EBOOK

Test & Learn

The Ultimate Guide to Delivering High-Quality, High-Impact Products

Optimizely

Contents

The background

This year, the Progressive Delivery and Experimentation Summit, brought together nearly 1500 professionals. Leaders from Product, Engineering, and Data presented frameworks for adapting software development and delivery practices. Attendees learned how to use feature flagging and experimentation to innovate with confidence.

Software delivery

Every product team wants to move quickly. Thanks to Agile, DevOps, and CI/CD, organizations are moving faster than ever before. But speed alone is not enough, and customer expectations are higher than ever. Customer journeys traverse multiple devices and touchpoints, and complex platform architectures make each release riskier and harder to test.

While speed and agility are essential, they don't guarantee that you are building the right thing. They can't validate quality. Feature Flagging and experimentation reduce uncertainty, allowing teams to move fast and fearlessly, developing the features they know users love.

What exactly do we mean when we say Test & Learn?

For many teams, product development still takes the form of something like **Design > Build > Launch > Pray**.

A Test & Learn approach empowers teams to challenge assumptions. It is a way to quantify customer impact early and often, using feature flags and A/B tests for product validation. Test and Learn transforms the delivery model to **Design > Build > Experiment > Iterate**.

Test & Learn is about safely accelerating product development to deliver high-quality software that users love.

- ✓ Using data and experimentation to drive products to greatness.
- ✓ Adopting development practices that bring products to life quickly and with fewer risks.
- ✓ Driving higher customer engagement and retention by building positive user habits.

Deliver better products with feature flags and experimentation

This is where progressive delivery and experimentation come into play, forging the link between speed and safety. Leading teams are now adopting feature flags for gradual rollouts, remote configuration, and A/B testing to drive the entire Test & Learn process.

Start shipping confidently with Rollouts—Optimizely's free feature flagging and experimentation solution.

Sign up for free

The summit speakers maximize the quality and impact of every release by quickly and safely validating features and code through the entire software development lifecycle.

Let's see how they do it.

01

Test & Learn to deploy fast and fearlessly

A Test & Learn approach to product development means continuously questioning, experimenting and measuring impact before rolling changes out to every user.

It combines feature flagging and experimentation to gradually roll out high-quality, high-impact products faster, more iteratively, and with less risk.

Nicole Forsgren, DevOps expert and VP of Strategy and Research at Github, believes that using innovation to go faster than your competition is all about testing new ways of doing things. According to Forsgren, no fully automated, out-of-the-box solution will give you a competitive advantage. In fact, she believes that if you rely on technology alone, then something is wrong.

Say you buy a server. Well, your competitors can do the same, and you don't achieve any competitive advantage. Everyone has gotten faster in the same way. That's why, to deliver higher quality software and deliver it faster, the best performing teams align strong technology solutions with the right processes and, just as importantly, the right culture. Only then can you get better in ways that no one else can.

"Whether you are a summer intern or the CTO, any good idea must be objectively tested, preferably, with real customers. Everyone must be able to experiment, learn, and iterate. For innovation to flourish, measurement must rule."

Nicole Forsgren
VP of Strategy and Research / Github

Accelerate by automating the simple things

Forsgren advocates avoiding heavyweight change management practices, which can slow you down and introduce instability. Instead, automating low-risk approvals lets you maintain stability and continue building the right products. Even if certain testing needs to remain manual, this doesn't have to slow down how frequently you integrate and deploy code.

Progressive delivery, feature flags, and experimentation

Feature flags empower teams to take code ownership, starting with your developers.

"Feature flags are the key to managing risk and delighting users. They are the first steps in progressive delivery, allowing you to leverage A/B testing, blue-green deployments, and gradual rollouts. By rolling out to a small subset of users, verifying, then gradually rolling out to more, we can also do a safe and quick rollback."

