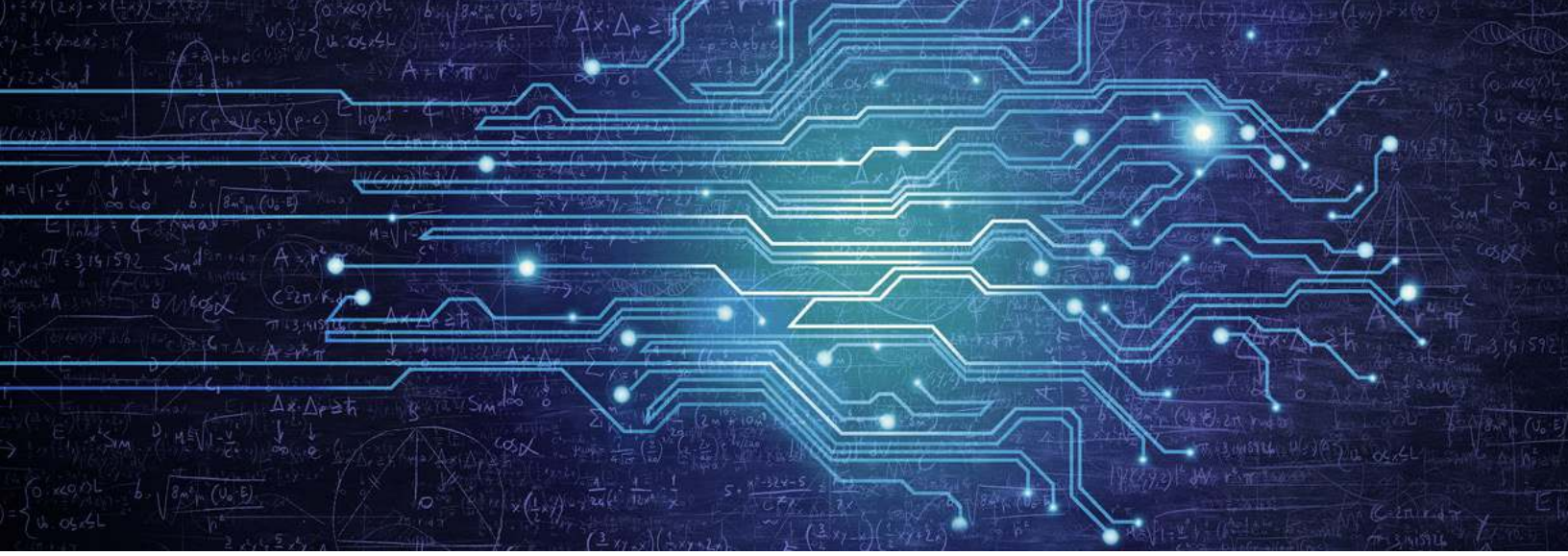
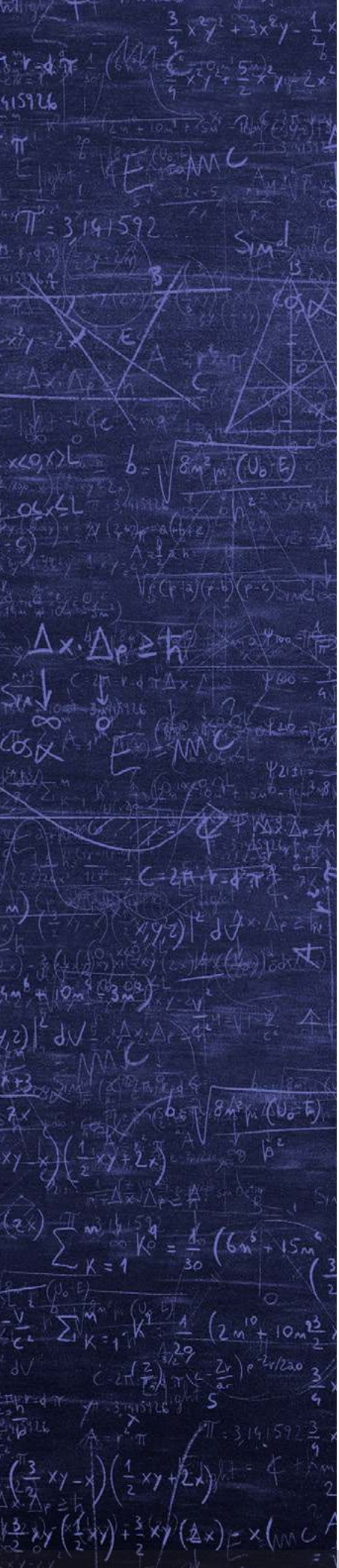


