2017 IEEE/ACM 39th International Conference on Software Engineering

The Evolution of Continuous Experimentation in Software Product Development

From Data to a Data-driven Organization at Scale

Aleksander Fabijan[1], Pavel Dmitriev[2], Helena Holmström Olsson[1], Jan Bosch[3]

[1]Malmö University
Faculty of Technology and Society
Malmö, Sweden
aleksander.fabijan@mah.se
helena.holmstrom.olsson@mah.se

[2]Microsoft
Analysis & Experimentation
Microsoft, One Microsoft Way,
Redmond, WA 98052, USA
padmitri@microsoft.com

[3]Chalmers University of Technology
Dep. of Computer Science & Eng.
Göteborg, Sweden
jan.bosch@chalmers.se

***Abstract*—Software development companies are increasingly aiming to become data-driven by trying to continuously experiment with the products used by their customers. Although familiar with the competitive edge that the A/B testing technology delivers, they seldom succeed in evolving and adopting the methodology. In this paper, and based on an exhaustive and collaborative case study research in a large software-intense company with highly developed experimentation culture, we present the evolution process of moving from ad-hoc customer data analysis towards continuous controlled experimentation at scale. Our main contribution is the "Experimentation Evolution Model" in which we detail three phases of evolution: technical, organizational and business evolution. With our contribution, we aim to provide guidance to practitioners on how to develop and scale continuous experimentation in software organizations with the purpose of becoming data-driven at scale.**

A/B testing; continuous experimentation; data science; customer feedback; continuous product innovation; Experimentation Evolution Model; product value; Experiment Owner

I. INTRODUCTION

Software development organizations and their product development teams are increasingly using customer and product data to support decisions throughout the product lifecycle [1], [2]. Data-driven companies acquire, process, and leverage data in order to create efficiencies, iterate on and develop new products, and navigate the competitive landscape [1]. Digitally adept and technology driven companies are as much as 26 percent more profitable than their competitors [3]. Recent software engineering research reflects this situation with a number of publications on how to change and efficiently conduct controlled experiments to become data-driven [4], [5], [6], [7], [8], [27]. The role of data scientists is increasingly gaining momentum in large software companies [9]. However, despite having data, the number of companies that efficiently use it and that successfully transform into data-driven organizations stays low and how this transformation is done in practice is little studied [10], [11].

In this paper, we present the phases that teams at Microsoft evolved through in order to become data-driven at scale by establishing a controlled experimentation platform and a data-driven mindset. The impact of scaling out the experimentation platform across Microsoft is in hundreds of millions of dollars of additional revenue annually. The journey from a company with data to a data-driven company, however, was not a jump but rather an evolution over a period of years. This development occurs through phases and we illustrate this process by creating the "Experimentation Evolution Model". With this model, we describe the steps to take while evolving data-driven development practices towards continuous experimentation at scale. With our contribution, we aim to provide guidance to practitioners on how to develop and scale continuous experimentation in software organizations and thus become truly data-driven.

The paper is organized as follows. In Section II we present the background and the motivation for this study. In section III, we describe our research method, the data collection and analysis practices and our case company. Our empirical findings are in section IV. In section V, we present our main contribution - the "Experimentation Evolution Model". Finally, we conclude the paper in section VI.

II. BACKGROUND

Rapid delivery of value to customers is one of the core priorities of software companies [8]. With this goal in mind, companies typically evolve their development practices. At first, they inherit the Agile principles within the development part of the organization [12] and expand them to other departments [13]. Next, companies focus on various lean concepts such as eliminating waste [14], removing constraints in the development pipeline [15] and advancing towards continuous integration [16] and continuous deployment of software functionality [10]. Continuous deployment, however, is characterized by a bidirectional channel that enables companies not only to send data to their customers to rapidly prototype with them [17], but also to receive feedback data from products in the field. The intuition of software development companies on customer preferences can be wrong as much as 90% of the time [18], [19], [20]. The actual product usage data has the potential to make the prioritization process in product development more accurate as it focuses on what customers do rather than what they say [21], [22]. Controlled experimentation is becoming the norm in advanced software companies for reliably

1558-1225/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE.2017.76

evaluating ideas with customers in order to correctly prioritize product development activities [4] [5], [6], [7], [8].

A. Controlled Experiments

In a controlled experiment, users are randomly divided between the variants (e.g., the two different designs of a product interface) in a persistent manner (a user receives the same experience multiple times). Users' interactions with the product are instrumented and key metrics are computed [4], [23]. One of the key challenges with metrics is to decide on which to include in an Overall Evaluation Criteria (OEC). An OEC is a quantitative measure of a controlled experiment's objective [24] and steers the direction of the business development. In controlled experimentation, it is intuitive to measure the short-term effect, i.e., the impact observed during the experiment [25]. Providing more weight to advertisement metrics, for example, makes businesses more profitable in the short-term. However, the short-term effect is not always predictive of the long-term effect and consequently should not be the sole component of an OEC [26]. Defining an OEC is not trivial and should be conducted with great care. Kohavi et al. [4], [26], [27] in their papers present common pitfalls in the process of establishing a controlled experimentation system and guidance on how to reliably define an OEC.

Research contributions with practical guides on how to develop an experimentation system have previously been published both by Microsoft [27], [28] and Google [29]. The Return on Investment (ROI) of controlled experimentation has been discussed a number of times in the literature [23], [27]. However, the count of companies that successfully developed an experimentation culture and became data-driven remains low and limited to other web service companies such as Facebook, Google, Booking, Amazon, LinkedIn, Etsy, Skyscanner [10], [28]. We believe that the reason for this unsuccessful adoption of continuous experimentation resides in the lack of knowledge on how the transition can be done in practice. Companies have the necessary instrumentation in place [30], are able to gather and analyze product data, but they fail to efficiently utilize it and learn from it [11].

The research contributions from Google and Microsoft provide guidance on how to start developing the experimentation platform. However, they do not provide guidance on which R&D activities to prioritize in order to incrementally scale the experimentation across the organization. This technical research contribution is aiming to address this gap and provide guidance on how to evolve from a company with data to a data-driven company. We focus on technical challenges (e.g. the necessary platform features that are required for successful scaling) as well as the organizational aspects (e.g. how to integrate data scientists in product teams) and business aspects (e.g. how to develop an Overall Evaluation Criteria). This leads to the following research question:

RQ: "*How to evolve controlled experimentation in software-intensive companies in order to become data-driven at scale?"*

To address this research question, we conducted a mixed methods study of how continuous experimentation scaled at Microsoft. We describe the research method in detail next.