Nicole Forsgren
VP of Strategy and Research / Github

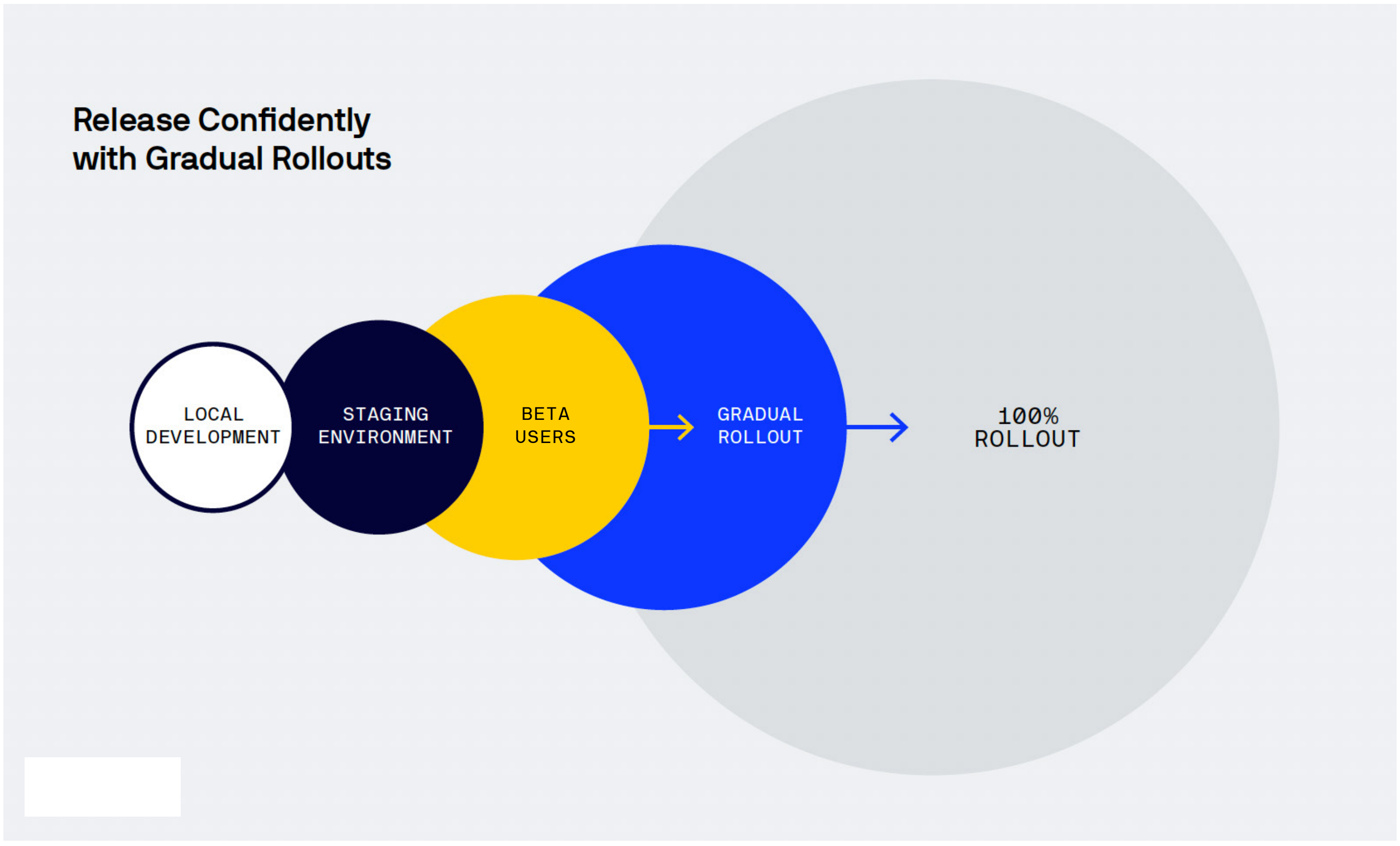
What is Progressive Delivery?

Progressive Delivery represents the latest evolution in software release after Agile and, more recently, Continuous Delivery. Progressive Delivery goes further to include more granular feature rollouts, canarying, A/B testing, and observability, allowing teams to work more effectively while reducing risk.

Progressive delivery validates quality and performance in production with feature flags, targeting specific audiences through multiple stages of release: internal users only, internal + Beta, then dialing up traffic 5, 10, 25, 50%, and up to 100% of users.

Experimentation is a natural extension of progressive delivery. It allows you to validate new features and changes as A/B tests, iterate on functionality and UX, and confirm you are building the right things in the right way.

At RedMonk, the developer-focused analyst firm, James Governor invented the term Progressive Delivery to capture a wide range of new software development practices beyond continuous delivery.



Testing in production

Feature flags mitigate the risk of obtaining the real-world data you need to Test & Learn effectively. Ellen Chisa, CEO and founder of Dark, is a strong advocate of programming and testing in production. According to Chisa, developing code in real-world environments helps engineers build and iterate faster with real user data.

Top product teams ensure they are moving fast, and in the right direction, by incorporating real-time user feedback into the entire development lifecycle. Embedding feature flagging and experimentation into their tech stacks makes the build and release processes faster and smarter. They can safely manage code quality with feature flags and identify what works and what doesn't work with real users. Their product analytics are enhanced with experiment data to understand long term impact.

Data is the lifeblood of the HelloFresh approach to product development. Combined with the firm belief that what they measure is what they get, data provides the evidence they need to avoid making decisions on gut feeling. This is also how they avoid being pressured into following the hippos' opinions, the people in the room with the strongest opinion because they have the highest seniority or have been in the company for a long time.

"Since we're in a production system, we have real data to work with and can see what's going on. Seeing a full trace makes it really easy to debug logic when it's going wrong. It also makes it simple to write it with fewer errors the first time."

Ellen Chisa
CEO & Founder / Dark

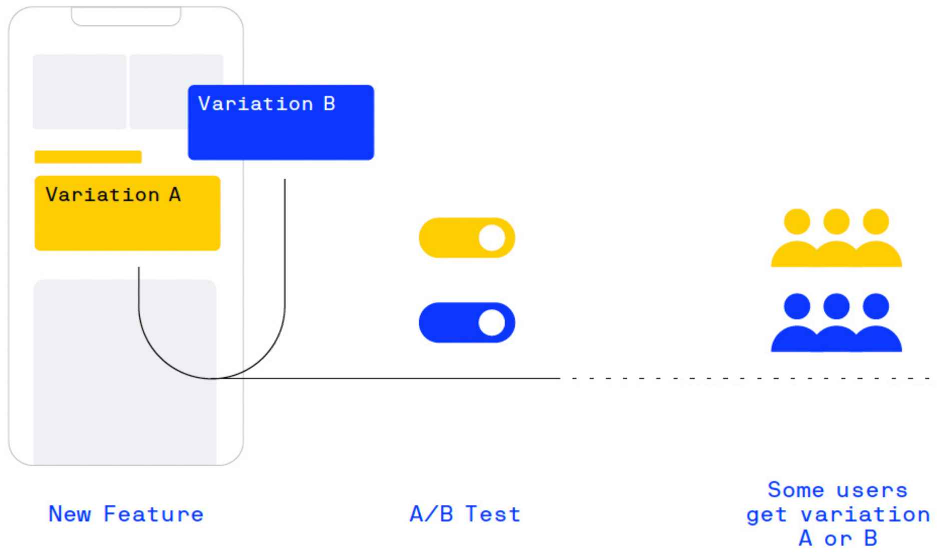
"Data is our vehicle to learn and to make decisions based on fact rather than gut-feeling. And this is how experimentation became so deeply ingrained into our culture."