The Definitive Guide to Evaluating Experimentation Platforms for Product and Engineering Teams

Key considerations and best practices to help technical teams get from evaluation to proof of concept

CITO RESEARCH

split



CONTENTS

EVALUATING EXPERIMENTATION PLATFORMS

At top Internet firms, experimentation guides data-driven product evolution. It provides objective data to aid in prioritizing product development efforts.

At top Internet firms, experimentation guides data-driven product evolution. It provides objective data to aid in prioritizing product development efforts.

Leading Internet companies have invested literally hundreds of millions of dollars in time and infrastructure to create the ideal experimentation platform for their business. Key team members from such organizations describe their architecture for experimentation, often presenting at conferences and writing blog posts.[1] A recent IEEE journal article based on in-depth interviews with data scientists, engineers, and program managers across Microsoft presents an experiment evolution model that can be used by companies to gauge their maturity in conducting experiments to drive product direction. In that article, the authors quantify the value of experimentation: "The impact of scaling out the experimentation platform across Microsoft is in hundreds of millions of dollars of additional revenue annually."[2]

Today, much of the infrastructure needed for experimentation has been productized via commercial platforms and as well as made available via various open source components.

The question then becomes how to select the experimentation platform that is right for your business.

This evaluation guide explores approaches to building or buying an experimentation platform, offering guidance for those seeking to find the right way to adopt experimentation as part of their product development process. It then describes how to use a proof of concept to vet the chosen approach to experimentation.

1. A few recent examples include Uber, Netflix, and Airbnb.

2. Aleksander Fabijan, Pavel Dmitriev, Helena Holmström Olsson, and Jan Bosch, "The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-driven Organization at Scale," Proceedings of the 39th International Conference on Software Engineering, 2017, https://dl.acm.org/citation.cfm?id=309746

DETERMINING YOUR EVALUATION CRITERIA AND COMPARING OPTIONS

The first step in evaluating an experimentation platform is to determine your buying criteria.

Criteria	Considerations
Scope of experiments	On what parts of a product can experiments be run? • Front end (UI/design level) • Backend (feature level)
Teams supported	Which teams can participate in creating, running, and analyzing results from experiments? • Engineers • Architects • Data Scientists • Product Managers • Marketers
UI/UX	Does the platform have a UI? Who is it designed for?
Targeting capabilities	How do you want to target users: • Based on percentages, randomizing who sees a feature? • Based on multiple criteria, such as which products a customer purchased, their location, their logon time? • By creating reusable cohorts of specific users?
Analytics capabilities and metric integration	• Does the platform have native analytics capabilities or do you need data scientists to gather and analyze data using external analytics tools? • Is data joined up with events so you can analyze which users saw which features? • Does the platform give you statistical analysis around what caused a change in metrics? • Can business metrics be defined across experiments or must they be defined per experiment?
Rolling back experiments if needed	• Can you define a point to roll back to if a new feature has issues? • Who can roll back to that point? (Engineering only? Or other users)
Technical specs and operational impact	• Which programming languages are supported? • What integrations are available? • How much engineering work is required to set up the product? Maintain SDKs? • What is the performance impact of using the platform?
Technical debt management	• Does the platform help you track and clean up code that is no longer needed?
Security, privacy, governance, management	• Does the platform require sharing of user details or is user information kept private? • Does it offer centralized management/governance features (permissioning, management console, management across development environments)?