III. METHOD

This research work is an inductive case study and was conducted in collaboration with the Analysis and Experimentation (A&E) team at Microsoft. The inspiration for the study originates from an internal model used at A&E, which is used to illustrate and compare progress of different product teams on their path towards data-driven development at scale. The study is based on historical data points that were collected over a period of two years and complemented with a series of semi-structured interviews, observations, and meeting participations. In principle, it is an in-depth and single case study [31], however, our participants are from different organizational units and product teams with fundamentally different product and service offerings. Several of the participants worked in other data-driven organizations before joining the Microsoft A&E team. The A&E team provides a platform and service for running controlled experiments for customers. Its data scientists, engineers and program managers are involved with partner teams and departments across Microsoft on a daily basis. The participants involved in this research work are primarily collaborating with the following Microsoft product and services teams: Bing, Cortana, Office, MSN.com, Skype and Xbox.

A. Data Collection

The data collection for this research was conducted in two streams. The first stream consisted of collection of archival data on past controlled experiments conducted at Microsoft. The first author of this paper worked with the Microsoft Analysis & Experimentation team for a period of 10 weeks. During this time, he collected documents, presentations, meeting minutes and other notes available to Microsoft employees about the past controlled experiments, the development of the experimentation platform and organizational developments conducted at Microsoft A&E over the last 5 years. In cumulative, we collected approximately 130 pages of qualitative data (including a number of figures and illustrations).

The second stream consisted of three parts. The first author (1) participated in weekly experimentation meetings, (2) attended internal training on controlled experimentation and other related topics, and (3) conducted a number of semi-structured interviews with Microsoft employees. In all three data collection activities, the first author was accompanied by one of the other three authors (as schedules permitted). At all meetings and training, we took notes that were shared between us at the end of each activity. The individual interviews were recorded and transcribed by the first researcher.

The second author of this paper has been working with the Analysis & Experimentation team at Microsoft for a period of six years. He was the main contact person for the other three researchers throughout the data collection and analysis period and advised the diverse selection of data

scientist, managers and software engineers that we interviewed. In total, we conducted 14 semi-structured interviews (1 woman, 13 men) using a questionnaire guide with 11 open-ended questions. The participants that work with different product teams were invited for a half an hour interview by the first two authors. The interview format started with an introduction and a short explanation of the research being conducted. Participants were then asked on their experience with conducting controlled experiments, how they document learnings from those experiments, and how their practices changed over time. We also asked for examples of successes, pitfalls, and pain points that they experience while conducting controlled experiments.

We provide a detailed list of our interviewees, their roles and their primary product teams in Table 1 below. The ones with n/a do not collaborate with product teams directly, but are rather focusing on platform development and other activities within the A&E team.

TABLE I. INTERVIEW PARTICIPANTS

	Interview details		
	Role	***Length (min)***	***Product***
1	Senior Data Scientist	45	Skype
2	Data Scientist	45	Skype
3	Principal Group Engineering Mgr.	30	n/a
4	Principal Data Scientist	30	Bing
5	Senior Software Engineer	30	n/a
6	Senior Data Scientist	45	MSN
7	Principal Data Scientist Mgr.	30	Office
8	Principal Data Scientist Mgr.	30	Office
9	Principal Data Scientist & Architect	30	Bing
10	GPM Program Manager	30	n/a
11	Principal Software Engineer	30	Bing
12	Senior Applied Researcher	30	Ads
13	Senior Program Manager	30	Bing
14	Senior Program Manager	30	Cortana

B. Data Analysis

We analyzed the collected data in two steps. First, we grouped the data that belonged to a certain product. Next, we grouped products in 4 buckets based on the number of experiments that their product teams are capable of executing per annum (i.e. 1-9, 10-99, 100-999, and 1000+). Second, and with the goal to model the evolution of continuous experimentation, we performed inductive category development [32]. In the first step, we emerged with three high level definitions of categories that represent our research interest (namely technical evolution, organizational evolution and business evolution). Next, we formulated the categories under each of the three categories by reading through the collected data and assigning codes to concepts that appeared in it. This approach is similar to the Grounded Theory approach as we didn't have preconceptions on which categories to form beforehand [33]. The final categories are visible in our model in Figure 5. To develop the content of the table, we backtracked the codes within the buckets. Using a 'venting' method, i.e. a process whereby interpretations are continuously discussed with professional colleagues, we iteratively verified and updated our theory on the content for each of the four phases of our models in Figure 5. The A&E team provided continuous feedback on the developing theory and helped to clear any discrepancies in the raw data.

C. Validity Considerations

1) Construct Validity

To improve the study's construct validity, we complemented the archival data collection activities with individual semi-structured interviews, meetings and trainings. This enabled us to ask clarifying questions, prevent misinterpretations, and study the phenomena from different angles. Meeting minutes and interview transcriptions were independently assessed by three researchers to guarantee inter-rater reliability. Since this study has been conducted in a highly data-driven company, all the participants were familiar with the research topic and expectations between the researchers and participants were well aligned. The constructed artifact was continuously validated with the A&E team members during the study.

2) External Validity

The main result of this paper details an evolution towards becoming a data-driven company as experienced at Microsoft. The first author conducted this research while collaborating with the second author who is permanently employed at the case company. This set-up enabled continuous access and insight. However, and since this approach differs from a traditional case study [31], the contributions of this paper risk being biased from this extensive inside view. The main contribution can thus not directly translate to other companies. However, we believe that the phases of our model, especially the dimension concerning the technical evolution, are similar to the ones that other software companies traverse on their path towards becoming data-driven. The 'Experimentation Evolution Model' can be used to compare other companies and advise them on what to focus on next in order to efficiently scale their data-driven practices. The embedded systems domain is one example area where companies are aiming to become data-driven and that we previously studied [34], [11], [35]. The phases of our model can be applied to this domain.

In the next section, we show the empirical data by describing four controlled experiments from different product teams.