Aleksandra Vangelova
Product Manager / HelloFresh

Feature flags: The ultimate insurance plan for your products

For many teams, however, testing and coding in production environments is a counterintuitive and stressful concept. That's where adopting feature flag-driven development comes in, giving product teams full control over rolling out and rolling back in production for safe testing, iteration, and experimentation—independent of code deployments.

According to Asa Schachar and Justina Nguyen from Optimizely's Developer Relations team, the ultimate fear is where an action you take risks bringing down an application in production. Think of feature flag-driven development as the ultimate insurance plan for your products.



We're not just writing some check in logic and hoping it works.
We're able to see the full incoming value.

Ellen Chisa
CEO & Founder / Dark

The essential role of feature flags

Feature flags allow you to put a new feature into production but hide it from end-users. Only the team working on it can see the feature and validate that it works as expected. You can then start releasing that feature to specific groups or percentages of your users.

If you detect a bug, the flag acts as an off switch to turn off that feature without a full rollback and without affecting any other work in progress. That saves a lot of time and rework, allowing developers to move faster and confidently and push feature code into the master line early without creating conflicts.

7 reasons to use Feature Flags

Risk Management

Feature flags allow development teams to mitigate the risk of testing new features in production. If a bug is discovered in a new feature, a kill switch lets you roll back instantly without touching your source code.

Canary Releases

Feature flags allow teams to test a new feature on a subgroup of end-users to see how it performs before rolling it out to a broader audience.

Faster Release Cycles

Feature flags make it possible to modify a system's behavior without making disruptive code changes to the live code. Feature flags decouple feature lifecycle management from code deployment, freeing up the engineering team to work on other tasks.

Simplicity

Enabling or disabling features using feature flags is as easy as switching it on or off from a dashboard. No more messy redeployments.

Server-side A/B Testing

One way to implement an A/B test with feature flags is by enabling a feature for half of a segment of users. The feature is disabled for the other half, your control group, to see how the two perform against each other for a certain metric. Feature flags also allow product managers and other non-engineers to A/B test various features within products or systems with no need for additional code deployments.

Feature Gating

Use feature flags to implement targeted rollouts of features to a specific subset of your users.

Continuous Deployment

Feature flags allow companies to perform gradual feature rollouts. Anyone on the team can disable buggy or poorly performing features with the flip of a switch. Developers can streamline development cycles, and roll code backs more easily.

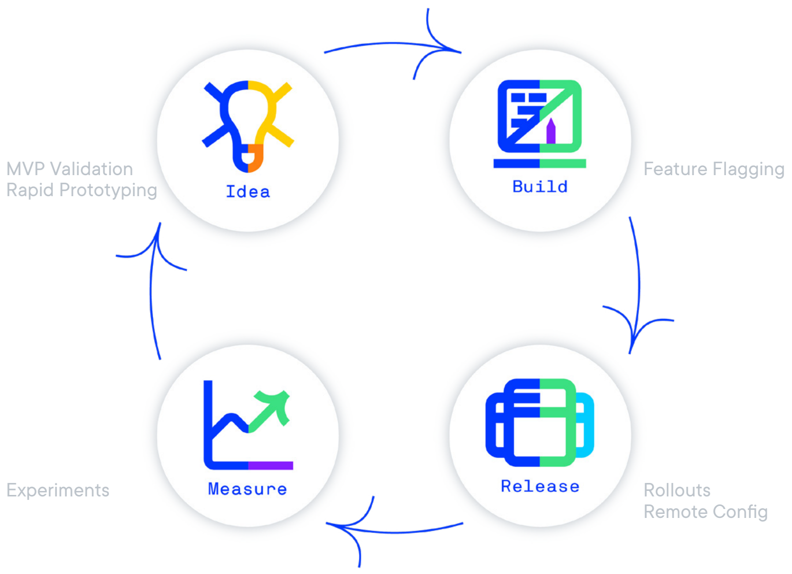
Take 'big swings' with experimentation

Oji Uduzue, VP of Product at Calendly, believes that fearlessness is the key to a successful Test & Learn approach. It's about taking 'big swings' with your product experiments to make an impact. But 'big swings' does not mean 'wild swings.' For Uduzue, innovation is grounded in user needs, motivations, and optimal workflows.

This is where we see experimentation come into the equation as the perfect complement to progressive delivery. Product leaders talk about testing as the other side of the coin when it comes to moving fast and fearlessly. Experimentation completes the circle as the only way a platform can serve your entire product development team's needs.

Product experimentation cycle

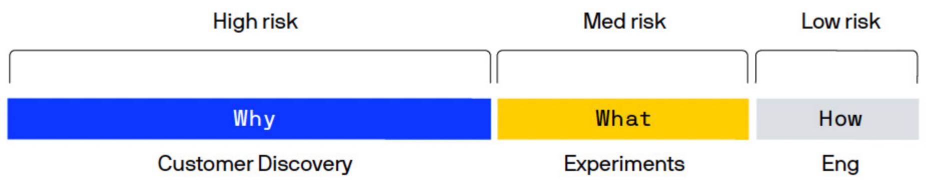
Together, feature flagging and experimentation power Test & Learn, helping teams put customers front and center and iterate quickly at every stage: rapid prototyping in the ideas phase, risk-free rollouts in the build and release phase, optimization and personalization in the Launch and Measurement phase. All supported by real-time results and analysis to make sure your company is making the best decisions possible to continually offer the best customer experience.