Experimentation can be performed on the front end (browser level) using A/B testing as well as at the feature level, by turning features on and off in backend code and rolling out features to select groups of users.

Comparing Options for Experimentation

Experimentation can be performed on the front end (browser level) using A/B testing as well as at the feature level, by turning features on and off in backend code and rolling out features to select groups of users. This section overviews options for experimentation along with characteristics to evaluate.

Open Source

Organizations that want to get started with experimentation often look at building their own capabilities using open source components. Build-versus-buy is an important consideration for many organizations.

Key characteristics to evaluate:

- To date, more open source projects are available for feature flags than for A/B testing.
- By nature, open source does not require upfront license costs. It does require engineering resources.
- Open source tools are language specific, so experiments cannot be managed across microservices implemented in different languages.
- The lack of a UI limits experimentation to engineering resources.
- Analytics of experiments must be performed using separate tools.

Feature Management Platforms

Feature management platforms enable you to perform experiments by turning features on and off via a UI.

Key characteristics to evaluate:

- Platforms often lack integrated analytics, so analyzing experiment data requires separate tools and data science resources.
- UI is typically designed for engineering, potentially limiting participation from groups like product managers.
- Often has strong targeting capabilities and multiple tool integrations.

Full-stack experimentation platforms support conducting experiments on backend code as well as on front end code.

A/B Testing Products

Testing variations in user interface and user experience at the browser or app level is the purpose of A/B testing products. This type of experimentation predates widespread use of feature flags, and as a result some organizations call all of their experimentation A/B testing.

Key characteristics to evaluate:

- Support for dynamic JavaScript or CSS injection.
- UI that enables web marketing managers to set up and analyze WYSIWYG codeless experiments.
- Alignment of platform with your overall experimentation goals. Do you want to conduct code-based experiments as well as UI/UX experiments?

Full-Stack Experimentation Platforms

Full-stack experimentation platforms support conducting experiments on backend code as well as on front end code. As such, they are designed to become a key element of the agile software development and delivery process, enabling data-driven product development.

Key characteristics to evaluate:

- How well do targeting and analytics capabilities match your requirements?
- Ease of use for product development teams
- Support for experiments throughout their lifecycle and across the development landscape
- Whether targeting requires pushing user data into the platform (which may raise privacy issues).

Platform Selection Matrix

	Open Source*	Feature Management	A/B Testing	Full-Stack Experimentation
Scope of experiments: Front end or backend?				
Front end: Support for UI-level A/B testing		✓	✓	✓
Backend: Support for feature flagging and code-based tests	✓	✓		✓
Teams designed for				
Engineering, software architects	✓	✓		✓
Marketing, product management, UI/UX teams			✓	✓
Data scientists				✓
UI/UX and planning for scale				
Includes a UI		✓	✓	✓
Support for tagging features		✓		✓
Support for starring features				✓
Support for projects		✓	✓	✓

* Open source projects vary as to their capabilities. Values here apply to most open source projects in this category.

	Open Source*	Feature Management	A/B Testing	Full-Stack Experimentation
Targeting capabilities				
Randomization of experiments (rollout to a percentage of users)	✓	✓	✓	✓
Create reusable cohorts of specific users		✓		✓
Types of targeting criteria available:				
Exact string match	✓	✓	✓	✓
Strings	✓	✓		✓
Numeric	✓	✓		✓
Date and time		✓		✓
Boolean	✓	✓		✓
Regular expression		✓		✓
Sets of values (customer bought these SKUs)				✓

* Open source projects vary as to their capabilities. Values here apply to most open source projects in this category.