IV. EMPIRICAL FOUNDATION

In this section, we briefly present examples of controlled experiments conducted at Microsoft. The space limitations make it difficult to show all the depth and breadth of our

empirical data. Due to this limitation, we select four example experiments. With each of them, we aim to illustrate the capabilities and limitations that product teams at Microsoft experience as they evolve their data-driven practices. We start with Office, where data-driven development is beginning to gain momentum and where the first controlled experiments were recently conducted. Next, we present an example from Xbox and an example from MSN where the experimentation is well established. Finally, we conclude the section by providing an illustrative experiment from Bing where experimentation is indispensable and deeply embedded in the teams' development process.

A. *Office Contextual Bar Experiment*

Microsoft Office is a well-known suite of products designed for increasing work productivity. Data-driven practices in Office product teams are in the early stages. The product team responsible for the edit interface in Office mobile apps recently conducted a design experiment on their Word, Excel, and PowerPoint apps. They believed that introducing a Contextual Command Bar (see Figure 1 below) would increase the engagement compared to a version of the product without the contextual bar. Their hypothesis was that mobile phone users will do more editing on the phone because the contextual command bar will improve editing efficiency and will result in increased commonality and frequency of edits and 2-week retention.

Figure 1. The "Contextual Bar" experiment on Word mobile app.

During the set-up of the experiment, the team ran into issues with measuring the number of edits. The instrumentation was still in the early stages, and the telemetry teams did not accurately log the edit events. These issues had to be fixed prior to the start of the experiment. The results of a two-week experiment indicated a substantial increase in engagement (counts of edits), but no statistically significant change in 2-week retention. The experiment provided the team with two key learnings: (1) Proper instrumentation of existing features and the new feature is essential for computing experiment metrics, (2) It is important to define global metrics that are good leading indicators and that can change in a reasonable timeframe.

B. *Xbox Deals for Gold Members*

Xbox is a well-known platform for video gaming. Experimentation is becoming well established with this product and their teams have been conducting experiments on several different features.

In one of the experiments, a product team at Xbox aimed to identify whether showing prices (original price and the discount) in the weekly deals stripe, and using algorithmic as opposed to editorial ordering of the items in the stripe impacts engagement and purchases. They experimented with two different variants. On Figure 2, we illustrate the experiment control (A) and both of the treatments (B, C).

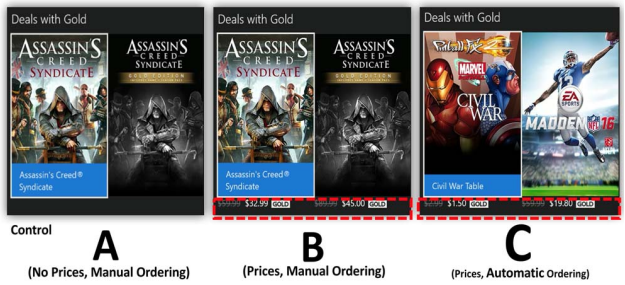

Figure 2. The "Xbox deals" experiment.

At Xbox, instrumentation is well established and a reliable pipeline for data collection exists. Metrics that measure user engagement and purchases are established and consist of a combination of different signals from the logs aggregated per user, session and other analysis units. In contrast to the Office Word experiment above, the Xbox team autonomously set-up their experiments, however, they still require assistance on the execution and monitoring of the experiment and at the analysis stage to interpret results. The two-week experiment showed that, compared to control, treatment B decreased engagement with the stripe. The purchases, however, did not decrease. By showing prices upfront treatment B provided better user experience by engaging the users who are interested in a purchase and sparing a click for those not interested. Treatment C provided even greater benefit, increasing both engagement with the stripe and purchases made. In this experiment the team learned that (1) Showing prices upfront results in better user experience, and (2) Algorithmic ordering of deals beats manual editorial ordering.

C. *MSN.com News Personalization*

In contrast to Office Word and Xbox where experimentation is primarily conducted with features focusing on design changes, teams at MSN.com experiment with most feature changes. In one of the recent experiments, they aimed to test a personalization algorithm developed within Microsoft Research for their news page. The hypothesis was that user engagement with the version that uses the machine learning personalization algorithm would increase in comparison to the manually curated articles. In contrast to Word and Xbox teams, the MSN product team autonomously set-up and execute experiments. A number of

Data Scientists were hired in their product team and they partner with the central Analysis and Experimentation team to interpret and analyze complex experiments. Contrary to the expectations, in the initial iteration of the experiment machine learning algorithm performed worse than the manual ordering. After some investigation, a bug was found in the algorithm. The bug was fixed and several subsequent iterations of the experiment were run to tune the algorithm. At the end, the algorithmic ordering resulted in a substantial lift in engagement. In Figure 3 below we show an example screenshot from one of the iterations.

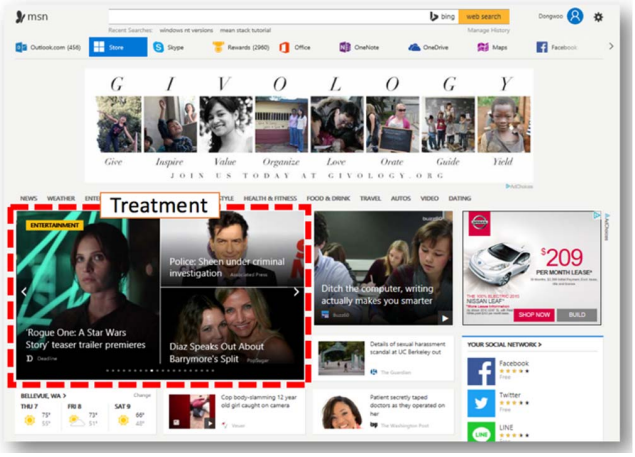

Figure 3. The "MSN.com personalization" experiment.

D. Bing Bot Detection Experiment

Bing is a search engine developed by Microsoft. On this product, several teams at Microsoft conduct over 10.000 experiments per year ranging from large design modifications to every bug fix or minor improvement. In contrast to the previous examples, teams at Bing set-up, execute and analyze experiments autonomously and without the help of the Analysis & Experimentation data scientists. At any given point in time, almost every user of the product is in at least a few of the experiments simultaneously. As users are put into more and more concurrent experiments, the chance of unexpected interactions between those experiments increases, which can lead to bad user experience and inaccurate results. Preventing interactions where possible, and detecting where not (alerts fire automatically when experiments hurt the user experience, or interact with other experiments) has been a critical element for delivering trustworthy, large-scale experimentation.