"Experimenting is about dealing with risk and figuring out why we are doing what we're doing. How does a user behave? What are the problems? And then we design a solution. That's where experimentation comes in. You do experiments, you make better product decisions, and then you build from there."

Oji Uduzue
VP of Product / Calendly

You do experiments, you make better product decisions, and then you build from there.



HelloFresh leverages experimentation alongside feature flagging to deliver data-driven products to customers around the world.

Don’t fear failure

One of the critical aspects of experimentation is the acceptance that failure is a good source of learning. Luis Trindade, a Senior Product Owner at the ecommerce marketplace, Farfetch, is a proponent for celebrating failures when A/B tests yield unfavorable results. After all, when a feature doesn’t work, you don’t have to release it.

Experimental features that disprove a hypothesis are opportunities to use data to inform the next thing that you build. Losing tests should be discussed. They are likely to provide the best insights for improving your product in ways you would not have considered otherwise.

Learning never stops, and for us experimentation is a kind of learning.

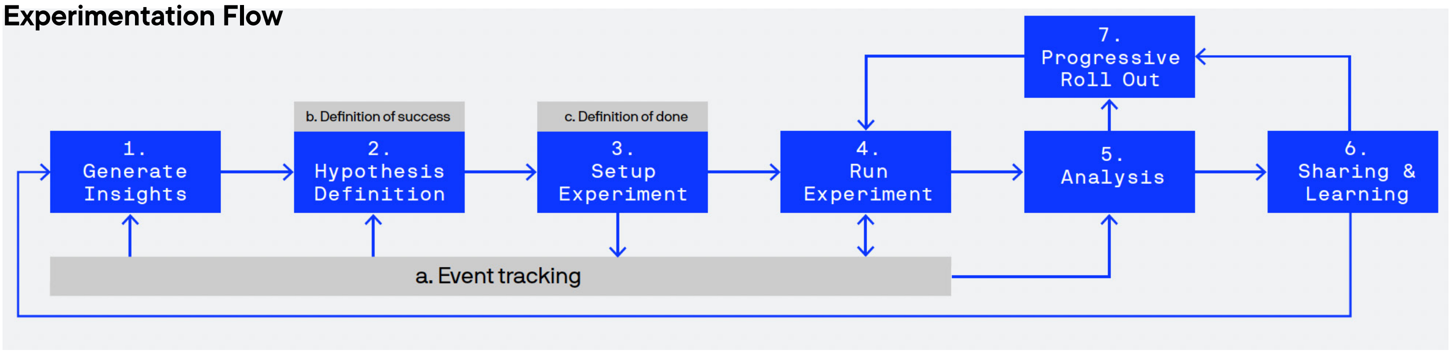
Aleksandra Vangelova & Abdelrahman Hussein
Product and Engineering / HelloFresh

“

We actually celebrate losing tests and discuss them even more than the things that merely confirm the initial expectation because we get new insights that could make us better.”

Luis Trindade
Senior Product Owner / Farfetch

Experimentation Flow



02

How product teams use data to test, learn & adapt

When Product and Engineering leaders from across the globe share the techniques they use to deliver better software, they point to curiosity as a key driver of higher engagement, growth and retention.

“Questions are the engine of exploration and discovery. It’s a process. Good questions lead to great questions, which create beautiful questions with answers that provide blinding insight.”

Bilal Mahmood
Head of Machine Learning, Amplitude

In this section, we’ll look at how that starts by asking bold questions to generate the data they need to make confident product decisions.

Why beautiful questions are the secret to growth opportunities

According to Amplitude’s Tanner McGrath, Head of Product Analytics, and Bilal Mahmood, Head of Machine Learning, organizations need to stay curious and adapt to survive. They believe it all comes down to asking ‘beautiful questions’ that shine a spotlight on opportunities with your customers and products.

The Amplitude model would look a little like this

Good Questions
How can I increase the number of subscriptions?

Insights
Users who previously downgraded but increased engagement are the most likely to subscribe—and so churn is not the end of the customer journey.

Beautiful Questions
Will resurrecting churned users be a scalable source of paying customers?

Answers that drive 10x impact.
Amplitude calls this process 'frictionless curiosity'—making the process between question and answer as fast and smooth as possible until, eventually, the best questions emerge. McGrath asserts that the answers to beautiful questions provide insights that significantly impact your business.

Product experimentation is a powerful way of reaching those answers with as little friction as possible and determining which insights inspire growth and innovation.

`Questions are the engine of exploration and discovery.`

Bilal Mahmood
Head of Machine Learning / Amplitude

"And one thing is clear, time and time again, the best product teams are curious, and this gives them a massive advantage over the competitors who are not. And this gap just keeps widening. Teams are constantly asking and answering other questions with each iteration. They're learning as fast as they ship and this is the edge."

Tanner McGrath
Head of Product Analytics / Amplitude

Why? What? How?

Oji Udezue takes a similar approach, describing the classic innovation loop as driven by 'Why, What, and How' questions. And for Udezue, you only get the right answers by leveraging customer discovery and experimentation.

When it comes to experimentation, Udezue lists a number of aspects you should keep in mind as you plan your experimentation pipeline. For example, don't get too attached to any specific preconceived ideas. Instead, use experimentation to follow the data, and if the data reveals something unexpected, then dig a little deeper to find out why this happened. Gaining insights from reliable statistically significant data early in the process reduces the risk of taking your product and business down the wrong path.