	Open Source*	Feature Management	A/B Testing	Full-Stack Experimentation
Analytics capabilities and metric integration				
Analytics to support data-driven decisions based on feature uptake			✓	✓
Includes native analytics capabilities			✓	✓
Visibility into which users see which experiments				✓
Statistical analysis to surface cause for changes in metrics			✓	✓
Ability to define business metrics across experiments (such as conversion rate, page load times, etc.)				✓
Ability to join event data automatically to experiments to support business metrics collection				✓
Rolling back experiments if needed				
Ability to define a roll back point for experiments	✓	✓		✓
Teams that can roll back experiments				
Engineering	✓	✓		✓
Any team		✓		✓
Audit trail included		✓	✓	✓

* Open source projects vary as to their capabilities. Values here apply to most open source projects in this category.

	Open Source*	Feature Management	A/B Testing	Full-Stack Experimentation
Technical specs and operational impact				
Support for multiple programming languages		✓		✓
Integrations:				
Chat tools (such as Slack, HipChat)		✓		✓
APM tools (such as New Relic, AppDynamics, Datadog, Jira, Librato, Rollbar)		✓		✓
Webhooks		✓		✓
Logging tools and exception handling software (such as Sumo Logic and Papertrail)				✓
Web analytics (such as Google Analytics)			✓	
Designed for minimal performance impact (milliseconds)		✓		✓
Technical debt management				
Monitors ongoing status of experiments				✓
Enables central view of which experiments are complete, so code can be cleaned up				✓
Integrates with ticketing system for followup		✓		✓

* Open source projects vary as to their capabilities. Values here apply to most open source projects in this category.

	Open Source*	Feature Management	A/B Testing	Full-Stack Experimentation
Security, privacy, governance, management				
User data kept private (not shared with platform)		✓		✓
Permissioning system (read-only for certain teams, access to staging for engineers)		✓		✓
Management console		✓		✓
Ability to manage experiments across development environments (dev, staging, production)		✓		✓
Engineering effort required for:				
Platform setup	High	Medium	Medium	Low
Experiment setup	High	Medium	Low	Low
Experiment cleanup (tech debt management)	High	High	High	Low
Integrating/maintaining SDKs	N/A	Low	High	Low
Integrating event data per experiment	N/A	N/A	High	None

* Open source projects vary as to their capabilities. Values here apply to most open source projects in this category.

CITO Research Recommendations for Selecting an Experimentation Platform

An experimentation platform should support your long-term vision and goals for data-driven product development. That means selecting or assembling a solution that enables experimentation to become a way of life for all the teams in your organization, from product management to engineering.

An experimentation platform should support your long-term vision and goals for data-driven product development. That means selecting or assembling a solution that enables experimentation to become a way of life for all the teams in your organization, from product management to engineering.

Build versus buy

Open source platforms generally do not support ease of use for broader teams, nor do they provide analytics to drive decision making (analytics can be performed with the aid of data science resources).

Build versus buy decisions are not always straightforward. If you're leaning toward open source, we recommend considering:

- The impact of diversion of engineering talent from your own product to an area that is not your core competency.
- Conducting at least one PoC to evaluate alternatives before deciding to build your own solution.
- Limitations of open source platforms (e.g., most are tied to a single development language, and many have no UI)

Room to iterate and grow

CITO Research recommends considering whether the experimentation platform you're evaluating offers room to grow in the depth, targeting, and complexity of experiments you'd like to perform, the metrics you would like to collect from those experiments, and the teams you'd like to have support them. As with most mature software, initial use or a PoC may not take advantage of all features. The question is which features you need to support your own experimentation initiatives, both now and in the future.

For example, consider criteria for targeting experiments. Do you want to be able to target using customer characteristics in your databases, customer location, and precise log-in time? Show a particular feature only to customers who have purchased particular SKUs?

Look for a platform that aligns with your most important goals, particularly in terms of metrics you'd like to collect and insights you'd like to glean.