The core purpose of Bing is to provide search results to its users. Finding relevant results, however, is a computational operation that extensively consumes infrastructure capacity. One way to save on resources is to prevent computer bots from performing the actual search by e.g. returning results from a smaller in-memory index that is orders of magnitude cheaper to serve. The experiment that we briefly present in this section targeted exactly this scenario. The hypothesis was that with an improved and more pervasive bot-detection algorithm, human users will not be harmed and fewer resources will be used for the computation of search results. Conducting such experiments, however, involves the use of advanced features that prevent potentially harmful variants (see e.g. Figure 4 below) from affecting a large population by automatically checking alerts and incrementally ramping the number of users assigned to the treatment.

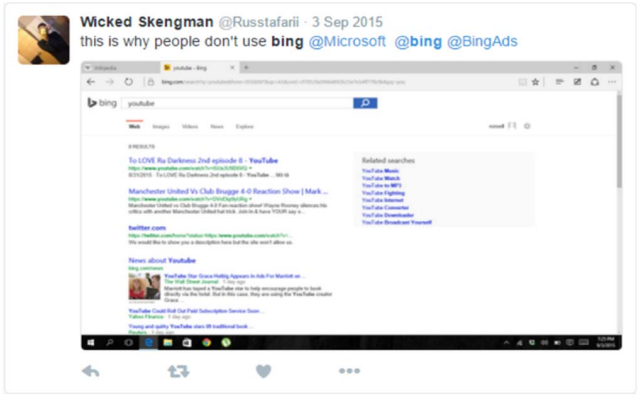

Figure 4. An archival experiment with Bing that introduced user harm.

The results of the particular experiment indicated a ~10% saving on infrastructure resources without introducing user harm. Screenshot on Figure 4 is, however, a part of another experiment with a slightly 'different' outcome.

V. The Experimentation Evolution Model

In this section, and based on the empirical presentation of products and related experiments in section IV, we present the transition process model of moving from a situation with ad-hoc data analysis towards continuous controlled experimentation at scale. We name this process the "Experimentation Evolution Model" and use this term to describe the phases that companies and their product teams follow while evolving their data-driven development practices towards continuous experimentation at scale. It is based on the empirical data collected at Microsoft and inspired by a model developed internally at A&E.

In our model, and after listing a number of prerequisites for experimentation, we present three dimensions of evolution: technical, organizational and business evolution. In the technical evolution part, we focus on the technical aspects such as the complexity of the experimentation platform, the pervasiveness of experimentation in product teams, and the overall focus of the development activities. The organizational evolution focuses on the organization of the data science teams and their self-sufficiency for experimentation. Finally, in the business evolution part, we discuss the focus of the Overall Evaluation Criteria.

The four phases of the "Experimentation Evolution Model", namely "crawl", "walk", "run" and "fly", are summarized on Figure 5 below and described in detail in the remainder of this section.

	Category/ Phase	Crawl	Walk	Run	Fly
Technical Evolution	Technical focus of product dev. Activities	**(1) Logging of signals** **(2) Work on data quality issues** **(3) Manual analysis of experiments** Transitioning from the debugging logs to a format that can be used for data-driven development.	**(1) Setting-up a reliable pipeline** **(2) Creation of simple metrics** Combining signals with analysis units. Four types of metrics are created: debug metrics (largest group), success metrics, guardrail metrics and data quality metrics.	**(1) Learning experiments** **(2) Comprehensive metrics** Creation of comprehensive set of metrics using the knowledge from the learning experiments.	**(1) Standardized process for metric design and evaluation, and OEC improvement**
	Experimentation platform complexity	**No experimentation platform** An initial experiment can be coded manually (ad-hoc).	**Platform is required** 3rd party platform can be used or internally developed. The following two features are required: • **Power Analysis** • **Pre-Experiment A/A testing**	**New platform features** The experimentation platform should be extended with the following features: • **Alerting** • **Control of carry-over effect** • **Experiment iteration support**	**Advanced platform features** The following features are needed: • **Interaction control and detection** • **Near real-time detection and automatic shutdown of harmful experiments** • **Institutional memory**
	Experimentation pervasiveness	**Generating management support** Experimenting with e.g. design options for which it's not a priori clear which one is better. To generate management support to move to the next stage.	**Experiment on individual feature level** Broadening the types of experiments run on a limited set of features (design to performance, from performance to infrastructure experiments)	**Expanding to (1) more features and (2) other products** Experiment on most new features and most products.	**Experiment with every minor change to portfolio** Experiment with any change on all products in the portfolio. Even to e.g. small bug fixes on feature level.
Organizational Evolution	Engineering team self-sufficiency	**Limited understanding** External Data Scientist knowledge is needed in order to set-up, execute and analyse a controlled experiment.	**Creation and set-up of experiments** Creating the experiment (instrumentation, A/A testing, assigning traffic) is managed by the local Experiment Owners. Data scientists responsible for the platform supervise Experiment Owners and correct errors.	**Creation and execution of experiments** Includes monitoring for bad experiments, making ramp-up and shut-down decisions, designing and deploying experiment-specific metrics.	**Creation, execution and analyses of experiments** Scorecards showing the experiment results are intuitive for interpretation and conclusion making.
	Experimentation team organization	**Standalone** Fully centralized data science team. In product teams, however, no or very little data science skills. The standalone team needs to train the local product teams on experimentation. We introduce the role of Experiment Owner (EO).	**Embedded** Data science team that implemented the platform supports different product teams and their Experiment Owners. Product teams do not have their own data scientists that would analyse experiments independently.	**Partnership** Product teams hire their own data scientists that create a strong unity with business. Learning between the teams is limited to their communication.	**Partnership** Small data science teams in each of the product teams. Learnings from experiments are shared automatically across organization via the institutional memory features.
Business Evolution	Overall Evaluation Criteria (OEC)	OEC is **defined** for the first set of experiments with a few key signals that will help ground expectations and evaluation of the experiment results.	**OEC evolves** from a few key signals to a structured set of metrics consisting of Success, Guardrail and Data Quality metrics. Debug metrics are not a part of OEC.	OEC is **tailored** with the findings from the learning experiments. Single metric as a weighted combination of others is desired.	OEC is **stable**, only periodic changes allowed (e.g. 1 per year). It is also used for setting the performance goals for teams within the organization.

Figure 5. The "Experimentation Evolution Model".