Put controls in place

Controls are another essential aspect to understanding the answers to your questions and the results of your experiments. When teams start conducting more experiments, it can be hard to determine the absolute rate of impact. Holdouts and controls provide the basis for getting this right by providing an absolute against which you can measure your results.

List your priorities

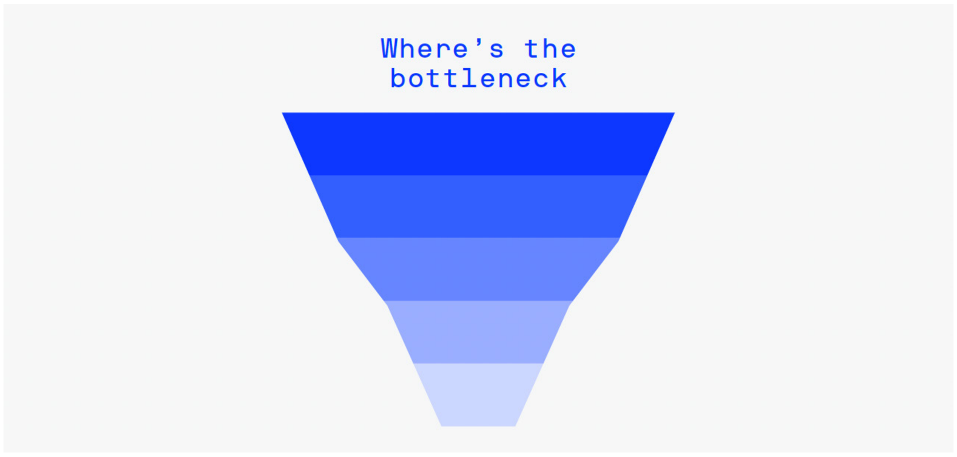
You also want to create a list of priorities. Think about all the metrics you want to optimize in terms of your strategy, then try and focus on them one at a time. Those teams that attempt to address multiple goals in a short space of time find it notoriously difficult. (More about metrics in the next section.)

Teams are learning as fast as they ship – and this is the edge.

Tanner McGrath
Head of Product Analytics / Amplitude

Focus on the right part of the funnel

Focusing on substantial changes will deliver faster significance, decisions and impact. Yet customer value tends to be towards the bottom of the funnel, so while directing your efforts there won't deliver the biggest lifts in the world, it will allow you to deploy large changes compared to the top of the funnel. Consider starting your optimizations at the bottom of the funnel, and then move up into the pure growth part and optimize how people come in.



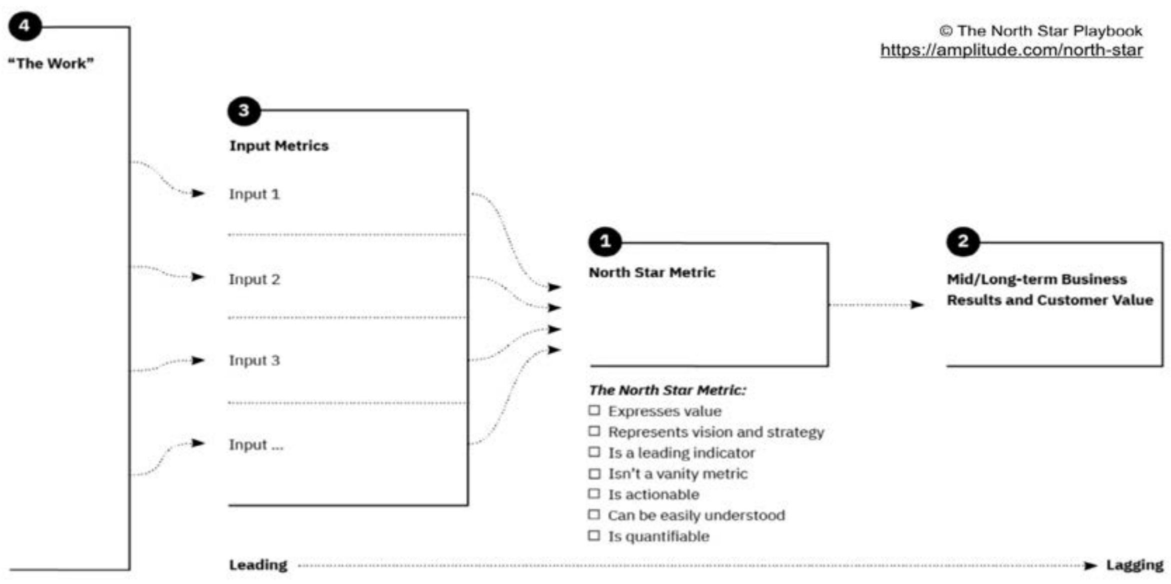
The single greatest thing you can do to make your testing experimentation better is to start with really great customer discovery.

Oji Udezue
VP of Product / Calendly

Align metrics for success

Your North Star metric is essential to aligning your approach to Test & Learn, helping you focus your goals and move faster. From that North Star metric, you can map out a whole tree of input metrics across different business areas. This helps different teams to understand the big picture, what they should focus on, and why. And as you go about trying to improve those different metrics, you also know you are feeding into the overall primary metric.