Do you envision broad support for data-driven product development? If so, perhaps the most important area as you consider room for growth and scaling is the number and makeup of the teams that may eventually participate in experimentation. Experimentation platforms vary widely in terms of the type of users they are designed to support. Many open source platforms do not even have a UI, limiting those who can participate to more technical users. Other platforms are designed to allow multiple teams and cross-functional users to use one unified platform to collaborate as needed. Look for a platform that offers proper segmentation so that various groups conducting experiments can work independently and don't slow each other down while at the same time guarding against mistakes could impact end users.

Align with business goals and policies

What is most important to your business? Response time? Privacy of user data? Look for a platform that aligns with your most important goals, particularly in terms of metrics you'd like to collect and insights you'd like to glean.

Also consider how easily metrics can be integrated into experiments. Can metrics be captured across all experiments, so that page load times, conversion rates, or other key business metrics can be compared across all the code you deploy? Platforms that capture a consistent panel of metrics across experiments may require less setup and data integration work than those that require you to set up metrics on a per-experiment basis.

Performance may be a key consideration business-wide, and you may have high expectations in terms of response time for your users. Experimentation platforms vary in terms of performance overhead; be sure to evaluate the impact as you consider experimentation platforms.

Another area with business-wide implications is sharing of user information with other platforms. For certain industries, this narrows your choices to building or buying an experimentation platform that is architected in such a way that experiments do not require sharing user data. Even if that's not an industry-specific or regulatory requirement, it's a point well worth considering as you evaluate experimentation platforms.

Look for a platform that makes it easy to clean up experiments that have run their course.

Minimize technical debt

Experimentation means that multiple versions of features are deployed. Look for a platform that makes it easy to clean up experiments that have run their course. Creating branches in your code ultimately means that unused code needs to be cleaned up. Look for a platform designed to help you manage technical debt.

SETTING UP A SUCCESSFUL POC

After determining the criteria that are most important to you, the next step is to launch a PoC where you prove the value of the solution to your organization, with your teams and your data. The following sections offer a suggested test plan and elements to evaluate during your PoC. Adapt these resources to meet your needs.

High-level PoC test plan

Scoping, use case, and design	Identify and define key evaluation criteria, and design accordingly.
Onboarding	Learn about the solution to get up and running quickly.
Cross-team evaluation	Ensure that all relevant teams can participate in the PoC (engineering, product management, data science, etc.)
Scheduling	Set a timeframe for achieving PoC goals. Schedule intermediary objectives to stay on track. Ramp up time for the PoC can be an indication of the difficulty of adopting the solution at scale.
Preparation and evaluation	Prepare experiments and evaluate results (see "Product evaluation elements" in the next section)

Product evaluation elements

PRODUCT EVALUATION	WHAT TO LOOK FOR
Scope and UX	***Evaluate the scope of tests and fit for intended users.***
Scope	Does the product support feature flagging, A/B testing at the UI level, or both?
Teams	Is the product easy to use for all desired teams? Does it have a UI?
Tagging	Can experiments be tagged for easy categorization?
Starred	Can experiments be starred for easy retrieval?
Projects	Are experiments grouped by project to simplify the user experience?
Targeting Capabilities	***Evaluate how the platform enables targeting of experiments.***
Randomization	Does the platform enable you to roll out experiments to a random group of users, determined by a percentage?
Targeting criteria	What flexibility does the platform offer in terms of targeting criteria?
Time-based criteria	Is time based targeting available? How narrow can the time window be?
Regex matching	Is targeting criteria defined as an exact match or is it more flexible (starts with, contains, etc.)?
Sets	Can you target users with a set of characteristics (i.e., those who purchased certain SKUs)?
Boolean	Is Boolean logic supported in building up multiple targeting criteria?
Adding users	Can you create reusable cohorts of specific users or must you add users one by one?