A. Prerequisites

Although most of the requirements for successful experimentation arise while we scale the number of experiments and teams, a few need to be fulfilled beforehand. To evaluate the product statistics, skills that are typically possessed by data scientists [9] are required within the company. Here, we specifically emphasize the understanding of hypothesis testing, randomization, sample size determination, and confidence interval calculation with multiple testing. For companies that lack these skills and wish to train their engineers on these topics, online resources and kits are available [36]. Combining these skills with domain knowledge about the product will enable companies to generate the first set of hypotheses for evaluation. The second major prerequisite is the availability of accessing the product instrumentation data. We discuss how to implement the instrumentation in the following sections, however, companies first need to have policies in place that allow them to provide experimenters access to the data. In some domains, this is a serious concern and needs to be addressed both on legal and technical levels.

B. Crawl Phase

As the starting point on the path towards continuous experimentation at scale, product teams start by configuring the first experiment.

1) Technical Aspect

a) Focus: The technical focus of this phase is twofold. First, and the main focus of this phase is the implementation of the logging system. In non-data driven companies, logging exists for the purpose of debugging product features [30], [37], [38]. This is usually very limited and not useful for analyzing how users interact with the products. Logging procedures in the organization need to be updated by creating a centralized catalog of events in the form of class and enumeration, and implemented in the product telemetry. The goal of such systematic logging is that a data scientist, analyst, or anyone else in the organization who is not familiar with the feature or the product itself, can understand what action or event was triggered and logged by simply looking at the name of the event. Names for events should be consistent across products and platforms so that it is easy to search for them and link them with tangible actions in the product. We name the data collected or sent from a product or feature for the purpose of data-driven

development *signals*. Examples of signals are clicks, swipes over an image, interactions with a product, time spent loading a feature, files touched, etc. Based on the complete set of signals, an analyst should be able to reconstruct the interactions that a user had with the product.

Second, any quality issues with writing and collecting signals need to be solved. The goal is to have a reliable system where events are consistently logged and repetitive actions result in identical results.

b) Experimentation platform complexity: In this initial phase, an experimentation platform is not required. With signals systematically collected, a product team can perform the first controlled experiment manually. They can do this by splitting the users between two versions of the same product and measuring how the distribution of signals differs between the versions, for example. Practitioners can use the guidance on how to calculate the statistics behind a controlled experiment in [27]. In summary, if the difference between the values for the Treatment group and the Control group is statistically significant, we conclude with high probability that the change introduced in the treatment group caused the observed effect. Conventionally, a 95% confidence interval is used.

c) Experimentation pervasiveness: Experiments in this phase are for targeted components of a product and are not pervasive. Typically, product teams should start to experiment with a feature where multiple versions are available. The main purpose of the first experiments is to gain traction and evangelize the results to obtain the necessary funding needed to develop an experimentation platform and culture within the company. As an example, product teams can start with a design experiment for which it is not *a priori* clear which of the variants is better. The results of the first experiment should not be trusted without assuring that the data quality issues have been addressed.

d) Engineering team self-sufficiency: In this initial phase, experiment set-up, execution and analysis is conducted by a data scientist team. Product teams typically do not possess the necessary skills to conduct trustworthy controlled experiments and correctly analyze the results on their own. We use the term Experiment Owner (EO) to refer to one or more individuals from the product team involved with the experiment. Experiment Owners are the individuals that understand both the product and the experiment, and are used as the main contact between the data science team and the product teams for set up and interpretation of the experiments and their results.

e) Experimentation team organization: In this phase, product teams require training and help from a standalone data scientist team. This organization of data scientists allows freedom for generating ideas and long-term thinking that are needed for development of the experimentation platform.

2) Business Aspect

a) Overall Evaluation Criteria: The aim of the "Crawl" phase is to define an OEC for the first set of experiments that will help ground expectations and evaluation of the experiment results. In concept, an OEC stands for **Overall** (in view of all circumstances or conditions), **Evaluation** (the process determining the significance, worth, or condition of something by careful appraisal and study) and **Criteria** (a standard on which a judgment or decision may be based). In practice, and for the first experiments, data scientists and Experiment Owners should collaborate on defining the OEC from a few key signals. An OEC should typically be closely related to long-term business goals and teams should be informed upfront that it will develop over time.

C. Walk Phase

After the initial logging and instrumentation have been configured, the focus of the R&D activities transitions towards defining metrics and an experimentation platform.

1) Technical Aspect

a) Focus: In contrast to the "crawl" phase where experiments were evaluated by comparing the volume and distribution of signals such as clicks and page views, the focus in this phase is on defining a set of metrics combined from those signals. Metrics are functions that take signals as an input and output a number per unit. Signals should first be categorized into classes and combined into metrics by being aggregated over analysis units. Microsoft recognizes three classes of signals for their products: **action signals** (e.g. clicks, page views, visits, etc.), **time signals** (minutes per session, total time on site, page load time, etc.), and **value signals** (revenue, units purchased, ads clicked, etc.). The units of analysis vary depending on the context and product. The following apply at Microsoft for web products: **per user** (e.g. clicks per user), **per session** (e.g. minutes per session), **per user-day** (e.g. page views per day), and **per experiment** (e.g. clicks per page view).

For other types of products, units of analysis might be different. For a well-known video-conferencing Microsoft product, "per call" is a useful unit of analysis. And by combining signals with units of analysis, simple metrics are created. Microsoft typically aims to construct four types of metrics: success metrics (the ones that we will intend to improve), guardrail metrics (constraints that are not allowed to be changed), data quality metrics (the metrics that ensure that the experiments will be set-up correctly), and debug metrics (the ones that help deeper understanding and drill down into success and guardrail metrics). A popular research contribution from Google provides practical guidance on the creation of these metrics for measuring user experience on a large scale [39].

b) Experimentation platform complexity: With more experiments being run, a need for an experimentation platform arises. Software development organizations can

decide to either start developing their own experimentation platform or utilize one of the commercial products designed for this purpose. Several third party experimentation platforms are available to software companies out of the box [40], [41], [42]. Regardless of the decision, the experimentation platform should have two essential features integrated in this phase. (1) Power Analysis and (2) pre-experiment A/A testing.