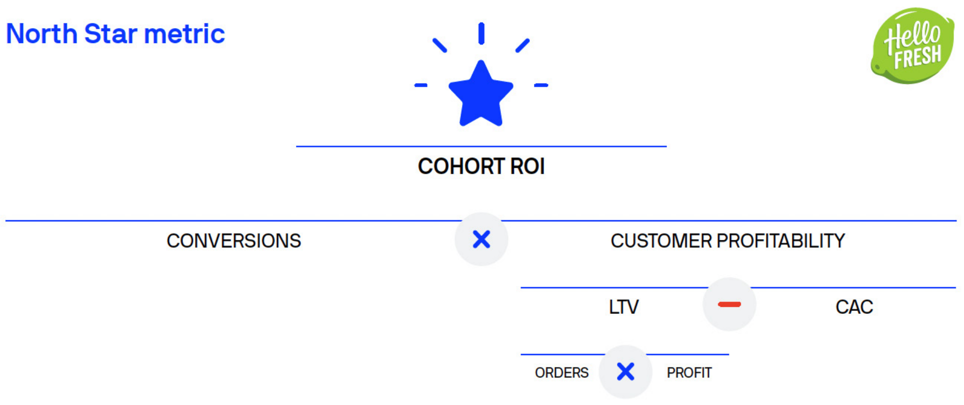
Retention metrics are a leading indicator of profitability for companies like HelloFresh, helping them measure the long-term impact of experiments and product changes. For example, Aleksandra Vangelova defines her North Star Metric as Cohort ROI, underlining how conversion and customer profitability are the focal points of their business.



Return on investment means converting users and making them profitable – and profitability means retention. We want customers to stay for longer and to keep orders coming."

Aleksandra Vangelova
Product and Engineering / HelloFresh

Aligning their North Star metric with customer retention focuses the business on sustainable growth. This breaks down into revenue per customer, average order value, and other financial metrics that tell product managers how much customers spend over what numbers of orders.



LTV = Total product benefit delivered over all time.

At Farfetch, Luis Trindade also evaluates experiments against success metrics, with certain provisos in place. Trindade promotes using an experiment to measure the impact of your product changes by focusing on the near-term effects for clarity.

You need to be careful about how far ahead you measure your chosen metric. If it's too far, things become very noisy, and results are slow to come in. To measure the impact of a change, focus on the short-term for clarity. Don't measure the change itself but the consequences of implementing it."

Luis Trindade
Principal Product Manager / Farfetch

For example, say you introduce a loyalty program to your customers as an experiment to improve LTV. Sign-ups and purchases are an interesting near-term metric for measuring engagement. Be sure to monitor what users do next. Do they return orders more frequently? What is their average order value? In looking down-funnel and segmenting user behavior over time and across audience segments, you gain a complete understanding of the experiment's impact on revenue.

Statistical significance underpins quality data

When you are looking to answer difficult questions, or make big decisions, statistically significant data is the key to certainty. That's why leading experts such as Oji Udezue and Aleksandra Vangelova recommend using a sample size calculator to measure how much time and traffic you need to reach significance in your tests.

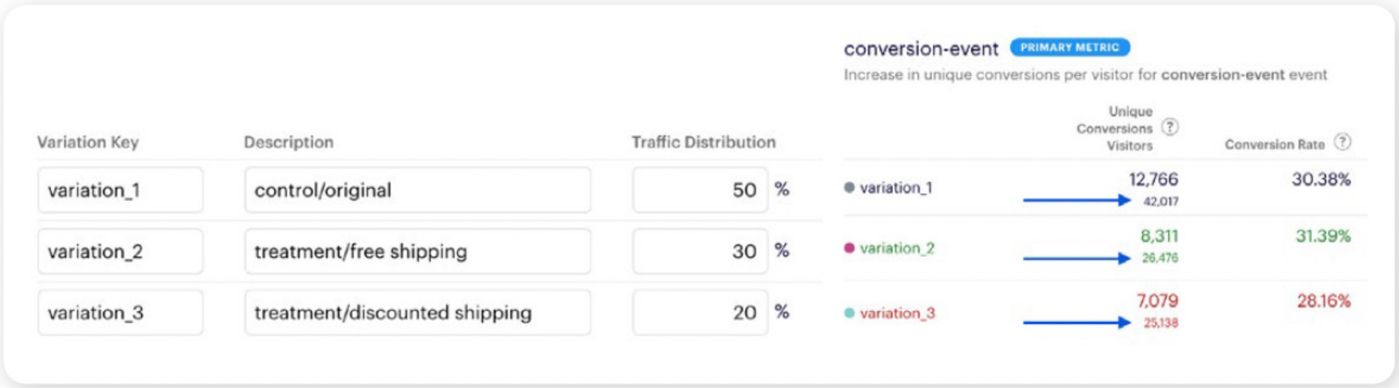
Lack of product traffic doesn't have to be a blocker. A platform like Optimizely will help you weed out inconclusive experiment variations early on by reducing the time it takes to get actionable results. Using machine learning to automatically allocate more traffic to experiment variations that show early promise impactful results, it results in a faster path to statistical significance.

Executing a product experiment requires a tremendous amount of infrastructure, and that means plenty of opportunities to unwittingly introduce bugs."

Michael Lindon
Staff Statistician / Optimizely

Watch out for data issues due to Sample Ratio Mismatch (SRM)

Data always needs to be tested for integrity. Sample ratio mismatch (SRM) is one example of where it can go wrong. Michael Lindon, Optimizely Staff Statistician, quotes SRM as a common issue that arises from incorrectly implemented experiments. It can be symptomatic of a range of potential data quality problems.



conversion-event PRIMARY METRIC

Increase in unique conversions per visitor for **conversion-event** event

Variation Key	Description	Traffic Distribution			Unique Conversions / Visitors	Conversion Rate
variation_1	control/original	50	%	● variation_1	12,766 / 42,017	30.38%
variation_2	treatment/free shipping	30	%	● variation_2	8,311 / 26,476	31.39%
variation_3	treatment/discounted shipping	20	%	● variation_3	7,079 / 25,138	28.16%

For example, say an experiment is set to 50% traffic to one variation and 50% traffic to another variation. If you saw a skew of 45%/55%, this would suggest an SRM issue. The main reason you should be concerned is when SRM causes invalid data results, a false bias can impact your metrics undetected.

