is complicated. So our tools are built to streamline your DevOps processes while seamlessly working with the products you use – and will use. Whether it's TestComplete, Swagger, ReadyAPI, Cucumber, Zephyr, Bugsnag, or one of our other tools, we span from test automation, API lifecycle, collaboration, performance testing, test management, app stability and error monitoring, and more. Whichever you need, they're easy to try, easy to buy, and easy to integrate. We're used by 16 million developers, testers, and operations engineers at 32,000+ organizations – including world-renowned innovators like Adobe, JetBlue, FedEx, and Microsoft. Wherever you're going, we'll help you get there. Learn more at smartbear.com, or follow us on LinkedIn, Twitter, or Facebook.