- ***Power analysis.*** This is a feature that is used to determine the minimal sample size for detecting the change in an experiment and it should be implemented early in order to automate decisions on the duration of the experiments. This will prevent some of the common pitfalls (e.g. running experiments longer than required in order to find the change or having an under-powered experiment). See [36] for details.
- ***Pre-experiment A/A testing.*** An A/A feature assigns to the treatment group the same experience as the control group is being exposed to. Data is collected and its variability is assed for power calculations and to test the experimentation system (the null hypothesis should be rejected about 5% of the time when a 95% confidence level is used). After ensuring that there is no imbalance on key OEC metrics, one of the A's is reconfigured into B – the A/B test is started on the same population.

The number of experiments in this phase is relatively low. This allows for central planning and scheduling of experiments to avoid interactions. Each experiment is still closely monitored to detect user harm or data quality issues.

c) Experimentation pervasiveness:

In contrast to the "crawl" phase where experiments were mostly with design variants or features with alternative implementations, product teams in this phase move on to different types of experiments with the same product. From design focused experiments (testing a set of design alternatives) the teams advance to performance experiments (testing performance between different variants of the same feature). Infrastructure experiments (testing resource alternatives) are another example of advancing the experimentation within the product domain.

2) Organizational Aspect

a) Engineering team self-sufficiency: In this phase, EO's responsibility for creating the experiments (scheduling the experiment, performing the power analysis etc.) is transitioning from a data science expert to a product/program manager employed in the product team. However, the execution, monitoring, and analysis of the experiments is still the responsibility of the data scientists.

b) Experimentation team organization: The results should be evangelized across the team and bad practices should be disputed (e.g. experimenting only on preview audience). We recommend embedded organization of data scientists that support product teams with increasing data quality, metrics creation and developing an Overall Evaluation Criteria. Embedded data scientists in the product teams can hold the role of Experiment Owners or work closely with other product team members that have this role. They communicate and work with the central platform team. The products within organizations will typically share certain characteristics. With this organization, a bridge in transferring learnings from one embedded data science product team to another is established.

3) Business Aspect

a) Overall Evaluation Criteria: Most investments by feature and product teams in this phase are to address data quality issues and instrumentation to build an initial set of metrics. It is important to understand and document metric movements, validate findings, and build experimentation muscle within the product and feature team. The initial Overall Evaluation Criteria should be improved with the findings from multiple experiments and supported by multiple metrics. In contrast to the "crawl" phase, the OEC will evolve from a few key signals to a structured set of metrics consisting of success metrics (the ones we intend to improve), guardrail metrics (constraints that are not allowed to be changed) and data quality metrics (the metrics that ensure that the experiments were set-up correctly and results can be trusted). It is very important to work close with many product team members and reach agreement on the OEC. When disagreements occur, the OEC should be backtracked and concerns addressed.

D. Run Phase

In the Run Phase, product teams ramp up the number of experiments and iterate quickly with the purpose of identifying the effect of the experiments on the business.

1) Technical Aspect

a) Focus: In the "walk" phase, product teams started to merge signals into metrics. In the "Run" phase, however, these metrics should evolve and become comprehensive. Metrics should evolve from counting signals to capturing more abstract concepts such as "loyalty" and "success", closely related to long-term company goals [43]. To evaluate the metrics product teams should start running learning experiments where a small degradation in user experience is intentionally introduced for learning purposes (e.g. degradation of results, slow down of a feature). With such learning experiments, teams will have a better understanding of the importance of certain features and the effect that changes have on the metrics. Knowingly hurting users slightly in the short-term (e.g., in a 2- week experiment) enables teams at Microsoft to understand fundamental issues and thereby improve the experience in the long-term [28].

b) Experimentation platform complexity: To scale above 100 data-driven experiments per year, the power analysis and pre-experiment A/A features that were implemented in the "Walk" phase will not be sufficient. The experimentation platform needs to be extended with

additional features that will both (1) prevent incidents and (2) increase the efficiency of product teams by automating certain aspects of the workflow. We describe the new features next:

- ***Alerting.*** With an increasing number of experiments, having a manual overview review of metric movements will become a resource-demanding task for Experiment Owners. Automated alerting should be introduced together with the ability to divert traffic to control if an emergency situation occurs (e.g. a decrease of an important metric). The naïve approach to alerting on any statistically significant negative metric changes will lead to an unacceptable number of false alerts and make the entire alerting system overloaded and hard to interpret. Detailed guidance on how to avoid this situation and develop alerting that works is available in [28].
- ***Control of carry-over effects.*** Harmful experiments have an effect on the population that may carry over into the follow-up experiments and cause biased results. A feature that re-randomizes the population between experiments should be implemented in order to prevent a high concentration of biased users in either treatment or control.
- ***Experiment iteration support.*** This is a feature that enables re-iteration of an experiment. Initially, experiments in this phase should start on a small percentage of traffic (e.g. 0.5% of users assigned to treatment). The reason is that, as it gets easier to configure and start an experiment, the risk of user harm also increases (changes to production software risk the introduction of degradations). Over time, the percentage should automatically increase (by e.g. running a new iteration of the experiment with a higher setting) if no alerts on guardrail metrics were triggered beforehand. The benefit of this feature is twofold. First, it offers assurance that the impact of a harmful experience will be limited to a low number of users. Second, it optimizes the time to ramp to full power, which minimizes the time to analysis of experimentation results.

c) Experimentation pervasiveness:

In contrast to the "Walk" phase where experiments were conducted on a single product, in the "Run" phase companies aim to expand the scope of controlled experimentation. They can achieve this by expanding (1) to more features within the products and more importantly, (2) to other product teams. Product teams should be experimenting with every increment to their products (e.g. introductions of new features, algorithm changes, etc.). Experimenting should be the norm for identifying the value of new features as well as for identifying the impact of smaller changes to existing features. Past experiment data can be used to understand the correlation and relationship between movements in different business goals.

2) Organizational Aspect

a) Engineering team self-sufficiency: Experiment Owners that were introduced in the "Crawl" phase and the ones that were responsible for the creation of experiments in the "Walk" phase now receive the complete responsibility to execute their experiments. The execution of experiments includes running power analysis to determine treatment allocation, monitoring for bad experiments (e.g. the ones with triggered alerts), making shut-down and ramp-up decisions, and resolution of errors. However, the analysis of results should still be supervised by the data scientists.

b) Experimentation team organization: We recommend to keep a partnership approach to the arrangement of data scientist teams by assigning a fixed number of data scientists to work with product teams (they are employed in the product teams directly). They review experiments, decide on the evaluation criteria, and are trained by the central platform data science team to become local operational data scientists capable of setting-up the experiments, executing them, and resolving basic alerts.