According to Microsoft, SRM is more common than you might think—around 6% of the experiments they ran in a year exhibited a mismatch. A common pitfall is when the observed metric change is not a result of the test, but due to a bug introduced when implementing it.

SRMs: a real world case

MSN experimented with increasing the number of items on a carousel in an attempt to boost user engagement on a web page. To everyone's surprise, this had a very negative effect on user engagement. Or did it?

In actuality, users were more engaged for much longer—for so long they were accidentally classified as a bots classification algorithm and removed from the analysis. By excluding these users, MSN unwittingly introduced a significant and invalid bias into the experiment.

How do you negate the threat of SRMs and preserve the integrity of your data? You normally run a test and, if you detect an anomaly, you know that something is probably wrong with the experiment. The trouble is, by doing this at the end of the test, you write off the data you've already generated.

Optimizely changes that by allowing you to test for SRMs in real-time as the experiment is running. That enables you to detect any SRM present early on, get it fixed and prevent a large number of users from entering a faulty test, and retain the integrity of your results.

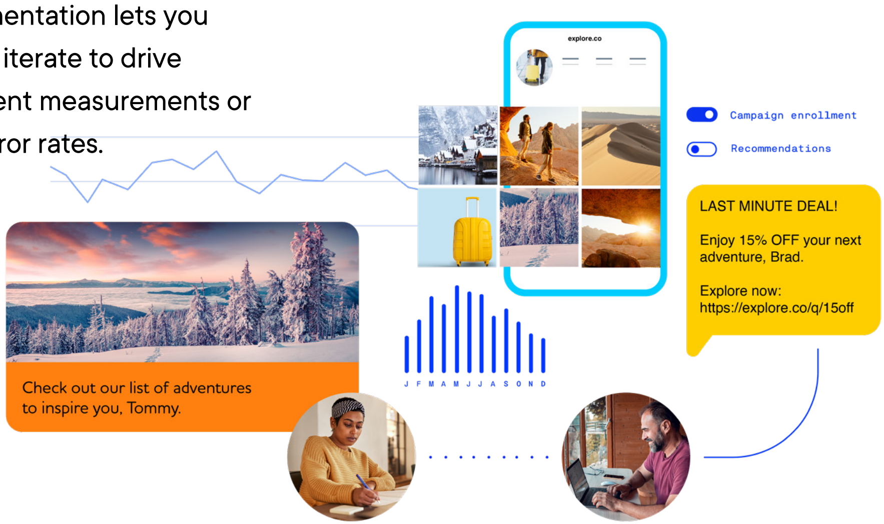
03

A quick guide to architecting a smarter tech stack

Delivering value is the ultimate goal for product development, growth hacking and data science teams. Feature flagging and experimentation are the fastest, safest route to testing and learning what works best with your users at scale. It provides the foundation for smarter tech stacks that drives ever better customer experiences.

Lawrence Bruhmuller, Optimizely's Chief Technology Officer, talks about the future of software development. He sees progressive delivery and experimentation as two sides of the same coin. Progressive delivery allows software development teams to validate quality and performance in production targeting key segments of your user base, using this real user impact to adapt quickly.

As a natural extension of progressive delivery, experimentation lets you validate new features and changes as A/B tests. Then iterate to drive important metrics, whether conversion and engagement measurements or 'under the hood' metrics such as latency, uptime or error rates.



LAST MINUTE DEAL!

Enjoy 15% OFF your next adventure, Brad.

Explore now:
https://explore.co/q/15off

Check out our list of adventures to inspire you, Tommy.

The 3 steps to building a Test & Learn architecture that drives better outcomes

STEP 1

Define your goals

First, identify your goals for managing product delivery and gathering trusted insights. Consider how they map back to phases in the product experimentation cycle. While the 'build-measure-learn' pattern is typical in product development cycles, it's essential to identify where you need to invest at each stage to hit your targets.

Your shortlist might look something like this:

1. Test and iterate on ideas.
2. Quickly identify the highest value product opportunities.
3. Reserve engineering time for the most important development.
4. Validate code quality and performance.
5. Prevent negative user experiences.
6. Introduce the right features to the right audiences, in the most impactful ways.
7. Use data to understand our customers.

STEP 2

Evaluate your performance

Next, figure out which areas of your tech stack need working on to achieve those goals.

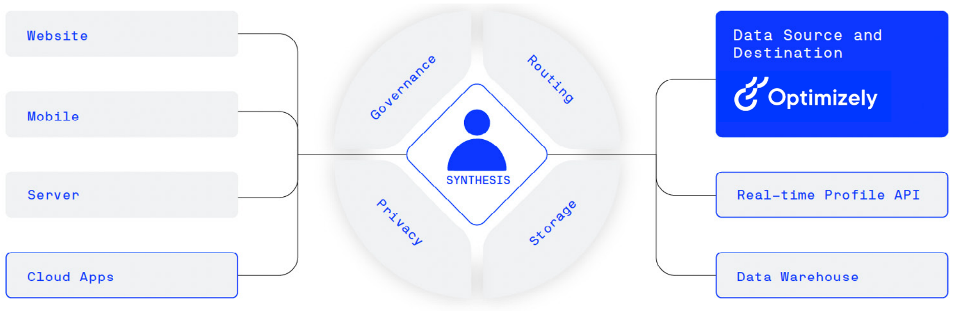
Where in your technology ecosystem could feature flags and experiments provide better results for your business?