PRODUCT EVALUATION	WHAT TO LOOK FOR
Analytics and metrics	***Evaluate the sophistication of the solution with regard to analytics per experiment as well as changes in metrics across the environment.***
Analytics capabilities	Does the platform have native analytics capabilities or are data scientist skills required to analyze results?
Visibility	Does the platform enable you to determine which users saw which experiments?
Trends:	Does the platform enable a view across the code base to see which experiments are statistically significant?
Measures	Does the platform provide the statistical significance measures you need?
Metrics and event data	How are business metrics defined? How is event data correlated? Per experiment or across the environment?
Causality	Can you determine the cause of changes in metrics?
Rollback	***Evaluate how problems with experiments are handled.***
Rollback point	Can you define a rollback point if an experiment causes issues?
Rollback authority	Who can roll experiments back if needed?
Audit trail	Is an audit trail included?

PRODUCT EVALUATION	WHAT TO LOOK FOR
Technical specs and operational impact	***Evaluate the technical capabilities and operational impact of the solution.***
Programming languages	Are all required languages supported?
Integrations	Does the platform include integrations with other platforms you use?
SDKs	Does the platform include SDKs? How are updates handled?
Performance	What is the performance impact of using the platform?
Technical debt management	***Evaluate how the platform helps manage technical debt.***
Status	Monitors status of all experiments to facilitate identification of code for cleanup
Follow up	Integrates with tools to assign cleanup tasks
Security and management	***Evaluate how the platform handles user data and its management capabilities.***
Privacy	Is user data pushed into the platform?
Permissioning	Can permission levels be assigned to various users?
Management console	Does the platform offer a central view of all experiments?
Support across environments	Does the platform enable you to manage experiments from dev to staging to production?

ESTIMATING TOTAL COST OF OWNERSHIP

After evaluating various platforms, the next step is to determine your ROI, both short and long term. Look to capture both your Capex and Opex, including:

- Costs to get the platform up and running
- Ongoing costs to define, analyze, and clean up experiments

Engineering costs

The amount of engineering work required to get an experimentation platform up and running, as well as its ongoing costs, varies widely. Estimate the engineering time it will take to set up the platform and onboard users. While this is an important dimension, it captures only one aspect compared to ongoing engineering costs, which include:

- **Defining each experiment.** How much can be done by product managers versus engineers? How long does it take? How would additional work impact the overall backlog of requests?
- **Keeping software up to date.** How much effort is involved? Must software libraries or SDKs be synchronized? Is a separate server required for synchronization?
- **Tracking experiments across the development environment.** How much effort is involved in moving experiments from dev to staging to production?
- **Analyzing experiments.** Does analysis of experiments require data science resources, perhaps using a separate platform, or does the platform facilitate analysis by various stakeholders?
- **Technical debt management.** How much engineering effort is required to clean up experiments over time?

Speed to deployment

The faster you can get started with experimentation, the faster you can start realizing value from data-driven product development.

In the build-versus-buy equation, building takes time and effort, and can take you into areas not directly related to your core competency. This in turn delays the value you could get from features such as centralized management.

To assess costs versus speed to deployment, use the Engineering Cost Estimator Worksheet included in the Appendix. Add up the engineering effort in weeks for each task and multiply by the average cost per week for an engineering FTE.

Costs to scale

Operational impact: The goal is to make sure that experiments don't interfere with each other and don't negatively impact the performance of the product. Operations tools must exist or be developed for this purpose.

- **Experiment creation and collaboration:** How much self-service for creation and analysis of experiments is supported? What parts of the process have expertise bottlenecks?
- **Experiment portfolio maintenance:** Experiments come and go, but they should not live on in the code base forever. Processes for maintaining a life cycle of tests must be developed and supported.
- **Data management:** How is data for segments managed? What level of personnel is required?
- **Software support:** Keep in mind the level of support you'll need from the vendor or your approach to keeping open source up to date. If you lack the needed expertise, this can also have a major impact on team productivity.

The major Internet companies have had continued success in large measure because of their experimentation platforms and their attendant ability to innovate, capture, and sustain market share.