3) Business Aspect

a) Overall Evaluation Criteria: The purpose of this phase is to tailor OEC using the knowledge obtained from the learning experiments. Typically, and as presented in the "Walk" phase, OEC will be a combination of success, guardrail and data quality metrics. In the "Run" phase, however, it will be evolved to capture concepts such as "loyalty" and "success", and corrected with the findings from learning experiments. Selecting a single metric, possibly as a weighted combination of objectives is highly desired. The reason for that is that (1) single metric forces inherent tradeoffs to be made once for multiple experiments and (2) it aligns the organization behind a clear objective. A good practice in this phase is to also start accumulating a corpus of experiments with known outcomes and re-run the evaluation every time changes are introduced to an OEC. A good OEC will correctly determine the outcome.

E. Fly Phase

In the "Fly" phase, controlled experiments are the norm for every change to any product in the company's portfolio. Such changes include not only obvious and visual changes such as improvements of a user interface, but also subtler changes such as different machine learning and prediction algorithms that might affect ranking or content selection. However, with such pervasiveness, a number of new features are needed in the experimentation platform and new responsibilities are assigned to experiment owners.

1) Technical Aspect

a) Focus: I*:* In the previous phases, technical activities focused on implementing reliable instrumentation, creating comprehensive metrics and conducting learning experiments. In the "Fly" phase, however, we recommend to focus on standardizing the process for the evaluation and improvement of the Overall Evaluation Criteria. An OEC should be used as a foundation to define the direction for

teams developing the product. At the same time, and since customers' preferences change over time [43], a product team should invest in standardizing metric design and evaluation practices and scheduling the activities for updating the existing OEC. See [43] for details.

b) Experimentation platform complexity: In addition to the features introduced in the previous phases, advanced features such as interaction control and detection, auto-detection and shut-down of harmful experiments, and institutional memory collection are needed. These features will enable experiment owners to conduct a larger number of experiments and protect users from harm. We describe them briefly below:

- ***Interaction control and detection.*** A statistical interaction between two treatments A and B exists if their combined effect is not the same as the sum of two individual treatment effects [27]. This is a feature that prevents such experiments with conflicting outcomes to run on the same sets of users (e.g. one experiment is changing the background color to black, another the text to gray). Control for such interactions should be established and handled automatically. After detecting an interaction, the platform should send an alert to the experiment owners. Detailed guidance on how to implement this feature is available in [28].
- **Near real-time detection and automatic shutdown of harmful experiments.** In the "Run" phase alerting was configured by periodically (e.g. bi-hourly) calculating scorecards on critical guardrail metrics. In the "Fly" phase, and with thousands of experiments simultaneously active, the detection of harmful experiments should be near real-time and automatic emergency shutdown functionality should be implemented (the time to exclude users minimized).
- **Institutional Memory.** To prevent an experiment owner from repeating an experiment that someone else previously conducted, an institutional memory of experimentation should be kept. It should be searchable and include all the essential metadata of the experiment (e.g. hypothesis, experiment outcome, selected markets and execution date).

c) Experimentation pervasiveness: In contrast to the previous phases where controlled experiments were primarily used to support decisions on new feature introductions and deletions, in the "Fly" phase every small change to any product in the portfolio (e.g. a minor bug fix) should be supported by data from a controlled experiment. Advanced features described above enable product teams to experiment at this scale and expand their experimentation capabilities to cover the complete portfolio.

2) Organizational Aspect

a) Engineering team self-sufficiency: In contrast to the previous phase where the analysis of experiment results was supported by a data science team, Experiment Owners in this phase work autonomously. They create, execute and analyze the results of the experiments. The central data science team reviews experiments only on demand.

b) Experimentation team organization: The partnership approach to the arrangement of data scientist teams will be efficient at this scale. Local product teams with their operational data science teams are empowered to run experiments on their own. A central data science team should be in charge of the experimentation platform and leasing its individual data scientists to cooperate with product teams to resolve issues and share experience.

3) Business Aspect

a) Overall Evaluation Criteria: The OEC at this phase should be rather stable and well defined. The OEC is used for setting the performance goals for teams within the organization. In contrast to the previous phases where the OEC was evolving, changes to the overall evaluation criteria in the "Fly" phase should occur only periodically (e.g. once per year) and follow and standardized process. This gives independent teams across the product portfolio a chance to focus their work on understanding how the features they own affect the key metrics, prioritizing their work to improve the OEC.

VI. CONCLUSIONS

Controlled Experimentation is becoming the norm in the software industry for reliably evaluating ideas with customers and correctly prioritizing product development activities [4] [5], [6], [7], [8], [21]. Previous research publications by Microsoft [27], [28], Google [29] and academia [5]–[8] reveal the essential building blocks for an experimentation platform; however, they leave out the details on how to incrementally scale (e.g. which technical and organizational activities to focus on at what phase). With our research contribution, which is based on an extensive case study at Microsoft, we aim to provide guidance on this topic and enable other companies to establish or scale their experimentation practices. Our main contribution is the "Experimentation Evolution Model". In the model, we summarize the four phases of evolution and describe the focus of technical, organizational and business activities for each of them. Researchers and practitioners can use this model to position other case companies and guide them to the next phase by suggesting the necessary features.

In future research, we plan to (1) research the impact of controlled experimentation with respect to the four phases from the "Experimentation Evolution Model" and (2), validate our model in other companies.

ACKNOWLEDGMENT

We wish to thank Brian Frasca, Ronny Kohavi and others at Microsoft that provided the input for and feedback on this research. Ronny Kohavi was the creator of a similar model used internally in A&E that was used as inspiration for this work. The first author of this paper would also like to thank the A&E team for the invaluable opportunity to work with them during his research internship at Microsoft.

REFERENCES

[1] D. J. Patil, "Building Data Science Teams," *Oreilly Radar*, pp. 1–25, 2011.

[2] A. Fabijan, H. H. Olsson, and J. Bosch, "Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review," in *Software Business, ICSOB 2015*, 2015, vol. 210, pp. 139–153.