The Product and Engineering teams at HelloFresh run retention experiments for extended periods to show long-term results. They use feature flagging and A/B testing on their website and work tirelessly to improve their iOS and Android apps. When they release new functionality, they do it gradually, as an experiment region by region. The outcome is a great user experience that leads to higher conversion, retention and business margins.

Plug gaps in your data pipeline

For modern product teams, speed and safety are not mutually exclusive. Data-driven product development is the key to moving fast without adding risk. But how customers react to those decisions is entirely dependent on the quality of the data itself. Incomplete, inconsistent, or hard to implement data will skew your understanding of customers and the way you deliver value.

Developing software with a smarter stack allows you to configure and test features with real users. Andy Yeo, Product Manager at Segment, also recommends using a Customer Data Platform (CDP) to avoid these gaps in your knowledge. A CDP collects and streamlines the inputs from your complete range of applications and channels to deliver accurate information you can easily put into action.



You can integrate Segment with Optimizely as a destination by measuring your experiments against the unified metrics tracked in Segment. Implementing Optimizely as a data source will automatically send experiment events to Segment, allowing you to track which customers see which A/B test variants.

Enhance existing internal platforms

Farfetch's Principal Product Manager, Luís Trindade, uses Optimizely's entire platform to accelerate prototyping, iteration and measure the value of his work. Farfetch developed an internal platform for server-side testing. By powering parts of their experimentation platform with Optimizely Full Stack, the team leverages advanced techniques like multi-armed bandits. They use Optimizely Web to quickly deploy painted door tests and measure the business impact of their exploratory ideas before committing developer time.

Buy or build?

Use this guide to understand the considerations and costs for building, buying, or enhancing a feature flagging and experimentation platform. If you decide to look for a partner to supplement your existing internal platform, make sure you evaluate how easily you can integrate their systems. Optimizely's Full Stack SDKs are open source, while Rest API is both flexible and developer-friendly.

STEP 3

Develop an innovation model

As far as Nicole Forsgren is concerned, staying ahead of your competition is about being data-driven and unafraid to try new things. To get Test & Learn right, you need the technology to quickly run A/B tests and achieve reliable results AND the culture to support these practices. Innovation happens when technology and development principles combine to support creativity and experimentation.

Make sure you are building the right kind of Test & Learn foundation where questioning, iterating and proving value provide the foundation for every feature.

Building the right questions into product development will help you develop features that balance customer-centricity, business value, resourcefulness and efficiency.

7 questions to build into your development plans

1. What is a good outcome for this product?
2. Are there different iterations of the feature that should be configurable without a deployment?
3. How can development be minimized to validate the approach?
4. Which users should be the first to test the product?
5. Which customer segments are likely to respond in unique ways?
6. How can the product rollout be phased, managed and iterated on outside of deployment cycles?
7. What upfront investments will minimize risk and unexpected engineering requests?

04 Conclusion

Test & Learn. This is how teams move quickly and confidently.

How do you get new products to customers as fast as possible? And how do you know that you're building the right thing? Answer these questions with trusted data by continually testing and learning with feature flags and experiments.

Feature flagging and experimentation belong together as the only way a platform can serve the needs of the entire product development team. This enables organizations to put customers front and center and iterate quickly at every stage: rapid prototyping in the ideas phase, performing rollouts in the build and release phase, optimizing and personalizing in the launch and measurement phase. All supported by real-time results and analysis, which lead to better decisions and the best customer experiences.

So, how do you move fast and get it right?

Optimizely products offer the capabilities to support your teams in moving fast and getting it right. As the leader in progressive delivery and experimentation, we offer a unified platform for feature flagging, progressive rollouts, and A/B/n testing across the entire end-to-end customer journey.

We help you put customer data at the center of your product development and delivery process.

You can validate ideas quickly through painted door tests and testing MVPs on real users. Accelerate delivery through frequent small releases behind feature flags. Progressively roll code out to your entire customer base. And then quantify the impact of your features through A/B tests and multivariate tests, helping you optimize the business impact of your products post-production.

Your customers get features they love. And your teams spend less time re-working ideas, fixing bugs and searching for data—freeing them up to innovate and create value.

Want to know more about using feature flags and experimentation to test, learn and build better user experiences quickly and safely? **Talk to us**.

Optimizely is built for your whole team

Optimizely is the only solution built for your entire team to address the challenges of engineers, product managers, growth marketing and data analysts. Our platform combines progressive delivery for engineers with deep experimentation capabilities for product, analytics and growth teams, helping you confidently deliver what your users want, faster.

Make every release high quality, and high impact by quickly and safely validating code and features with real users through the entire software development lifecycle.

Speed
Engineers can ship code multiple times a day, giving more granular control over launches and deploys to customers.

Quality
There's no testing like testing in production. Optimizely mitigates the risk, eliminating painful rollbacks and hot fixes and catching bugs early.

Confidence
Gain the confidence of knowing you're building the right thing by validating products and features with customers.

Freedom
PM, engineers, growth teams, and data analysts gain the capability and the freedom to do their jobs with minimum friction and as close to customers as possible.

At Optimizely, we're on a mission to help people unlock their digital potential. Our leading digital experience platform (DXP), equips teams with the tools and insights to create and optimize in new and exciting ways. Now, companies can operate with data-driven confidence to create hyper-personalized experiences. Building sophisticated solutions has never been simpler. Optimizely's 900+ partners and 1100+ employees in offices globally are proud to help over 9,000 brands, including Electrolux, Uber, Visa, WSJ, Santander, The Red Cross and Mazda enrich their customer lifetime value, increase revenue and grow their brands. Learn more at **Optimizely.com**