Total value of platform

Once you have assessed the engineering costs to support the types of feature experimentation you want to support and the levels of participation by all relevant parties, you'll have a clearer picture of the TCO and the ROI that the experimentation platform you've selected can bring to your business.

MOVING FORWARD WITH ADOPTION

The major Internet companies have had continued success in large measure because of their experimentation platforms and their attendant ability to innovate, capture, and sustain market share. The obvious problem with creating such a platform is the engineering investment in building it and evolving it over time to support broad and deep experimentation to enable data-driven product development.

Building a platform that offers the same type of enablement as web scale companies enjoy is not a simple undertaking. Such a platform must seamlessly fit into the CI/CD process, with minimal friction for engineering teams. At the same time, the platform must provide abundant data to drive product development, making sophisticated analytics broadly accessible to all stakeholders. It must address privacy and security concerns, and help manage the lifecycle of experiments across the deployment landscape as well as sunsetting them gracefully once they are complete. Its UI must enable all teams involved in product development, from product management to architects to marketers. It must support all the development languages in use today, and given the adoption of microservices, that may be in use tomorrow.

CITO Research has assessed the experimentation landscape from multiple angles, from data science to engineering to product management.

From the research we've done, it seems that Split is furthest along toward productizing full-stack experimentation for the following reasons:

- The convenience factor for technical teams before, during, and after testing is a huge ingredient for success. Split's engineering is focused on making the developer experience smooth and seamless during development, but also adding integrations that ensure that the code base is cleaned up after experimentation.

- The integration of full-stack experimentation with capabilities and practices that support A/B testing, agile, CI/CD, DevOps and feature management is also crucial. While our research is focused on understanding A/B testing, full-stack experimentation, and the deep experimentation of the web scale companies, we see that Split's vision for a higher level of integration of development and research-based, statistically mature product management is deserving of the name they give to it: Feature Experimentation. Numerous integrations, capabilities, and UX innovations allow Split to serve many audiences in ways that feel natural.
- Finally, the full-stack experimentation space is new and evolving. It is vital that when using products in such a space, you are eager to grow in the same direction you want to go. Full-stack experimentation is not a feature added to some existing system, but the final step that integrates A/B testing, Agile, DevOps, CI/CD, and feature management into a unified, research-based, statistically mature product development process. Split's feature experimentation vision is the closest I've seen in any product to what I believe full-stack experimentation will become.

Our recommendation is that organizations that are serious about embedding experimentation into their agile product development lifecycle conduct a PoC for Split.

We also believe in the value of products that are fully realized. Organizations that consider open source solutions or rolling their own often do so because there is simply no budget to consider alternatives. If there is money to be made from a better product, you want to be able to implement a wide program of experimentation as soon as possible. Pretending you can get there with a limited amount of in-house development must be weighed against the opportunity cost of not engaging in data-driven product development. It is important to do the math and understand the level of value at stake. In addition, we believe engaging with an experimentation product that is fully realized will teach you what experimentation really means to you and the whole organization.

This paper was written by CITO Research and sponosred by Split

CITO RESEARCH

APPENDIX: ENGINEERING COST ESTIMATOR WORKSHEET

Use this worksheet to estimate the engineering costs associated with the experimentation platform you select. Determine the weeks of effort required, multiplied by the average weekly cost for an engineering FTE.

	Work Effort Required (Weeks)	Total Costs (# of Weeks x Weekly FTE Cost)
Platform setup time		
Effort required to prepare data for experiments		
Effort to get needed targeting capabilities in place		
Effort to get required analytics capabilities in place		
Effort to handle set up of each experiment*		
Effort to analyze data from experiments (data prep, data science resources)*		
Effort to prune experiments once complete*		
Effort to field requests for creating experiments (if there is no self-service experimentation)*		
Effort to determine impact of experiments on operations		
Effort to move experiments from dev to staging to production*		

* Multiply by the number of experiments anticipated per quarter or per year.