[3] G. Westerman, M. Tannou, D. Bonnet, P. Ferraris, and A. McAfee, "The Digital Advantage: How Digital Leaders Outperform their Peers in Every Industry," *MIT Sloan Manag. Rev.*, pp. 1–24, 2012.

[4] R. Kohavi and R. Longbotham, "Online Controlled Experiments and A/B Tests," in *Encyclopedia of Machine Learning and Data Mining*, no. Ries 2011, 2015, pp. 1–11.

[5] H. H. Olsson and J. Bosch, *The HYPEX model: From opinions to data-driven software development*. 2014.

[6] H. H. Olsson and J. Bosch, "Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation," in *Lecture Notes in Business Information Processing*, 2015, vol. 210, pp. 154–166.

[7] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "Building Blocks for Continuous Experimentation," *Proc. 1st Int. Work. Rapid Contin. Softw. Eng.*, pp. 26–35, 2014.

[8] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "The RIGHT model for Continuous Experimentation," *J. Syst. Softw.*, vol. 0, pp. 1–14, 2015.

[9] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "The emerging role of data scientists on software development teams," in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, 2016, no. MSR-TR-2015-30, pp. 96–107.

[10] P. Rodríguez *et al.*, "Continuous Deployment of Software Intensive Products and Services: A Systematic Mapping Study," *J. Syst. Softw.*, 2015.

[11] A. Fabijan, H. H. Olsson, and J. Bosch, "The Lack of Sharing of Customer Data in Large Software Organizations: Challenges and Implications," in *17th International Conference on Agile Software Development XP2016*, 2016, pp. 39–52.

[12] R. C. Martin, *Agile Software Development, Principles, Patterns, and Practices*. 2002.

[13] H. H. Olsson, H. Alahyari, and J. Bosch, "Climbing the 'Stairway to heaven' - A mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software," in *Proceedings - 38th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2012*, 2012, pp. 392–399.

[14] S. Mujtaba, R. Feldt, and K. Petersen, "Waste and lead time reduction in a software product customization process with value stream maps," in *Proceedings of the Australian Software Engineering Conference, ASWEC*, 2010, pp. 139–148.

[15] E. M. Goldratt and J. Cox, *The Goal: A Process of Ongoing Improvement*, vol. 2nd rev. e, no. 337 p. 2004.

[16] D. Ståhl and J. Bosch, "Continuous integration flows," in *Continuous software engineering*, vol. 9783319112, 2014, pp. 107–115.

[17] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. 2011.

[18] G. Castellion, "Do It Wrong Quickly: How the Web Changes the Old Marketing Rules by Mike Moran.," *J. Prod. Innov. Manag.*, vol. 25, no. 6, pp. 633–635, 2008.

[19] The Standish Group, "The standish group report," *Chaos*, vol. 49, pp. 1–8, 1995.

[20] J. Manzi, *Uncontrolled : the surprising payoff of trial-and-error for business, politics, and society*. Basic Books, 2012.

[21] P. Bosch-Sijtsema and J. Bosch, "User Involvement throughout the Innovation Process in High-Tech Industries," *J. Prod. Innov. Manag.*, vol. 32, no. 5, pp. 1–36, 2014.

[22] H. H. H. H. Olsson and J. Bosch, "From opinions to data-driven software R&D: A multi-case study on how to close the 'open loop' problem," in *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014*, 2014, pp. 9–16.

[23] M. L. T. Cossio *et al.*, *A/B Testing - The most powerful way to turn clicks into customers*, vol. XXXIII, no. 2. 2012.

[24] R. C. Van Nostrand, "Design of Experiments Using the Taguchi Approach: 16 Steps to Product and Process Improvement," *Technometrics*, vol. 44, no. 3, pp. 289–289, Aug. 2002.

[25] H. Hohnhold, D. O'Brien, and D. Tang, "Focusing on the Long-term," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, 2015, pp. 1849–1858.

[26] R. Kohavi, A. Deng, and R. Longbotham, "Seven Rules of Thumb for Web Site Experimenters," *Kdd*, pp. 1–11, 2014.

[27] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: Survey and practical guide," *Data Min. Knowl. Discov.*, vol. 18, pp. 140–181, 2009.

[28] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1168–1176.

[29] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, 2010, p. 17.

[30] T. Barik, R. Deline, S. Drucker, and D. Fisher, "The Bones of the System: A Case Study of Logging and Telemetry at Microsoft," 2016.

[31] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, 2008.

[32] P. Mayring, "Qualitative content analysis - research instrument or mode of interpretation," in *The Role of the Researcher in Qualitative Psychology*, 2002, pp. 139–148.

[33] K. M. Eisenhardt, "Building Theories from Case Study Research.," *Acad. Manag. Rev.*, vol. 14, no. 4, pp. 532–550, 1989.

[34] J. Bosch and U. Eklund, "Eternal embedded software: Towards innovation experiment systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7609 LNCS, no. PART 1, pp. 19–31.

[35] A. Fabijan, H. H. Olsson, and J. Bosch, "Time to Say 'Good Bye': Feature Lifecycle," in *42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Limassol, Cyprus. 31 Aug.-2 Sept. 2016*, 2016, pp. 9–16.

[36] "Hypothesis Kit for A/B testing." [Online]. Available: http://www.experimentationhub.com/hypothesis-kit.html.

[37] D. Yuan, S. Park, and Y. Zhou, "Characterizing logging practices in open-source software," in *Proceedings - International Conference on Software Engineering*, 2012, pp. 102–112.

[38] Q. Fu *et al.*, "Where do developers log? an empirical study on logging practices in industry," *Companion Proc. 36th Int. Conf. Softw. Eng. - ICSE Companion 2014*, pp. 24–33, 2014.

[39] K. Rodden, H. Hutchinson, and X. Fu, "Measuring the User Experience on a Large Scale : User-Centered Metrics for Web Applications," *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, pp. 2395–2398, 2010.

[40] "Optimizely." [Online]. Available: https://www.optimizely.com/.

[41] "Mixpanel." [Online]. Available: https://mixpanel.com/.

[42] "Oracle Maxymiser." [Online]. Available: https://www.oracle.com/marketingcloud/products/testing-and-optimization/index.html.

[43] W. Wood, M. G. Witt, and L. Tam, "Changing circumstances, disrupting habits," *J. Pers. Soc. Psychol.*, vol. 88, no. 6, pp. 918–33, 2005.